

PICTURE PROCESSING TECHNIQUES

by

TIJEN MERGEN

B.S. in E.E., Boğaziçi University, 1981

Submitted to the Institute for Graduate Studies
in Science and Engineering in partial fulfillment
of the requirements for the degree of
Master of Science
in
Electrical Engineering

Bogazici University Library



39001100315384

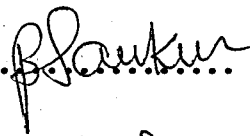
14

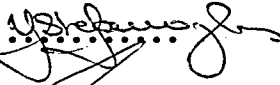
Boğaziçi University

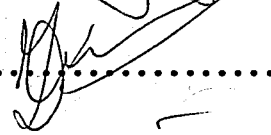
1984

PICTURE PROCESSING TECHNIQUES

Approved by

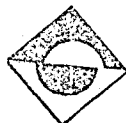
Doç.Dr.Bülent Sankur.....

Doç.Dr.Yorgo Istefanopulos.....

Dr.Irini Dimitriadis.....

Dr.Avni Morgül.....

Date of Approval :



ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis supervisor, Doç.Dr.Bilent Sankur for his advice and helpful criticisms that have been valuable in all parts of this work. I would like to thank Doç.Dr.Yorgo İstefanopulos for his kind help and understanding throughout my undergraduate and graduate study, as well as my thesis. My sincere appreciation goes to Cem Mergen for his unmeasurable help and understanding. I also wish to thank Komili Pazarlama ve Dış Ticaret A.Ş. who has provided me a private microcomputer during my whole study. Special thanks to all Mergens and Önözes.

PREFACE

The field of image processing especially digital picture processing has grown considerably during the past decade with the increased utilization of imagery in myriad applications coupled with improvements in the size, speed and cost effectiveness of digital computers and related signal processing technologies. Image processing has found a significant role in scientific, industrial, biomedical, space and governmental applications. Such applications include the digital transmission of space craft, imagery and personal telephone carrier, television, the resolution improvement of electron microscope, images and the compensation sensor and transmission errors of pictures transmitted from deep-space-probes, the automatic classification of terrain and detection of resources from earth resources satellite pictures, the formation and enhancement of biomedical imagery, including radiographs, thermograms and nuclear scanned images, automatic map making from serial photographs and the detection of cracks and flows in machine parts from industrial radiographs. In the future image processing will no doubt be utilized to a greater extent to aid medical practitioners in the detection and diagnosis of diseases from biomedical images. Industrial application should abound, image processing systems will analyze scenes from the "eyes" of industrial automations to control their actions.

The purpose of this study is to review the techniques, tools and applications of image processing. Among them two major techniques have been examined. The first one is the picture compression and coding techniques

which have a use of minimizing the number of bits required to transmit or store an image as much as possible. The second one is the edge detection techniques which give the description, i.e. certain features, of images.

This study is divided into three parts. Chapter 1 deals with mathematical models and representation of images. Topics covered include the problems in image processing like image enhancement and methods to solve these problems. Unitary transforms, autoregressive or state variable models for images or linear prediction models used in many problems, etc. are examined. Chapter 2 deals with image data compression and coding techniques. In this chapter different alternatives to quantizers are given and various Pulse Code Modulation (PCM), Differential PCM (DPCM) and adaptive DPCM (ADPCM) methods are examined. Their performances are evaluated due to SNR. A package program has been developed to simulate all these techniques. In the last part (Ch.3 and Ch.4) edge detection techniques are examined. Three typical and the most popular preprocessing algorithms are chosen among various different edge detection schemes. Different decision and postprocessing algorithms are added to the configuration of the schemes. A package program has been developed to simulate all these schemes. Evaluation and comparison is done according to the eight qualitative or quantitative performance criteria which newly derived. In chapter 4 all evaluations and comparison results are given by graphics and simulation results.

It is impossible even in this relatively long study to cover all the aspect of image processing. The reader is referred to the bibliography at the end of this work for further study.

ABSTRACT

Image processing techniques find applications in many areas, chief among which are image enhancement, picture compression, pattern recognition, efficient picture coding and image understanding. In this master thesis, some important aspects of image processing techniques which are listed below are discussed

1 - The mathematical models used in picture processing applications like fast unitary transforms, autoregressive and state variable models, linear prediction models. Applications of these models in several image processing problems, including image restoration, smoothing, enhancement, data compression and detection.

2 - A large variety of image data compression and coding techniques. Some simple sampling and quantization techniques, Pulse Code Modulation (PCM), Differential PCM (DPCM), predictive coding and adaptive coding techniques.

3 - Major edge detection techniques and alternative algorithms to the three different levels of edge detection schemes (preprocessing, labeling, postprocessing algorithms). In addition to the discussion of these three aspects, two package programs have been developed to simulate data compression and edge detection techniques by using a microcomputer. For evaluation of edge detection schemes, some qualitative and quantitative performance criteria are generated and comparison of three edge detection schemes is also done due to these criteria.

ÖZETÇE

Resim iletiřimi teknikleri birok alanda uygulanım olanađı bulmuřlardır. rneđin, resim vurgulama, resim sıkıřtırma, rnt tanıma, resim kodlama veya resim anlama. Bu tezde resim iletiřimi tekniđi ile ilgili bazı noktalar incelenmiřtir.

1 - Hızlı bılımsel dnřmler, AR modeli, durum deđiřkeni modeli, dođrusal ngrc modelleri gibi resim iletiřimi konusunda ok uygulaması olan matematiksel modeller ve bu modellerin resim iletiřimi problemlerinde rneđin, resim restorasyonu, dzeltme, vurgulama, veri sıkıřtırma, dallarında kullanılması.

2 - Deđiřik resim sıkıřtırma ve kodlama yntemleri. Basit rnekleme ve niceleme yntemlerinin yanısıra PCM, DPCM, ngrcl kodlama ve uyarlamalı kodlama yntemleri

3 - nemli ayırıt sezme yntemleri ve bu yntemlerin u iřlemi iin deđiřik algoritmalar (niřleme, deđerleme, soniřleme).

Bir mikro bilgisayar kullanılarak resim sıkıřtırma ve ayırıt sezme yntemleri ile ilgili iki ayrı paket program geliřtirilmiřtir. Ayırıt sezme yntemlerinin deđerlendirilmesi ve birbirleri ile karřılařtırılması yapılmıř ve bu amaca ynelik bařarı kriterleri tanımlanmıřtır.

LIST OF FIGURES

Fig 2.1	Process of quantizing and coding	30
Fig 2.2	I/O characteristics of 3-bit quantizer	33
Fig.2.3	Two common uniform quantizer characteristics	36
Fig 2.4	Additive noise model of quantization	36
Fig 2.5	Block diagram of logarithmic quantizer	42
Fig 2.6	Density function and quantizer characteristic for Laplacian density function and 3-bit quantizer	46
Fig 2.7	Optimum step-sizes for a uniform quantizer for Laplace, Gamma and Gaussian density functions	50
Fig 2.8	Adaptive quantization	51
Fig 2.9	General representation of feed-forward quantizers with step-size adaptation	53
Fig 2.10	General feed-forward adaptive quantizer with a time-varying gain	53
Fig 2.11	General feed-back adaptation of time-varying gains	57
Fig 2.12	General feed-back adaptation of step-sizes	57
Fig 2.13	I/O characteristics of Jayant's quantization scheme	60
Fig 2.14	Histogram of step-sizes in adaptive quantization with a Gauss-Markov input	61
Fig 2.15	General shape of optimal multiplier function; $B > 2$	62
Fig 2.16	General differential quantization	66
Fig 2.17	A PCM transmission system	76
Fig 2.18	A PCM picture transmission system	76
Fig 2.19	The sampling process	78
Fig 2.20	Optimal SNR gain G vs. no. of predictor coefficients	83
Fig 2.21	ADPCM system with feed-forward adaptive quant.	85
Fig 2.22	ADPCM system with feed-back adaptive quantization	86

Fig 2.23	ADPCM system with both adaptive quantization and adaptive prediction	88
Fig 2.24	Predictor gains vs.no.of predictor coefficients	90
Fig 2.25	Four types of input sequences used in the program each of which consists of 64 samples	93
Fig 2.26	Spectra of three quantized signals individually and their averages	95
Fig 2.27	SNR vs.quantizer step-sizes for uniform quantization with mid-riser values	96
Fig 2.28	Uniform quantization with 3-bit uniform mid-riser values	97
Fig 2.29	SNR vs.no.of bits for optimum quantization with Gaussian densities	100
Fig 2.30	Optimum quantization using 3-bit optimum quantizer with Gaussian densities	100
Fig 2.31	Quantization method with summer	102
Fig 2.32	Differential quantization with summer using 3-bit uniform mid-riser quantizer	104
Fig 2.33	SNR vs.no.of bits for differential quantization with a summer and uniform midriser quantizer	106
Fig 2.34	Logarithmic quantization with a 3-bit uniform mid-riser quantizer	108
Fig 2.35	SNR vs.no.of bits for logarithmic quantization	108
Fig 2.36	SNR vs. M value for feed-forward adaptive quant. using 4-bit uniform mid-riser quantizer	110
Fig 2.37	Feed-forward adaptation with 4-bit uniform mid-riser quantizer	110
Fig 2.38	SNR vs.no.of bits for feed-forward adptive quant. using uniform mid-riser quantizer for M=8,M=16	111
Fig 2.39	DPCM quantization of 1st and 2nd order AR sequence	113
Fig 2.40	DPCM quant. with 4-bit uniform mid-riser quant.and a 1st order predictor for a 1st order AR sequence	114
Fig 2.41	SNR vs.no.of bits for DPCM quantization with unif. quantizer for 1st order AR sequence with 1st and 3rd order predictor	116

Fig 3.1	Synthetic highs system	120
Fig 3.2	Thresholded edge detection scheme	134
Fig 3.3	Examples of horizontal and vertical differencing edge detection	135
Fig 3.4	Examples of Laplacian edge detection	139
Fig 3.5	Examples of Argyle and Macleod mask edge detection	141
Fig 3.6	Examples of Roberts squareroot and magnitude cross difference edge detection	144
Fig 3.7	Numbering for 3*3 edge detection operators	145
Fig 3.8	Examples of Sobel edge detection	145
Fig 3.9	Examples of Kirsch edge detection	146
Fig 3.10	Kirsch compass gradient masks	148
Fig 3.11	Prewitt compass gradient masks	150
Fig 3.12	Examples of logarithmic edge detection	151
Fig 3.13	Differences of running averages and product of these differences for a noisy 1-dim. step edge and for a TV line by Rosenfeld	153
Fig 3.14	Product of differences for a set of noisy vertical edges and a set of noisy circles	155
Fig 3.15	3- and 5-level simple masks	159
Fig 3.16	8 principle direction on a 3*3 grid	159
Fig 3.17	Block diagram of the proposed edge detection system	159
Fig 3.18	Performance of the edge detection system using 5-level simple directional masks	161
Fig 3.19	Indications of edge location by Pratt	164
Fig 3.20	Edge location figure of merit as a function of SNR and edge width	166
Fig 3.21	Lay-out of a test image	169
Fig 3.22	A pictorial display of the steps taken in computing the edge detection performance parameters	172
Fig 3.23	The edge detection performance parameters as a function of nominal contrast	174

Fig 3.24	Behaviour of the edge detection schemes in terms of the variation of parameters 1 and 2 with changes in nominal contrast, taken at various orientations.	176
Fig 4.1	The regions of three different test images	180
Fig 4.2.a	Vertical test image with assigned characters and intensity values with a contrast of C=5	181
Fig 4.2.b	Diagonal test image with assigned characters and intensity values with a contrast of C=5	182
Fig 4.2.c	Circular test image with assigned characters and intensity values with a contrast of C=5	183
Fig 4.3.a	Vertical test image with assigned characters and intensity values with a contrast of C=30	184
Fig 4.3.b	Diagonal test image with assigned characters and intensity values with a contrast of C=30	185
Fig 4.3.c	Circular test image with assigned characters and intensity values with a contrast of C=30	186
Fig 4.4	The outputs (edge maps) of three different methods of adaptive thresholding	191
Fig 4.5	Block diagram of the edge detection scheme using Kirsch masks	192
Fig 4.6	Outputs of the edge detection scheme with Kirsch Masks and fixed thr. applied on a vertical edge	193
Fig 4.7	Outputs of the edge detection scheme with Kirsch masks and fixed thr. applied on a diagonal edge	194
Fig 4.8	The outputs of the edge detec. scheme with Kirsch masks and fixed thr. applied on a circular edge	195
Fig 4.9	The outputs of the edge detec. scheme with Kirsch masks and adaptive thr. applied on a vertical image	196
Fig 4.10	The outputs of the edge detec. scheme with Kirsch masks and adaptive thr. applied on a diagonal image	197
Fig 4.11	The outputs of the edge detec. scheme with Kirsch masks and adaptive thr. applied on a circular image	198
Fig 4.12	Block diagram of the edge detection scheme using Sobel operator	200
Fig 4.13	Simulation results of the Sobel operator and fixed threshold applied on a vertical edge	201

Fig 4.14	Simulation results of the Sobel operator and fixed threshold applied on a diagonal edge	202
Fig 4.15	Simulation results of the Sobel operator and fixed threshold applied on a circular edge	203
Fig 4.16	Simulation results of the Sobel operator and locally adaptive threshold applied on a vertical edge	204
Fig 4.17	Simulation results of the Sobel operator and locally adaptive threshold applied on a diagonal edge	205
Fig 4.18	Simulation results of the Sobel operator and locally adaptive threshold applied on a circular edge	206
Fig 4.19	Block diagram of the edge detection scheme using Rosenfelds difference equation	208
Fig 4.20	Simulation results of the Rosenfeld difference eqs.with fixed thr.using vertical edge	209
Fig 4.20	(continued)	210
Fig 4.21	Simulation results of Rosenfelds product of diff. with fixed thres.applied on a vertical edge	211
Fig 4.21	(continued)	212
Fig 4.22	Simulation results of Rosenfelds algorithm with threshold using a diagonal edge	213
Fig 4.22	(Continued)	214
Fig 4.23	Simulation results of Rosenfelds algorithm with fixed thr.using a circular edge	215
Fig 4.23	(Continued)	216
Fig 4.24	% error vs.contrast for four different edge detec. schemes using Sobel operator w/o existance of noise	222
Fig 4.25	% error vs.contrast for four different edge detec. schemes using Sobel operator with noise of $\sigma=40$	224
Fig 4.26	Simulation result of Sobel operator:Original image, thresholded image with fixed threshold and output of postprocessing operation respectively	225
Fig 4.26	(continued) b)with noise, $\sigma=40, C=20$	226
Fig 4.26	(continued) c)with noise, $\sigma=100, C=20$	227
Fig 4.27	Graphs of % error vs contrast for fixed and adaptive thr.with and w/o postprocessing,no noise	229

Fig 4.28	Graphs of % error vs.contrast for fixed and adap. thr.with and w/o postpro.with noise of $\sigma =40$	231
Fig 4.29	Simulation results of Kirsch masks:Original image, thresholded image with fixed threshold and output of postprocessing operation respectively	232
Fig 4.29	(continued) b)with noise, $\sigma =40, C=20$	233
Fig 4.29	(continued) c)with noise, $\sigma =100, C=20$	234
Fig 4.30	Graphs of % error vs. contrast for fixed thr.,w/o postprocessing and w/o noise for Rosenfelds alg.	235
Fig 4.31	Graphs of % error vs.contrast for fixed thr.,w/o postpro.and with noise ($\sigma =40$) for Rosenfelds alg.	237
Fig 4.32	Simulation results of Rosenfelds algorithm:Original image, thresholded image using fixed threshold and output of postprocessing operation, respectively	238
Fig 4.32	(continued) b)with noise, $\sigma =40, C=20$	239
Fig 4.32	(continued) c)with noise, $\sigma =100, C=20$	240
Fig 4.33	Comparison of schemes: % error vs contrast w/o the existance of noise	242
Fig 4.34	Comperison of schemes: % error vs contrast with the existance of noise ($\sigma =40$)	243
Fig 4.35	Graphs of % error vs.noise variance for Sobel Op.	245
Fig 4.36	Simulation results of Sobel operator:Original image,thresholded image with fixed threshold and output of postprocessing operation, respectively	246
Fig 4.36	(continued) b)with noise, $\sigma =10, C=20$	247
Fig 4.36	(continued) c)with noise, $\sigma =50, C=20$	248
Fig 4.37	Graphs of % error vs.noise variance for Kirsch masks	250
Fig 4.38	Simulation results of Kirsch masks:Original image, thresholded image using fixed threshold and output of postprocessing operation respectively	251
Fig 4.38	(continued) b)with noise, $\sigma =10, C=20$	252
Fig 4.38	(continued) c)with noise, $\sigma =50, C=20$	253
Fig 4.39	Graphs of % error vs.noise variance for Rosenfeld difference equations	255

Fig 4.40	Simulation results of Rosenfelds algorithm:Original image,thresholded image with fixed threshold and output of postprocessing respectively	256
Fig 4.40	(continued) b)with noise, $\sigma=10, C=20$	257
Fig 4.40	(continued) c)with noise, $\sigma=50, C=20$	258
Fig 4.41	Comparison of schemes: % error vs noise var.	259
Fig 4.42	Graphs of P parameter vs.contrast for Sobel oper.	261
Fig 4.43	Graphs of P parameter vs.contrast for Kirsch Masks	262
Fig 4.44	Graphs of P parameter vs.contrast for Rosenfeld difference equations	266
Fig 4.45	Comparison of schemes: P parameter vs.contrast	267
Fig 4.46	Graphs of mean square distance vs.contrast for Sobel operator,w/o noise	269
Fig 4.47	Graphs of mean square distance vs.contrast for Sobel operator, with noise ($\sigma=40$)	270
Fig 4.48	Graphs of mean square distance vs.contrast for Kirsch masks, w/o noise	271
Fig 4.49	Graphs of mean square distance vs.contrast for Kirsch masks, with noise ($\sigma=40$)	272
Fig 4.50	Graphs of mean square distance vs.contrast for Rosenfelds algorithms, w/o noise	274
Fig 4.51	Graphs of mean square distance vs.contrast for Rosenfeld's algorithms with noise, ($\sigma=40$)	275
Fig 4.52	Graphs of missing edge points vs.contrast for Sobel operator,w/o noise	276
Fig 4.53	Graphs of missing edge points vs.contrast for Sobel operator,with noise ($\sigma=40$)	278
Fig 4.54	Graphs of missing edge points vs.contrast for Kirsch masks,w/o noise	279
Fig 4.55	Graphs of missing edge points vs.contrast for Kirsch masks,with noise ($\sigma=40$)	280
Fig 4.56	Graphs of missing edge points vs.contrast for Rosenfeld's algorithm,w/o.noise	282
Fig 4.57	Graphs of missing edge points vs.contrast for Rosenfeld's algorithm,with noise	283

Fig 4.58	Comparison of three schemes:Missing edge points vs.contrast,w/o noise	284
Fig 4.59	Comparison of three schemes:Missing edge points vs.contrast,with noise ($\sigma=40$)	285
Fig 4.60	Behaviour of the edge detection scheme using Sobel operator on curved edges with C=2	291
Fig 4.61	The behav.of Kirsch masks on curved edges with C=2	292
Fig 4.62	The behav.of Rosenfeld's alg.on curved edges with C=2	293
Fig 4.63	The behav.of the Sobel op.on curved edges with C=20	294
Fig 4.64	The behav.of the Kirsch masks on curved edg., C=20	295
Fig 4.65	The behav.of Rosenfeld's algorithm on curved edges with C=20	296
Fig 4.66	The ability of the Sobel operator to detect edges sharply with C=30	297
Fig 4.67	The ability of Kirsch masks to detect edges sharply with C=30	298
Fig 4.68	The ability of Rosenfeld's algor.to detect edges sharply with C=30	299

LIST OF TABLES

Table 1.1	Typical problems in image processing	6
Table 1.2	Desirable properties of image processing algorithms	6
Table 2.1.a	I/O values of uniform quantizers with mid-tread values ($m=0, \sigma=1$)	34
Table 2.1.b	I/O values of uniform quantizers with mid-riser values ($m=0, \sigma=1$)	35
Table 2.2.a	Optimum quantizers with Gaussian density	47
Table 2.2.b	Optimum quantizers with Laplacian density	48
Table 2.2.c	Optimum quantizers with Gamma density	49
Table 2.3	Adaptive 3-bit quantization with feed-forward adaptation	56
Table 2.4	Step-size multipliers for $B=2,3,4$	61
Table 2.5	Numerical outputs of the 3-bit uniform quantization	98
Table 2.6	SNR values for 3 different types of 2- and 3-bit quantizers	99
Table 2.7	Numerical output of the 3-bit optimum quantizer with 101 gaussian densities	101
Table 2.8	Numerical output of differential quantization with a 105 summer and a 3-bit uniform quantizer	105
Table 2.9	Numerical output of logarithmic quantization with 3-bit uniform quantizer	107
Table 2.10	Numerical output of feed-forward adaptive quant. using 4-bit uniform quantizer	112
Table 2.11	Numerical output of DPCM quantization using 4-bit uniform quantizer and first order linear predictor	115
Table 3.1	Evaluation results of the performance for the detector schemes	173
Table 4.1	The value of P parameters for Sobel operator with adaptive and fixed threshold ($\sigma=40$)	260
Table 4.2	The value of P parameters for Kirsch masks with adaptive and fixed threshold ($\sigma=40$)	264
Table 4.3	The value of P parameters for Rosenfeld's algor. with fixed threshold ($\sigma=40$)	265

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
PREFACE	iv
ABSTRACT	vi
OZETCE	vii
LIST OF FIGURES	viii
LIST OF TABLES	xvi
I. APPLICATIONS AND MATHEMATICAL MODELS FOR IMAGE PROCESSING ...	1
1.1 INTRODUCTION	1
1.2 TYPICAL PROBLEMS IN IMAGE PROCESSING	2
1.3 TWO DIMENSIONAL SIGNAL PROCESSING	3
1.4 LINEAR TRANSFORM PROCESSING OF IMAGES	5
1.5 IMAGE REPRESENTATION BY ONE-DIMENSIONAL STOCHASTIC	9
PROCESSES	
1.5.1 AUTOREGRESSIVE REPRESENTATION	10
1.5.2 STATE VARIABLE MODELS	11
1.5.3 NONCAUSAL MODELS	12
1.6 LINEAR PREDICTION MODELS IN TWO DIMENSIONS	12
1.6.1 CAUSAL PREDICTION	13
1.6.2 SEMICAUSAL PREDICTION	14
1.6.3 NONCAUSAL PREDICTION	14
1.7 IMAGE QUALITY CRITERIA	15
1.7.1 MEAN-SQUARE ERROR CRITERIA	16
1.7.2 A DISTORTION MEASURE	17
1.8 IMAGE PROCESSING PROBLEMS AND APPLICATIONS	17

1.8.1	SMOOTHING	18
1.8.2	IMAGE ENHANCEMENT	18
1.8.3	IMAGE RESTORATION	20
1.8.4	IMAGE DATA COMPRESSION -- PICTURE CODING	20
1.8.5	EDGE EXTRACTION, PATTERN RECOGNITION	25
1.9	CONCLUSION AND SUMMARY	26
II.	PICTURE CODING TECHNIQUES	27
2.1	QUANTIZATION	28
2.1.1	ZERO MEMORY QUANTIZATION	32
2.1.2	ADAPTIVE QUANTIZATION	51
2.2	GENERAL THEORY OF SEQUENTIAL OR DIFFERENTIAL QUANTIZATION.	65
2.3	LINEAR PREDICTION	69
2.4	DIFFERENTIAL PULSE CODE MODULATION (DPCM)	76
2.4.1	PULSE CODE MODULATION	76
2.4.2	DIFFERENTIAL PULSE CODE MODULATION (DPCM)	81
2.4.3	DPCM WITH ADAPTIVE QUANTIZATION	85
2.4.4	DPCM WITH ADAPTIVE PREDICTION	87
2.5	RESULTS OF THE COMPUTER SIMULATION	91
2.5.1	A PACKAGE PROGRAM OF QUANTIZATION,PCM,DPCM TECHNIQUES.	91
2.5.2	UNIFORM QUANTIZATION WITH SIGNAL VARIANCE ADAPTATION .	96
2.5.3	NONUNIFORM OPTIMUM QUANTIZATION WITH SIGNAL	99
	VARIANCE ADAPTATION	
2.5.4	DIFFERENTIAL QUANTIZATION WITH A SUMMER	102
2.5.5	LOGARITHMIC QUANTIZATION	106
2.5.6	FEED-FORWARD ADAPTIVE QUANTIZATION	109
2.5.7	DIFFERENTIAL PULSE CODE MODULATION (DPCM).....	111
2.6	CONCLUSION AND SUMMARY	116

III.EDGE DETECTION AND THEIR COMPARISON	118
3.1 INTRODUCTION	118
3.2 FUNDAMENTALS OF EDGE DETECTION	119
3.2.1 TEXTURAL AREAS	119
3.2.2 GRADIENT AND LAPLACIAN	120
3.2.3 DIRECTIONAL DERIVATIVES	122
3.2.4 TWO DIFFERENT APPROACHES TO EDGE DETECTION	125
3.2.5 THRESHOLDING	131
3.3 EDGE DETECTION TECHNIQUES	134
3.3.1 LINEAR EDGE DETECTION	135
3.3.2 NONLINEAR EDGE DETECTION	143
3.4 QUANTITATIVE EVALUATION OF EDGE DETECTION SCHEMES	163
3.4.1 EDGE DETECTION PERFORMANCE BY PRATT	163
3.4.2 QUANTITATIVE EVALUATION BY J.R.FRAN AND E.S.DEUTSCH ..	167
3.4.3 QUALITATIVE COMPARISON OF DIRECTIONAL LAPLACIAN	177
MASKS, SOBEL AND PREWITT OPERATORS BY E.ALPAZLAN	
IV. COMPUTER SIMULATION RESULTS OF EDGE DETECTION SCHEMES	178
AND FURTHER RESEARCH ON THEIR COMPARISON	
4.1 INTRODUCTION	178
4.2 TEST IMAGES	179
4.3 EDGE DETECTION SCHEMES	187
4.4 PERFORMANCE CRITERIA	218
4.5 EVALUATION OF RESULTS	221
4.6 CONCLUSION	300
V. CONCLUSION AND SUMMARY	303
APPENDIX A	306
APPENDIX B	333
BIBLIOGRAPHY	360

I. APPLICATIONS AND MATHEMATICAL MODELS FOR IMAGE PROCESSING

1.1 INTRODUCTION

The steady growth of modern communication requirements has resulted in a steady increase in the volume of pictorial data that must be transmitted from one location to another. In some cases, although image transmission to a remote location is not necessary, one does need to store the images for future retrieval and analysis.

The function of an image transmission system is to convey to an observer a "best" reproduction of a distant scene or document, subject to limitations which may be imposed by the cost and/or technical characteristics of the channel and the terminal equipment. In a broad sense, the field of image processing deals with the manipulation of data which are inherently two-dimensional in nature. The techniques of image processing find applications in many areas, notably: image enhancement, pictorial pattern recognition, smoothing, image understanding and the efficient coding of pictures for transmission or storage.

Mathematical models are becoming increasingly important because of their role in the development of useful algorithms for image processing. Virtually all applications of image processing utilize some sort of mathematical models. The common questions underlying these areas are:

- How do we describe or characterize images ?
- What mathematical operations do we want to use on the images ?

- How do we implement (in hardware) these mathematical operations ?
- How do we evaluate image quality ?

In this chapter a brief discussion about problems in image processing techniques, mathematical models and applications for image transmission will be given. The details of formulations of the mathematical models are beyond the subject of this study. Some detailed studies have been performed on this matter by A.K.Jain [1],[9],[11], T.S.Huang [4], W.F.Schreiber [2],[4], O.J.Tretiak [4] and W.K.Pratt [10].

In Sec.1.2, typical problems in image processing and desirable properties of image processing algorithms will be given. In Sec.1.3 the idea of two-dimensional signal processing will be summarized. In Sec.1.4 linear-transform processing of images will be examined. In section 1.5 image representation by one-dimensional stochastic models as autoregressive representations will be considered briefly. Then, linear prediction models in two dimensions will be examined. Section 1.7 will deal with image quality and in the last section applications in image processing will be summarized.

1.2 TYPICAL PROBLEMS IN IMAGE PROCESSING

As mentioned above, all applications of image processing utilize some sort of mathematical model. The continuing advances in high-speed digital processors, digital memories and very large scale integration have led to successful algorithms for many difficult problems. Table (1.1) gives a description of some of the typical problems in image processing and their associated modeling requirements. A typical algorithm requires a model such as bandwidth, power spectrum, etc., of the data (input to the algorithm).

While most of the problems listed in Table (1.1) also occur in one-dimensional signal processing, special care is needed in the development of two- and higher dimensional algorithms. The major difference besides the higher dimensionality is that of causality. A large number of one-dimensional signal processing methods are based on the fact that the observed data is the output of a causal system. For two-dimensional images the data coordinates are spatial and any causality associated with an image is purely due to its scanning or acquisition technique. Therefore it is not surprising that a large number of image processing algorithms for edge extraction, enhancement, restoration, data compression, etc., are noncausal.

The computational efficiency of algorithms is often measured by their memory and operation count requirements. The most efficient algorithms would be such that the required number of operations per pixel would be independent of the size of the image. Unfortunately, a large number of algorithms require an operation count which is proportional to $\log N$ or higher for $N * N$ images. Table (1.2) lists some of the desirable properties of two-dimensional models and algorithms which tend to minimize their computational complexity.

1.3 TWO DIMENSIONAL SIGNAL PROCESSING

Let us consider a picture as a two-dimensional signal $A(x,y)$ of the two spatial coordinates x and y . The extension of signal processing theory from a one-dimensional time-varying signal is straightforward. Sharp changes in brightness which require a high resolution optical system for accurate reproduction give rise to high spatial frequencies; thus, the spatial frequency spectrum can play the same role for pictures as the temporal

frequency spectrum does for time-varying signals. One can speak of the two-dimensional Fourier Transform of $A(x,y)$ which represents its spatial frequency spectrum

$$A_F(W_x, W_y) = \iint_{-\infty}^{\infty} A(x,y) \exp[-j(W_x x + W_y y)] dx dy \quad (1.1)$$

Where $A_F(W_x, W_y)$ represents the Fourier Transform of $A(x,y)$.

The impulse response in two dimensions, sometimes called the point spread function in optics, is also very useful. In a time-domain filter, let $H(x-\alpha, y-\beta)$, the output is some weighted average of the past history of the input where the weighing function is the impulse response. The output of a spatial filter, let $A_o(x,y)$, is some weighted average of the surrounding points in the input:

$$A_o(x,y) = \iint_{-\infty}^{\infty} A(\alpha,\beta) H(x-\alpha, y-\beta) d\alpha d\beta \quad (1.2)$$

Just as a time-varying signal can be sampled, $A(x,y)$ can be sampled by a two-dimensional array in which the spacing between the sample points is small enough so that the picture may be reproduced without aliasing.

Clearly, the maximum allowable space between samples depends on how rapidly the brightness changes, or in other words, on the spatial bandwidth.

The sampling theorem [7], [8] holds in two or more dimensions, and is useful to indicate that a bandlimited picture may be represented by the brightness at a finite number of points. This allows the digital computer to be used for the simulation of various image processing systems by discrete operations on the value of brightness at the sample points.

1.4 LINEAR TRANSFORM PROCESSING OF IMAGES

A classical way of analyzing a function is by its series expansion in terms of a set of complete orthonormal functions. The advantage of this method is the capability to obtain a relatively uncorrelated new set of variates by transforming the correlated ones. This new set of variates will have a varying degree of significance in contributing to both the information content and the subjective quality of the resulting picture. Then one can disregard the less significant of these variables without affecting the statistical information content of the picture or causing a severe degradation in the subjective quality of the resultant picture.

In the context of image processing a general orthogonal series expansion for an $N * N$ image $\{u_{xy}\}$ is a pair of UNITARY TRANSFORMATIONS of the form

$$u_{xy} = \sum_{k=1}^N \sum_{l=1}^N v_{kl} \alpha^*(x,y;k,l) \quad (1.3)$$

$$v_{kl} = \sum_{x=1}^N \sum_{y=1}^N u_{xy} \alpha(x,y;k,l) \quad (1.4)$$

where $\{\alpha(x,y;k,l)\}$ usually called the IMAGE TRANSFORM, is a set of complete orthonormal basis functions satisfying the properties

$$\sum_{k=1}^N \sum_{l=1}^N \alpha(x,y;k,l) \alpha^*(x',y';k,l) = \delta_{xx'} \delta_{yy'} \quad (\text{orthonormality}) \quad (1.5)$$

$$\sum_{x=1}^N \sum_{y=1}^N \alpha(x,y;k,l) \alpha^*(x,y;k',l') = \delta_{kk'} \delta_{ll'} \quad (\text{completeness}) \quad (1.6)$$

The elements v_{kl} are called the TRANSFORM COEFFICIENTS and $\{u_{xy}\}$ the

	PROBLEM	DESCRIPTION	MODELS
1	Smoothing	Given noisy image data filter it to smooth out the noise variation	noise and image power spectra
2	Enhancement	Bring out or enhance certain features of the image, eg, edge enhancement, contrast stretching etc	Features
3	Restoration and filtering	Restore an image known (or unknown) degradation as close to its original form as possible, eg, image deblurring, image reconstruction, etc	Degradations criterion of "closeness"
4	Data Compression	Minimize the number of bits required to store/transmit an image for a given level of distortion	Distortion criterion. Image as an information source
5	Feature Extraction	Extract certain features from an image eg, edges	Features, detection criterion
6	Detection and identification	Detect and identify the presence of an object from a scene, eg, matched filter, pattern recognition, image segmentation, etc	Detection criterion object and scene
7	Interpolation and Extrapolation	Given image data at certain pnts in a region estimate the value of all other pnts inside this region(interpol.) and also at pnts outside this region.(extrapol.)	Estimate criterion and degree of smoothness of the data
8	Spectral estimation	Given image data in a region, estimate its power spectrum	Criterion of estimation, apriory model for data
9	Spectral factorization	Given the mag. of the frequency response of a filter-design a realizable filter, eg, stable causal filter	Criterion of realizability
10	Synthesis	Given a description or some features of an image, design a system which reproduces a replica of that image, eg, texture synthesis	Features, criterion of reproduction

Table 1.1 Typical problems in Image processing by A.K.Jain [1]

PROPERTY	DESCRIPTION
Linearity	Linear operation on data
Separability	Independent row and column operations
Shift invariance	Operations leading to Toeplitz and circulant matrix manipulations
Marcovian or finite memory	Only local and/or sparse operations required in each pixel, eg, FIR filters

Table 1.2 Desirable properties of image processing algorithms by A.K.Jain [1]

transformed image. It is readily seen from Eq (1.3) and (1.4) that the general unitary transformation would require $O(N^4)$ operations, one operation being a multiplication and a summation. For typical size ($N = 256$) images this means over a billion operations would be needed to compute the transform coefficients. To reduce dimensionality, the unitary transforms in Eq (1.3) and (1.4) are restricted to the PRODUCT SEPARABLE CLASS, satisfying the condition:

$$\alpha(x,y;k,l) = a_{xk} b_{yl} \quad (1.7)$$

where $A = \{\beta_{xy}\}$ and $B = \{\gamma_{xy}\}$ are UNITARY MATRICES. (i.e., $A^{-1} = A^{*T}$). Often in image processing one chooses $A = B$ so that Eq (1.3) and (1.4) yield

$$V = A U A^T \quad (1.8)$$

$$U = A^{*T} V A^* \quad (1.9)$$

Now the transformations require column operations followed by row operations on the result, reducing the computations to $O(N^3)$ operations. Even this reduction is insufficient and the choice of image transforms is further restricted to FAST TRANSFORMS. Typically, these transform matrices have structural properties which lead to Fast Fourier Transform (FFT) type algorithms. Hence a transformation of the type $y = Ax$, for an $N * 1$ vector x could be performed in $O(N \log N)$ operations, so that for images the operation count is $O(N^2 \log N)$ or $\log N$ per pel. Examples of fast unitary transforms are the discrete Fourier (DFT), cosine (DCT), sine (DST), Walsh-Hadamard (WHT) transforms [9], [10], [12], [14]. Other fast transforms include the Haar, Slant [9], [10], [14], and a family of sinusoidal transforms [15]. A useful property of all unitary transforms is their energy conservation property

$$\sum_{x=1}^N \sum_{y=1}^N |u_{xy}|^2 = \sum_{k=1}^N \sum_{l=1}^N |v_{k,l}|^2 \quad (1.10)$$

known as PARSEVAL's relation. This follows from the fact that a unitary transform is simply a rotation of the image viewed as a vector in an N^2 dimensional vector space so that the length of the vector remains unchanged.

KARHUNEN-LOEVE TRANSFORM

Of particular significance among unitary transforms is the so-called Karhunen-Loeve transform (KLT) for random fields. Generally, one could consider the autocorrelation function itself. It is the complete orthonormal set of images $\phi(x,y;k,l)$ determined from the Eigenvalue equation

$$\sum_{m=1}^N \sum_{n=1}^N r(k,l;m,n) \phi(x,y;m,n) = \lambda_{xy} \phi(x,y;k,l) \quad (1.11)$$

where $r(\cdot)$ is the image covariance function. For separable covariance functions, the KLT is also separable. Two significant properties which make the KLT very desirable are as follows [1], [9], [16]:

- 1 - It completely decorrelates the transform coefficients
- 2 - Compared to all other unitary transforms, the KLT packs the maximum expected energy in a given number of samples. It has been shown in [13] and [15] that a suitable fast sinusoidal transform such as DCT or DST could be found as a good approximation to the KLT. The sinusoidal transforms have equivalent performance to KLT as $N \rightarrow \infty$. In image processing N can be quite large, and one often processes smaller blocks (typically $16 * 16$) of an image at a time. The performance differences between the various transforms

are significant enough to warrant the use of the KLT or a reasonable substitute of it. It has been shown by A.K.Jain [17] that the separable DCT is a good substitute for the nonseparable KLT of other stationary random fields also. Image transforms have been applied extensively in data compression, noise smoothing and restoration of images.

1.5 IMAGE REPRESENTATION BY ONE-DIMENSIONAL STOCHASTIC PROCESSES

Often it is desired to design image processing algorithms for an ensemble of images. For practical reasons this ensemble is generally characterized by the covariance and mean functions. These functions could be specified by a mathematical formula or simply as arrays of numerical values.

A simple way to characterize an image is to consider it as a collection of one-dimensional signals, e.g., as an output of a raster scanner, or as a sequence of rows (or columns) ignoring the interrow (or column) dependencies. This approach will be explained in chapter 2 for simulation of DPCM and PCM systems. (See CH:2, section 2.5) For such cases, one-dimensional representations of stochastic processes are useful. One-dimensional stochastic models have been applied in line by line processing of images for DPCM coding, hybrid coding, recursive filtering and restoration etc.[18] - [22]. Some examples for image representation by one-dimensional stochastic processes will given below.

1.5.1 AUTOREGRESSIVE REPRESENTATION

If $\{u_k\}$ is a zero-mean stationary gaussian random sequence then a causal representation of the type

$$u_k = \sum_{n=1}^P a_n u_{k-n} + \varepsilon_k \quad (1.12)$$

is called a (one-sided) AUTOREGRESSIVE (AR) representation. The sequence $\{\varepsilon_k\}$ is a zero-mean white-noise random sequence process independent of the past outputs. AR models have the following important properties.

1) The quantity

$$\begin{aligned} \bar{u}_k &= \sum_{n=1}^P a_n u_{k-n} \\ &= \text{mean}[u_k | u_n, \forall n \leq k-1] \end{aligned} \quad (1.13)$$

is the best mean-square predictor of u_k based on all of its past and depends only on the past p samples. Thus, Eq (1.12) becomes

$$u_k = \bar{u}_k + \varepsilon_k \quad (1.14)$$

which says the sample at k is the sum of its minimum variance causal prediction estimate plus the prediction error. The sequence $\{u_k\}$ defined by Eq (1.12) is called a p th order Markov process.

2) The AR process is stationary and causally stable if and only if the roots of the polynomial

$$A_P(z) = 1 - \sum_{n=1}^P a_n z^{-n} \quad (1.15)$$

lie inside the unit circle. Applications of autoregressive representations will be given in chapter 2. The algebra of this algorithm is not the subject of this study. The details of formulations are studied in [1], [18] - [22].

1.5.2 STATE VARIABLE MODELS

State variable models have been used to represent two-dimensional images considered as the output of a raster scanner. The scanner output is modeled as a one-dimensional random process and is characterized by a set of state variable equations of the form,

$$\begin{aligned} \frac{dx(t)}{dt} &= A(t)x(t) + B(t)\mathcal{E}(t) \\ y(t) &= C(t)x(t) \end{aligned} \quad (1.16)$$

where $y(t)$ is the scanner output at time t , and $x(t)$ is an $n \times 1$ vector, $\mathcal{E}(t)$ is a $p \times 1$ zero-mean white-noise vector such that

$$\text{mean}[\mathcal{E}(t) \mathcal{E}^T(t')] = K\delta(t-t') \quad (1.17)$$

A, B, C, K are appropriate matrices which are determined such that $y(t)$ satisfies (approximately, if not exactly) the statistics of the scanner output.

The first attempt to model images by state variable techniques was made by Nahi and Asseffi [20]. Although their final model has limitations because of several approximations, their modeling procedure does expose several

difficulties in representing two-dimensional random field by one-dimensional models.

1.5.3 NONCAUSAL MODELS [21], [23]

In section 1.5.1 we saw that a causal AR representation is of the type

$$u_k = \bar{u}_k + \varepsilon_k \quad ; \quad \text{mean}[\varepsilon_k] = 0 \quad (1.18)$$

where \bar{u}_k is the best linear mean square predictor of u_k based on the past values $\{u_l, 1 < l < k\}$ and $\{\varepsilon_k\}$ is a white-noise sequence. Thus, \bar{u}_k is a minimum variance causal predictor of u_k . In an analogous fashion, we can define a minimum variance noncausal predictor \bar{u}_k which depends on the past as well as the future values of u_k . Let $\{u_k\}$ be any zero-mean gaussian random sequence and let \bar{u}_k denote the best linear mean-square estimate of u_k based on all $\{u_l, 1 \neq k\}$. Writing

$$\bar{u}_k \triangleq \sum_{l \neq k} a_{k,l} u_l \quad (1.19)$$

we determine coefficients $a_{k,l}$ by minimizing the mean-square error $E[(u_k - \bar{u}_k)^2]$

1.6 LINEAR PREDICTION MODELS IN TWO DIMENSIONS

One important property of many one-dimensional systems is that of causality. For two-dimensional images, causality is not inherent in the data. Moreover, the data could be such that a causal realization by a finite-order linear system is not possible. This is because it is generally not possible to factorize a two-dimensional polynomial as a product of lower order

polynomials. In general, one can think of causal, semicausal and noncausal representations for two-dimensional images. These representations are the discrete equivalent of the classical categories, initial-value (or hyperbolic), initial-boundary value (or parabolic) and boundary value (or elliptic) representations of two-dimensional linear systems characterized by partial differential equations.

Linear prediction models in two dimensions are useful in image data transmission and storage via DPCM coding, and hybrid coding, design of recursive, semirecursive and nonrecursive filters for image estimation, restoration and filtering and in image analysis. For these purposes, three main prediction schemes have been developed which are briefly outlined below.

1.6.1 CAUSAL PREDICTION

Let $\{u_{x,y}\}$ be an arbitrary zero-mean gaussian random field and let $\hat{u}_{x,y}$ denote a prediction estimate of the random variable $u_{x,y}$. Some examples for linear prediction models will be given below. Suppose the samples of the random field $\{u_{x,y}\}$ are arranged in any desired, one-dimensional ordered sequence $\{u(k)\}$. Then $\hat{u}_{x,y}$ is defined as a causal prediction of $u_{x,y}$ if it depends only on the elements that occur before the element $u_{x,y}$. A common example occurs when an image raster scanned, say column by column, and $\hat{u}_{x,y}$ is a linear estimate based on all the elements scanned before arriving at (x,y) , i.e.,

$$\hat{u}_{x,y} = \sum_{m,n \in \delta} a(x,y;m,n)u_{m,n} \quad (1.20)$$

where $\delta = \{m,n : n < y \text{ for all } m\} \cup \{m,n : n = y, m < x\}$

1.6.2 SEMICAUSAL PREDICTION

If the estimate $\bar{u}_{x,y}$ is causal in one of the coordinates and noncausal in the other, it is called a semicausal predictor. For example, a linear semicausal predictor which is causal in "y" and noncausal in "x" would be of the form

$$\bar{u}_{x,y} = \sum_{m,n \in \delta} a(x,y;m,n)u_{m,n} \quad (1.21)$$

where $\delta = \{m,n : n < y \text{ for all } m\} \cup \{m,n : n = y, \text{ for all } m \neq x\}$

1.6.3 NONCAUSAL PREDICTION

The quantity $\bar{u}_{x,y}$ is defined as a causal prediction of $u_{x,y}$ if it can be written as a linear combination of possibly all the variables in the random field, except $u_{x,y}$ itself. For example, a linear noncausal predictor would be of the type

$$\bar{u}_{x,y} = \sum_{m,n \in \delta} a(x,y;m,n)u_{m,n}$$

where $\delta = \{m,n; (m,n) \neq (x,y)\}$ (1.22)

As mentioned at the beginning of this chapter, these mathematical models are becoming increasingly important because of their role in the development of useful algorithm for image processing. Virtually all applications of image processing utilize some sort of mathematical models. In section 1.8, we consider applications of these algorithms in several image processing problems, but before that, an important concept in image processing will be discussed in the following section.

1.7 IMAGE QUALITY CRITERIA

The effects of various parameters on picture quality have been discussed by Schreiber [2]. In most applications, picture quality is defined in subjective terms, and can only be measured in terms of observer response. There is no good reason to suppose that subjective quality is a one-dimensional quantity, however if it is multi-dimensional, it cannot be ranked. In practice, it is treated as a scalar, and is defined in terms of the protocol used to measure it. It is usually specified in terms of a four-to-six scale running from "excellent" to "unacceptable". A more attractive scale is graded in just noticeable differences in image quality, but in this case subjective measurement of relevant system parameters is quite a difficult and tedious procedure. Measurement of subjective image quality is very difficult; one can see this by reflecting on the fact that a 30% increase in bandwidth produces a just noticeable difference increment in appearance.

The practical problems in the measurement of subjective quality and the desire to design systems analytically brought forth several objective measures of subjective image quality. The most popular measures proposed to date are the mean-square error and its variants, such as the weighted mean-square error. These measures have the distinct advantage that they are mathematically tractable. They also appear to agree reasonably well with subjective evaluation in many cases.

1.7.1 MEAN-SQUARE ERROR CRITERIA

Let $u_{x,y}$ be the input image and $g_{x,y}$ the output image where (x,y) are the spatial coordinates, and u and g are brightness. We define the error as:

$$\epsilon_{x,y} = u_{x,y} - g_{x,y} \quad (1.23)$$

and denote its Fourier transform by $E_{u,v}$ where (u,v) are spatial frequencies.

Then the mean-square error is

$$D_1(u,g) = \iint_{-\infty}^{\infty} dx dy \epsilon^2(x,y) = \iint_{-\infty}^{\infty} du dv |E(u,v)|^2 \quad (1.24)$$

and the weighted mean-square error is

$$D_2(u,g) = \iint_{-\infty}^{\infty} du dv W(u,v) |E(u,v)|^2 \quad (1.25)$$

where $W(u,v)$ is called the weighting function. The weighting function reflects the sensitivity of the eye to various spatial frequency components in the image.

The mean-square error criteria have at least two disadvantages. First, the subjective quality of a degraded image $g_{x,y}$ depends not only on the error $\epsilon_{x,y}$ but also on the original image $u_{x,y}$; hence context dependent. Second, some image degradation are geometrical in nature - for example, block quantization using Hadamard transform (Ch:2, section 2.1.3), sometimes yields pictures, containing "staircases" along the edges. The mean-square error criteria do not seem appropriate for geometrical distortions. A more

satisfactory criterion should be based on some kind of edge error.

1.7.2 A DISTORTION MEASURE

In order to make up at least partially for the two defects of MSE criterion, in [4], the distortion measure is proposed

$$D(u,g) = A D_a(u,g) + B D_b(u,g) \quad (1.26)$$

here A and B are positive constants, D_a is a weighted mean-square error, modified to take care of the dependence on the original image and D_b is a measure of edge error. One possible choice for D_a is

$$D_a(u,g) = \iint_{-\infty}^{\infty} dudv |E(u,v)|^2 W_1(u,v)W_2(u,v) \quad (1.27)$$

where W_1 reflects the eye sensitivity, and W_2 reflects the dependence on the original image (and is a function of u). Much experimentations need to be done to determine suitable forms for D_b and W_2 .

1.8 IMAGE PROCESSING PROBLEMS AND APPLICATIONS

The models which were discussed have applications in most of the image processing problems listed in Table 1. As mentioned before the choice of a particular model depends not only on the accuracy of the model but also on the associated algorithm architecture.

1.8.1 SMOOTHING

Smoothing is a method which is used to suppress the noise that may be present in the picture. The basic difficulty with the smoothing techniques is that, if applied indiscriminately, tend to blur the picture, which is usually objectionable. In particular, one usually wants to avoid blurring sharp edges or lines that occur in the picture. The common smoothing problem is to find the best linear mean-square estimate of the image $u_{x,y}$ given the noisy observations

$$g_{x,y} = u_{x,y} + \epsilon_{x,y} \quad (1.28)$$

where $\{\epsilon_{x,y}\}$ is a white-noise field independent of $\{u_{x,y}\}$. Causal models have been used by several authors [22],[24]-[27] to develop recursive filter implementations. Semicausal models have been considered in [22] and [28] to develop semirecursive or line to line recursive algorithms. Noncausal models have been shown to yield fast transform based nonrecursive algorithms [22] and [29] and have also been found useful in developing moving average FIR filters.

1.8.2 IMAGE ENHANCEMENT

Image enhancement processes consist of a collection of techniques that seek to improve the visual appearance of an image, or to convert the image to a form better suited to human or machine analysis. In an image enhancement system there is no conscious effort to improve the fidelity of a reproduced image with regard to some ideal form of the image, as is done in image

restoration. Actually, there is some evidence to indicate that often a distorted image, for example, an image with edge overshoot, is actually more subjectively pleasing than a perfectly reproduced original.

An image enhancement system might emphasize the edge outline of an image by high-frequency filtering. This edge enhanced image would then serve as an input to a system that would trace the outline of the edges, and perhaps make measurements of the shape and size of the outline.

The prediction operator denoted by $A(Z_1, Z_2)$ generally performs some sort of differentiation. When applied to a real world image, the prediction error generally contains a nonstationary component which generally represents high spatial frequencies e.g. edges. Thus the operator

$$C(Z_1, Z_2) = 1 + \lambda A(Z_1, Z_2) \quad (1.29)$$

would add to the image a quantity proportional to high spatial frequencies (or gradients).

1.8.3 IMAGE RESTORATION

Image models have also been used in restoration of images blurred due to motion, atmospheric turbulence etc. Image restoration may be viewed as an estimation process in which operations are performed on an observed or measured field to estimate the ideal image field that would be observed if no image degradation were present in an imaging system.

There are two basic approaches to the modelling of image degradation effects: a priori modelling and a posteriori modelling. In the former case, measurements are made on the physical imaging system, digitizer and display to determine their response for an arbitrary image field. In some instances, it will be possible to model the system response deterministically, while in other situations it will only be possible to determine the system response in a stochastic sense. The posteriori modelling approach is to develop the model for the image degradations based on measurements of a particular image to be restored. Basically, these two approaches differ only in the manner in which information is gathered to describe the character of the image degradation.

1.8.4 IMAGE DATA COMPRESSION - PICTURE CODING

The trend in image transmission and storage is to use digital instead of analog techniques. This is because of the many inherent advantages of digital communication systems in the case of transmission and the flexibility of digital computers in the case of storage. An encoding scheme widely used for the transmission of digital signals is pulse code modulation

(PCM). Such an encoding scheme generally involves the sampling of the analog input signal at a uniform rate and encoding the samples in a binary code. (see Ch:2) PCM techniques require a high data rate for the transmission of images. Therefore, many digital schemes have been explored in order to reduce the capacity requirements of the digital image communication systems.

In picture compression one needs to represent a picture by uncorrelated data, and this can be achieved by using some reversible linear transformations. The data then must be ranked according to the degree of significance of their contribution to both the information content and the subjective quality of the picture. Once such a ranking is achieved then those elements of the data that are unimportant from the point of view of the grey scale and the spatial resolution capability of the receiver can be neglected (see section 1.4).

Transform coding techniques use FFT, KLT or other orthogonal transform models. If we do not limit ourselves to linear orthogonal transformations, there are other techniques which achieve the same result. One such technique which has the advantage of easy implementation is predictive coding (The models have been discussed in 1.6) Differential Pulse Code Modulation or DPCM, or Adaptive DPCM are used most frequently (see Ch:2). A DPCM encoder uses a linear prediction to generate a differential signal, then quantizes the signal with a quantizer that is designed for the probability density function of this signal. Both DPCM and transform coding techniques have been used with some success in coding pictorial data. A study of both these systems has indicated that each technique has some attractive characteristics and some limitations. The transform coding systems achieve superior coding performance at lower bit rates, they distribute the coding degradation in a manner less objectionable to a human viewer and show less

sensitivity to data statistics. DPCM systems on the other hand, when designed to take advantage of spatial correlations of the data, achieve a better coding performance at a higher bit rate. The equipment complexity and the delay due to the coding operations are minimal. Perhaps the most desirable characteristic of this system is the ease of design and the speed of operation.

A hybrid coding system that combines the attractive features of both transform coding and DPCM systems has also been used. This system exploits the correlation of the data in the horizontal direction by taking a one-dimensional transform of each of line of the picture, then operating on each column of the transformed data using a one-element predictor DPCM system.

For pictures in which the number of possible grey levels is small and which are composed of only a few regions each having a constant grey level, an inefficient method of compression is contour coding [33]. This involves tracing of the contours or boundaries between the constant grey level regions and sending only that informations to the receiver which would enable it to reproduce these contours.

Another method of compression for pictures composed of few regions each having constant or slowly varying grey level is run-length coding [30], [31], [32]. Here raster scanning of the picture followed by quantization will give rise to a relatively small number of "runs" of constant grey level, and the picture can be encoded by specifying lengths, or equivalently the positions, of these runs.

Another method of picture compression separates a picture into "highs" and "lows" [33]. The "lows" picture is obtained by low-pass spatial filtering of the picture. This results in essentially an out-of-focus picture with no sharp edges, i.e., a blurred picture. By the two dimensional sampling theorem, this "lows" picture can be represented by much fewer samples than would have been needed for the original picture. The "highs" signal is obtained by taking either the gradient or the Laplacian of the picture and consists of essentially the edges in the picture. A two-dimensional high-frequency picture, also called the "synthetic highs", can be synthesized from this edge information. This high-frequency information, when combined with the "lows" picture gives back essentially the original picture. The "highs" picture may be efficiently transmitted by contour coding edge information [33].

Roberts [34] has suggested a pseudo-random noise modulation technique for picture compression. If a continuous picture is sampled at an array of points, the samples usually need to be quantized to between 16 and 256 grey levels depending on the requirements of the user of the picture. If an attempt is made to reduce the number of bits by making the quantization too coarse, the result is the appearance of artificial discontinuities (false contours) in the picture. These discontinuities are a result of the quantization noise, which is correlated with the picture samples. If uncorrelated random noise of the same rms value as the quantization noise is added to the original continuous picture, these discontinuities do not appear. Roberts used this observation in his pseudo-random noise modulation technique in which noise of uniform amplitude distribution, and peak-to-peak value equal to one quantum step, is added to the picture samples before quantization, and the identical noise is subtracted at the receiver. The

result looks like an unquantized output, in which random noise of the same rms value has replaced the quantization noise (Dithering). Usable pictures are produced at 2 bits per picture element, fair ones at 3 bits and good ones at 4 bits. Roberts' system also has excellent performance in the presence of channel noise.

Another method of picture compression, which is known as interframe coding, has been widely used for television signals. In a television signal a large fraction of picture elements correspond to the background material and do not change from one frame to the next. On the other hand a relatively small number of picture elements change from one frame to the next to convey the new information that is generated by a relative movement of camera with the object in each frame. From a statistical viewpoint, the similarity of pixels from one frame to the next corresponds to a high level of interframe correlation. Thus the statistical coding techniques exploiting the spatial correlation of the data which were considered for coding single frames of data, could be extended to take advantage of the frame-to-frame correlation, further reducing the bit rate required to transmit the data. The design of these systems is based on the fact that only a small percentage of picture points in a television signal change in successive frames [35], [36], [3], [37]

Picture compression is the most important application where image models have probably had the most significant impact. Causal prediction models (see section 1.6) have been used most widely in the design of interframe and interframe predictive or the so-called DPCM coders. Semicausal models have been employed in transform DPCM coding. Noncausal models give rise to transform coding algorithms which have been found to give high performance.

The prediction model parameters determine the coder design details such as quantizer design, bit allocation, optimum transform, etc. In chapter 2, the most common picture coding techniques, PCM, DPCM, ADPCM, will be given in more detail.

1.8.5 EDGE EXTRACTION, PATTERN RECOGNITION

In examining a picture, one is very often interested only in extracting from it a description of what it depicts; this is the problem of pictorial pattern recognition. The desired description may be merely a classification of the picture into one of a small set of prespecified classes, in this case it can often be accomplished by measuring various properties of the picture as a whole.

An image feature or an edge is a distinguishing primitive characteristic or attribute of an image field. Some features are natural in the sense that such features are defined by the visual appearance of an image, while other so-called artificial features result from specific manipulations or measurements of an image. Natural edges include the brightness of a region of pixels, edge outlines of objects, and grey scale textural regions. Image amplitude histograms and spatial frequency spectra are examples of artificial features. Many pictorial pattern recognition problems involve more than just the assignment of a picture to one of a set of prespecified classes; they require a description of the picture, where the number of possible descriptions is so large that it makes it impractical to regard each description as defining a class. Typically, a description refers to various subjects of the picture, "objects", and specifies properties of these subjects. To arrive at such a description, an automated pattern

recognition system must be capable of singling out the appropriate picture subjects, "segmentation". There is no universal method of segmenting a picture; many different types of subjects can be "objects", depending on the type of description that is required. Noncausal models are very often used for edge extraction and pattern recognition methods. In chapter 3, edge extraction (edge detection) techniques will be studied in detail.

1.9 CONCLUSION AND SUMMARY

This chapter has briefly presented several mathematical models which have been used and are of potential use in image processing. Typical problems in image processing have also been discussed and the solutions and applications of these problems have been summarized. Several mathematical models like series expansion, one-dimensional AR, state variables, the causal, semicausal, and noncausal prediction models and type of applications like coding techniques, image enhancement were included in the discussion. It is impossible to cover all the important aspects of image processing in this study. But, the reader is referred to the bibliography for more detailed information about the topics which have been briefly mentioned in this chapter. Several important aspects of image processing will be discussed in more detail in the succeeding chapters.

II. PICTURE CODING TECHNIQUES

The advantages of coding a signal digitally are well known and are widely discussed in the literature. Briefly, digital representation offers ruggedness, efficient signal generation, the possibility of combining transmission and switching functions, and the advantage of a uniform format for different types of signals. The price paid for these benefits is the need for increased bandwidth. Here some techniques of waveform coding will be discussed. The discussion will be confined to the encoding of image waveforms by means of a straightforward reconstruction of the waveform: specifically by means of the closely related discrete-time, discrete amplitude representation known as Pulse Code Modulation (PCM) and Differential Pulse Code Modulation (DPCM). These techniques are appropriate for the communication of any bandlimited time function.

Although the main subject of this work is the investigation of the various Image Transmission Techniques, this chapter will handle the coding techniques such as quantization, PCM or DPCM in a broad sense because of two reasons:

- 1- PCM or DPCM are used for digital speech communication as well as for digital image transmission. In fact, there are many more studies about speech processing than image processing. Then it is preferred to examine PCM and DPCM techniques in more general sense.
- 2- As mentioned earlier, the computer which is used for this study has limited memory and storage capacity and also has no ability to draw pictures

as differences of grey level intensities. Because of these limitations, the picture coding techniques mentioned above were tested on one-dimensional AR model sequences which are very suitable models for image coding techniques. (see chapter 1). It must be kept in mind that if a coding technique is applied on a real continuous image, the system in which this process is performed must have a two-dimensional filter, two-dimensional sampler, two-dimensional quantizer and a scanner. A scanner transforms a two-dimensional array of samples into a one-dimensional sequence.

In section 2.1 the concept of quantization is discussed in detail and different types of quantizers will be given. In section 2.2 the general theory of sequential or differential quantization is considered, then in section 2.3 a brief explanation of linear predictors is given. In the next section (2.4) Pulse Code Modulation (PCM), Differential PCM, Adaptive DPCM are discussed. In the last section the computer simulations of PCM and DPCM techniques are introduced and some examples of outputs are given. A package program which includes different types of quantizers, different types of waveforms and coding techniques is introduced.

2.1 QUANTIZATION

In order to transmit the sequence of samples over a digital communication channel or to store them in a digital memory or to use them as the input to a digital signal processing algorithm, the sample values must be quantized to a finite set of amplitudes so that they can be represented by a finite set of symbols.

Quantization begins with the availability of analog samples. Each sample may

in general take on any of a continuum of amplitude values ranging from $-\infty$ to $+\infty$. The quantizer replaces each of these sample values with an output value which is an approximation to the original amplitude. The key feature is that each output value is one of a finite set of real numbers. Hence a symbol from a finite alphabet can be used to represent and identify the particular output value that occurs. A distinct B-bit binary word can be associated with each output value if the set of output values contains no more than 2^B members. With this procedure a sequence of analog samples can be transformed into a sequence of binary words suitable for storage, transmission, or some other form of digital signal processing. A receiver having the table of output values (sometimes called "quanta" or "quantum levels") associated with the set of binary words, can then reconstruct an approximation to the original sequence of samples.

It is generally desirable to maintain the bit rate as low as possible while maintaining a required level of quality. For a given waveform bandwidth (such as speech or image intensity), the minimum sampling rate is fixed by the sampling theorem. Therefore, the only way to reduce the bit rate is to reduce the number of bits/sample. For this reason, here a variety of techniques for quantizing a signal will be discussed.

In general it is reasonable to assume that the samples $\{x(n)\}$ will fall in a finite range of amplitudes such that

$$|x(n)| \leq X_{\max} \quad (2.1)$$

for convenience, it may be desirable to assume that X_{\max} is infinite as for example when we assume a particular form for the probability density

function of the amplitudes of $x(n)$, such as the Gamma or the Laplace distribution. However, the assumption of a finite range of amplitudes is more realistic. The quantity X_{max} is a parameter of the quantizing system, which specifies the "loading factor" of the system, i.e., the amplitudes beyond which all samples are clipped. In order to avoid overloading and clipping a "dynamic range" for the quantizer must be defined. This is related to σ_x (variance of the input signal). It can be easily shown that only 0.35% of samples would fall outside the range if the dynamic range is

$$-4\sigma_x \leq x(n) < 4\sigma_x \quad (2.2)$$

where
$$\sigma_x = 1/N \sum_{i=1}^N x_i^2 \quad (2.3)$$

N =total number of samples

which is also known as the "peak to peak range". For the Laplacian density the peak to peak range of the quantizer will be $2(4\sigma_x)$.

The process of quantizing and coding is generally depicted as in Fig 2.1.

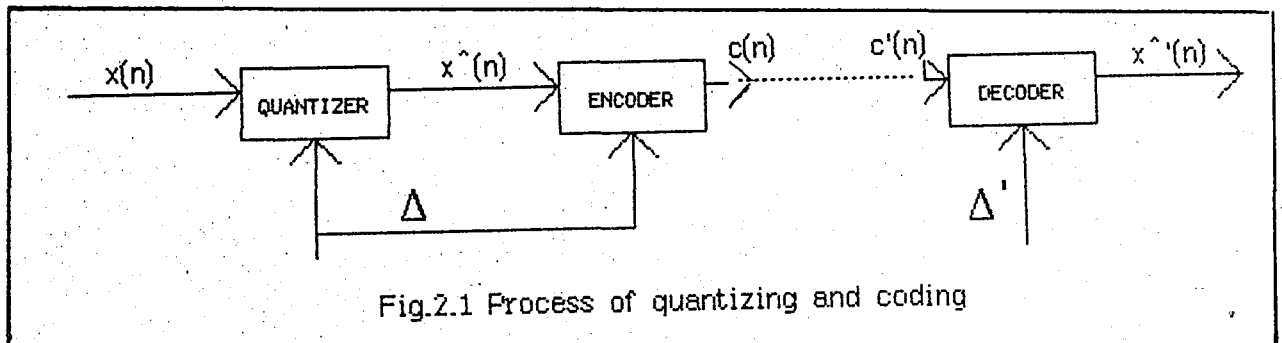


Fig.2.1 Process of quantizing and coding

where $\{x(n)\}$ represents input samples

$\{\hat{x}(n)\}$ represents quantized values of input samples

$\{c(n)\}$ represents code word

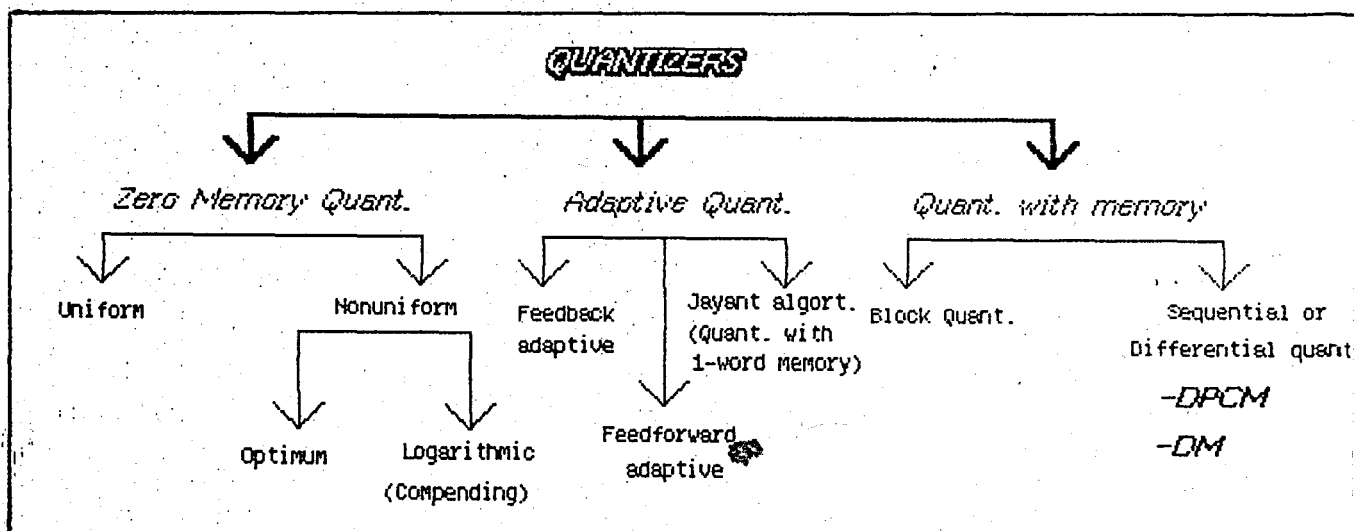
Δ represents quantization step size at the transmitter side;

and $\{x'(n)\}$ represents input samples

$\{c'(n)\}$ represents code word

Δ' represents quantization step size at the receiver side.

Quantizers can be classified as:



The simplest and most common form of quantizer is the zero-memory quantizer. In this case, the output value is determined by the quantizer only from one corresponding input sample, independent of the values, taken on by earlier (or later) analog samples, applied to the quantizer input.

The types of quantizers which are defined as zero memory quantizers in the above table can be transformed into adaptive quantizers by adding memory to

the configuration of the circuit. In the case of adaptive quantizers, the output values are determined by the quantizer from the values taken on by earlier (or later) analog samples applied to the quantizer input.

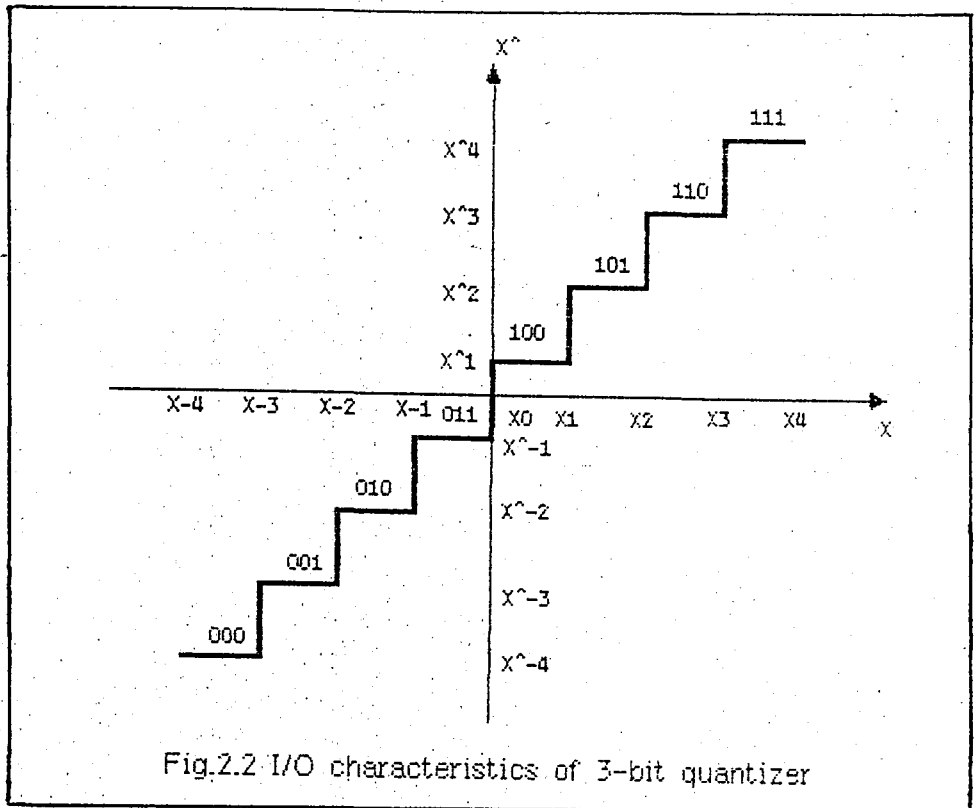
A more sophisticated quantizer is the one which looks at a group or "block" of input samples simultaneously and produces a block of output values, chosen from a finite set of possible output blocks, approximating the corresponding input samples. In general, for a given number of bits per sample representing the output values, a better quality approximation can be achieved by block quantization.

2.1.1 ZERO MEMORY QUANTIZATION

A zero memory N -point quantizer Q may be defined by specifying a set of $N+1$ decision levels x_0, x_1, \dots, x_N and a set of output points $\hat{x}_1, \hat{x}_2, \dots, \hat{x}_N$. When the value x of an input sample lies in the i^{th} quantizing interval, namely,

$$R_n = \{ x_{n-1} < x < x_n \} \quad (2.4)$$

the quantizer produces the output value \hat{x}_n . The input-output characteristic of a quantizer has a staircase form.



UNIFORM QUANTIZATION

The quantization levels and ranges are generally distributed uniformly. Thus

$$x_n - x_{n-1} = \Delta \quad (2.5)$$

$$\hat{x}_n - \hat{x}_{n-1} = \Delta \quad (2.6)$$

where Δ is the quantization stepsize. Two common uniform quantizer characteristics are shown below and the value of the quantization levels and stepsizes are shown in Table (2.1). (Midtread and Midrise uniform quantizer output and decision values). Values are calculated according to Eqn (2.9) where $X_{max} = 4 \sigma_x$.

N	2		4		8		16		32		64	
	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i
1	∞	4.000	3.000	2.000	0.500	1.000	0.250	0.500	0.125	0.250	0.063	0.125
2			∞	4.000	1.500	2.000	0.750	1.000	0.375	0.500	0.188	0.250
3					2.500	3.000	1.250	1.500	0.625	0.750	0.313	0.375
4					∞	4.000	1.750	2.000	0.875	1.000	0.438	0.500
5							2.250	2.500	1.125	1.250	0.563	0.625
6							2.750	3.000	1.375	1.500	0.688	0.750
7							2.250	3.500	1.625	1.750	0.813	0.875
8							∞	4.000	1.875	2.000	0.938	1.000
9									2.125	2.250	1.063	1.125
10									2.375	2.500	1.188	1.250
11									2.625	2.750	1.313	1.375
12									2.875	3.000	1.438	1.500
13									3.125	3.250	1.563	1.625
14									3.375	3.500	1.688	1.750
15									3.625	3.750	1.813	1.875
16									∞	4.000	1.938	2.000
17											2.063	2.125
18											2.188	2.250
19											2.313	2.375
20											2.438	2.500
21											2.563	2.625
22											2.688	2.750
23											2.813	2.875
24											2.938	3.000
25											3.063	3.125
26											3.188	3.250
27											3.313	3.375
28											3.438	3.500
29											3.563	3.625
30											3.688	3.750
31											3.813	3.875
32											∞	4.000

Table 2.1.a I/O values of uniform quantizers with mid-tread values ($m_x=0, \sigma_x^2=1$)

N	2		4		8		16		32		64	
	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i
1	∞	2.000	2.000	1.000	1.000	0.500	0.500	0.250	0.250	0.125	0.125	0.063
2			∞	3.000	2.000	1.500	1.000	0.750	0.500	0.375	0.250	0.188
3					3.000	2.500	1.500	1.250	0.750	0.625	0.375	0.313
4					∞	3.500	2.000	1.750	1.000	0.875	0.500	0.438
5							2.500	2.250	1.250	1.125	0.625	0.563
6							3.000	2.750	1.500	1.375	0.750	0.688
7							3.500	3.250	1.750	1.625	0.875	0.813
8							∞	3.750	2.000	1.875	1.000	0.938
9									2.250	2.125	1.125	1.063
10									2.500	2.375	1.250	1.188
11									2.750	2.625	1.375	1.313
12									3.000	2.875	1.500	1.438
13									3.250	3.125	1.625	1.563
14									3.500	3.375	1.750	1.688
15									3.750	3.625	1.875	1.813
16									∞	3.875	2.000	1.938
17											2.125	2.063
18											2.250	2.188
19											2.375	2.313
20											2.500	2.438
21											2.625	2.563
22											2.750	2.688
23											2.875	2.813
24											3.000	2.938
25											3.125	3.063
26											3.250	3.188
27											3.375	3.313
28											3.500	3.438
29											3.625	3.563
30											3.750	3.688
31											3.875	3.813
32											∞	3.938

Table 2.1.b I/O values of uniform quantizers with mid-riser values ($m_x=0, \sigma_x^2=1$)

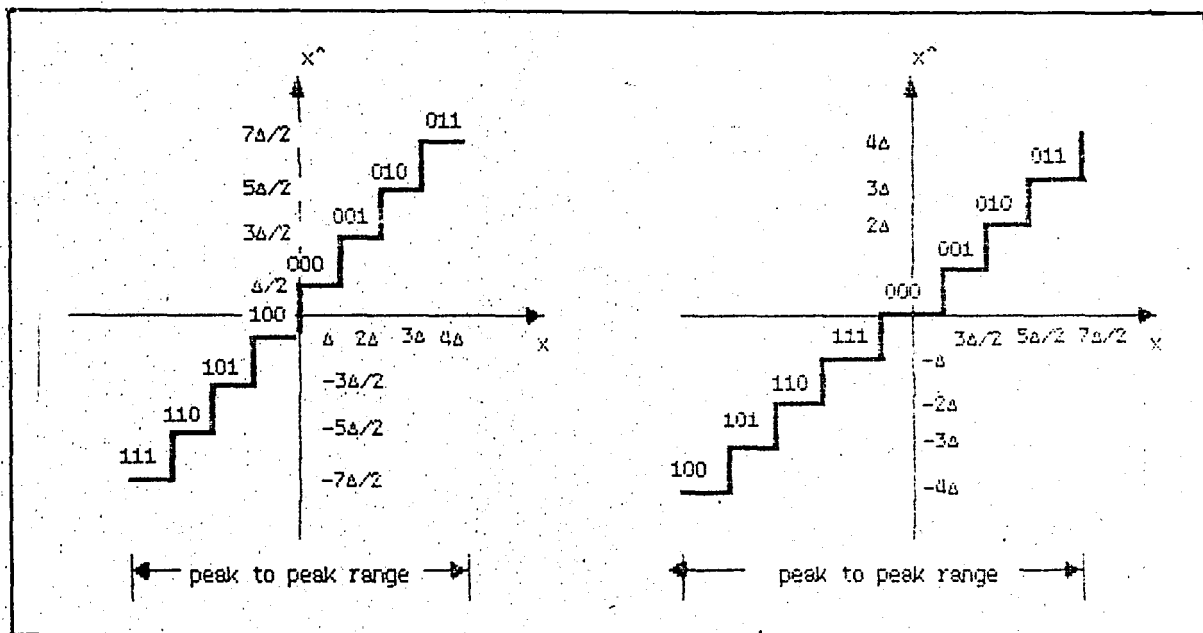


Fig 2.3 Two common uniform quantizer characteristics

a) Mid-riser b) Mid-tread

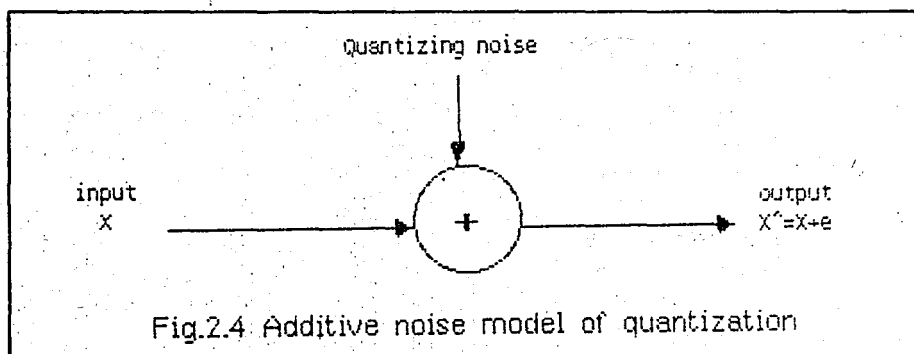


Fig.2.4 Additive noise model of quantization

Here the quantization error is $e(n) = x(n) - \hat{x}(n)$ for the n th sample.

If we interpret the binary code words as a sign-magnitude representation with the left-most bit being the sign-bit, then the quantized samples are

related to the code words by the relationship

$$\hat{x}(n) = \frac{\Delta}{2} \text{sign}(c(n)) + \Delta c(n) \quad (2.7)$$

where $\text{sign}(c(n))$ is equal to +1 if the first bit of $c(n)$ is 0 and -1 if the first bit of $c(n)$ is 1. Similarly, one can interpret the binary code words in Fig 2.3b as a 3-bit two's complement representation, in which case the quantized samples are related to the code words by the relationship

$$\hat{x}(n) = \Delta c(n) \quad (2.8)$$

This latter method of assignment of code words to quantization levels is most commonly used when the sequence of samples is to be processed by a signal processing algorithm which is implemented with two's complement arithmetic, since the code words can serve as a direct numerical presentation of the sample values.

For uniform quantizers there are only two parameters: The number of levels and the quantization stepsize Δ . The number of levels is generally to be of the form 2^B so as to make the most efficient use of B-bit binary code words. Together, Δ and B must be chosen so as to cover the range of input samples. If we assume that $|x(n)| \leq X_{\max}$, then we should get

$$2X_{\max} = \Delta 2^B \quad (2.9)$$

If Δ and B are chosen as in Eq.(2.9) then quantization error $e(n)$ will be restricted to the interval:

$$-\frac{\Delta}{2} \leq e(n) \leq \frac{\Delta}{2} \quad (2.10)$$

PERFORMANCE MEASURE = SNR

In order to compute the strength of the noise on images it is convenient to compute the signal-to quantization noise ratio defined as

$$\text{SNR} = \frac{\sigma_x^2}{\sigma_e^2} = \frac{\{E x^2(n)\}}{\{E e^2(n)\}} \cong \frac{\sum_{n=1}^N x^2(n)}{\sum_{n=1}^N e^2(n)} \quad (2.11)$$

in its simple form. Since the quantization error is modelled as a random variable, a measure of the performance of a quantizer must be based on a statistical average of some function of the error. Then we can define SNR which is related to the mean-square distortion measure D, defined as

$$D = \sigma_e^2 = \int_{-\infty}^{\infty} [Q(x)-x]^2 p(x) dx = \int_{-\infty}^{\infty} e^2 p(x) dx = \int_{-\infty}^{\infty} (\hat{x}-x)^2 p(x) dx \quad (2.12)$$

where $Q(x)$ is the output characteristic of the quantizers. This quantity can be used to measure the degradation introduced by the quantizer for a fixed input pdf $p(x)$. SNR is often defined as

$$\text{SNR} = 10 \log_{10} (\sigma_x^2 / D) \quad (2.13)$$

where σ_x^2 is the variance of the input samples. In most applications of quantization, the number of levels N is very large so that a sufficiently high SNR is obtained. A useful formula for mean-squared error can then be used. Eq.(2.12) can then be written in the form,

$$D = \sum_{n=1}^N \int_{x_{n-1}}^{x_n} (\hat{x}_n - x)^2 p(x) dx \quad (2.14)$$

by breaking up the region of integration into the separate intervals R_n and noting that $Q(x) = \hat{x}_n$ when x is in R_n . For large N each interval R_n can be made very small. Then it is reasonable to approximate the probability density $p(x)$ as being constant within the interval R_n . On setting $p(x) \cong p(\hat{x}_n)$ when x is in R_n and approximating $p(x) \cong 0$ for x in the overload regions (R_1 & R_N), the integral for each term of the sum in Eq.(2.14) is readily found and we get

$$D = 1/12 \sum_{n=2}^{N-1} p(\hat{x}_n) \Delta_n^3 \quad (2.15)$$

where $\Delta_n = x_n - x_{n-1}$, the length of interval R_n . This approximate formula is based on the assumption that, for large N , a sufficient number of quantizing levels are available for both the granularity and the overload noise to be very small.

The staircase quantizer characteristic of the uniform quantizer has equal width and equal height steps. The expression for mean-square error simplifies to

$$D = \Delta^2/12 \sum_{n=2}^{N-1} p(\hat{x}_n) \Delta \quad (2.16)$$

But

$$\sum p(\hat{x}_n) \Delta \cong \int p(s) ds = 1 \quad (2.17)$$

So that

$$D \cong \Delta^2/12 \quad (2.18)$$

Thus the mean-square distortion of a uniform quantizer grows as the square of the stepsize. This is perhaps the most often used result in quantization.

As mentioned above, the peak-to-peak range quantizer range is assumed to be $2X_{\max}$, then, for a B-bit quantizer, we get $\Delta = 2X_{\max} / 2^B$. Then

$$D = \sigma_e^2 \approx \frac{\Delta^2}{12} = \frac{X_{\max}^2}{(3)2^{2B}} \quad (2.19)$$

Substituting Eqn (2.19) into Eqn (2.11) gives

$$\text{SNR} = \frac{(3)2^{2B}}{\left[\frac{X_{\max}}{\sigma_x}\right]^2} \quad (2.20)$$

or expressing the signal-to quantizing noise in dB units

$$\text{SNR(dB)} = 10 \log_{10} \left[\frac{\sigma_x^2}{\sigma_e^2} \right] \quad (2.21)$$

$$\text{SNR(dB)} = 6B + 4.77 - 20 \log_{10} \left[\frac{X_{\max}}{\sigma_x} \right] \quad (2.22)$$

If we assume that the quantizer range is such that $X_{\max} = 4\sigma_x$, then eq.(2.22) becomes

$$\text{SNR(dB)} = 6B - 7.2 \quad (2.23)$$

Eqn(2.23), which states that each bit in the code word contributes 6dB to the signal-to-noise ratio.

In order to maintain a fidelity of representation with uniform quantization that is acceptable perceptually, it is necessary to use many more bits than might be implied by the previous analysis in which we have assumed that the signal is stationary. For example, whereas Eqn(2.23) suggests that $B = 7$ would provide about 36 dB SNR, which would most likely provide adequate quality in a communication system, it is generally accepted that about 11

bits are required to provide high quality representation of speech signals with a uniform quantizer.

An example to uniform mid-riser quantizer will be given in section 2.5. An AR model waveform, which is explained in section 1.5, is used as an input to the quantizer.

For all of the above reasons, it would be very desirable to have a quantizing system for which the SNR was independent of the signal level. That is, rather than the error being of constant variance independent of signal amplitude as for uniform quantization, it would be desirable to have a constant percentage error. This can be achieved by using a non-uniform distribution of quantization levels.

NON-UNIFORM QUANTIZERS

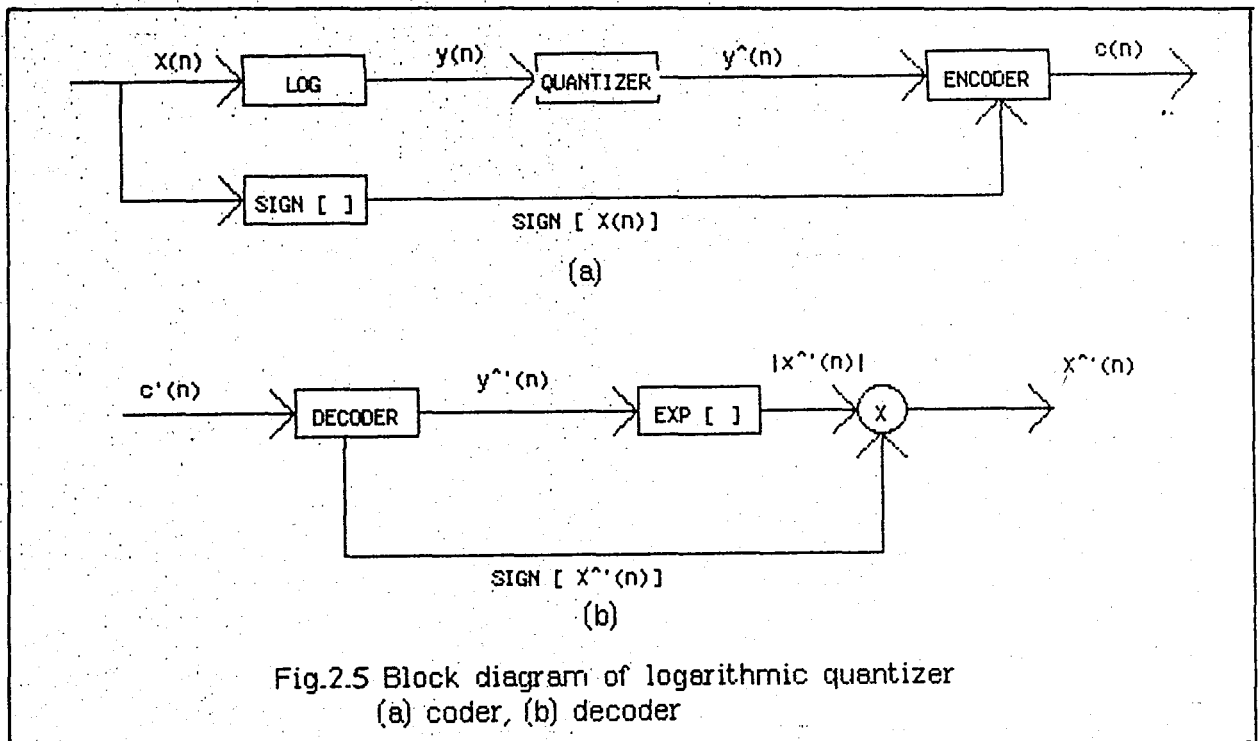
The use of a non-uniform quantizer is equivalent to taking compressed signals as an input to uniform quantizer and a subsequent expansion of the output.

Consider a non-uniform quantizer with the feature that the stepsize increases as we go away from the center of the (zero-mean) quantizer. The advantage of such a quantizer is that without increasing the total number of quantization levels (and hence the needed bit rate), one can allow large end steps in the quantizer to take care of possible excursions of the signal into the (relatively infrequent) large amplitude ranges. Equivalently, for a given quality of encoding, over a specified dynamic range of the signal, a nonlinear quantizer permits a reduction of the bit rate. For example, the

logarithmic quantizer of Smith [47] can typically be designed to provide waveform quantization with 128 levels or 7 bits per sample, while a uniform quantizer needs about 11 bits for a similar performance.

LOGARITMIC QUANTIZER

As mentioned above, it would be very desirable to have a quantizing system for which the SNR was independent of the signal level. That is, rather than the error being of constant variance independent of the signal amplitude as for uniform quantization, it would be desirable to have a constant percentage error. In order to have a constant percentage error, either the quantization levels must be logarithmically spaced or the logarithm of the input should be quantized rather than the input itself.



The procedure which is used in this study is the following: The logarithm of the input samples are taken. Sign of the original input values $x(n)$ are

saved.

$$y(n) = \ln |x(n)| \quad (2.24)$$

Then the $y(n)$'s are quantized by a uniform quantizer. The inverse transformation is

$$x(n) = \exp [y(n)] \text{ sign}[x(n)] \quad (2.25)$$

where $\text{sign}[x(n)]$ is + if $x(n)$ is positive, and - if $x(n)$ is negative. Now the quantized log magnitude is

$$\hat{y}(n) = Q [\log|x(n)|] \quad (2.26)$$

$$\hat{y}(n) = \log|x(n)| + e(n) \quad (2.27)$$

where we have assumed as before that $e(n)$ is independent of $\log|x(n)|$. The inverse of the quantized log magnitude is

$$\begin{aligned} \hat{x}(n) &= \exp[\hat{y}(n)] \text{sign}[x(n)] \\ \hat{x}(n) &= |x(n)| \text{sign}[x(n)] \exp[e(n)] \\ \hat{x}(n) &= x(n) \exp[e(n)] \end{aligned} \quad (2.28)$$

if $e(n)$ is small, we can approximate the above equation by

$$\hat{x}(n) \cong x(n)[1+e(n)] = x(n) + e(n)x(n) = x(n) + f(n) \quad (2.29)$$

where $f(n) = e(n)x(n)$. Thus, since $x(n)$ and $e(n)$ are assumed to be independent

$$\sigma_f^2 = \sigma_x^2 \cdot \sigma_e^2 \quad (2.30)$$

and

$$\text{SNR} = \frac{\sigma_x^2}{\sigma_f^2} = \frac{1}{\sigma_e^2} \quad (2.31)$$

That is, the SNR is independent of the signal variance. It depends only upon the stepsize. The simulation results of the logarithmic quantization will be given in section 2.5.

OPTIMUM QUANTIZATION

The logarithmic quantizer strives to achieve constant SNR over a wide range of signal variances. As we have just seen this is achieved at some sacrifice over the SNR performance that can be achieved if the quantizer stepsize is matched with the variance of the signal. In cases where the signal variance is known, it is possible to choose the quantizer levels so as to minimize the quantization error.

For applications where one particular probability density function is known to describe adequately the distribution of input samples to be quantized, it is natural to seek the best possible quantizer characteristic for that density. Two approaches have been taken to solve this problem: One uses the assumption that N is large and leads to explicit solutions; the other is valid for any N and leads to algorithmic procedures for finding the optimum decision levels and output points.

In 1960, Max [48] formulated the necessary conditions for optimality for a k th absolute mean error criterion. He examined the optimization of the stepsize for uniform quantization. Max also tabulated the optimum quantizer levels for the Gaussian distribution, for various values of N .

For the mean-square error criterion, with some fixed values of N , the necessary conditions for optimality on the values of x_1, x_2, \dots, x_{N-1} and $\hat{x}_1, \hat{x}_2, \dots, \hat{x}_N$ are found simply by setting the derivatives of D with respect to each of these parameters to zero. The resulting conditions are as follows.

1- Each output level of \hat{x}_n must be the centroid or center of mass of the interval R_n with respect to the input density $p(x)$. In other words, \hat{x}_n is the conditional mean value of the input random variables x given that x is in the region R_n .

2- Each decision level x_n must be half way between the two adjacent output points.

These conditions do not give the optimum values explicitly. Since the value of the output point \hat{x}_n for an interval R_n depends on the value of the decision levels x_{n-1} and x_n defining R_n , and the decision level x_n depends on the output level \hat{x}_n and \hat{x}_{n+1} . However, these conditions are used in Max's algorithm for computing iteratively a set of parameters that simultaneously satisfy both conditions. Using the Max algorithm, Paez and Glissen [49] tabulated the optimum quantizer parameters for the Laplacian and a particular form of the Gamma density. It can be shown that

$$\frac{d}{dx^2} [\log p(x)] < 0 \quad (2.32)$$

for all x , in other words, if $\log p(x)$ is concave, then only one quantizer exists which satisfies the Max conditions 1) and 2) and that quantizer is indeed optimal. It should be noted that the converse is not true, so that it

is possible to have a density $p(x)$ not satisfying Eqn (2.32) and yet a unique optimal quantizer may exist. Eqn (2.32) holds for the Gaussian density as well as for many other common densities. Hence, the tabulated quantizer parameters, given by Max for the Gaussian density, are in fact unique and optimal.

Table (2.2) shows optimum quantizer parameters for Laplacian, Gamma and Gauss densities. These numbers are derived assuming unit variance. If the variance of the input is σ_x^2 then the numbers in the table should be multiplied by σ_x .

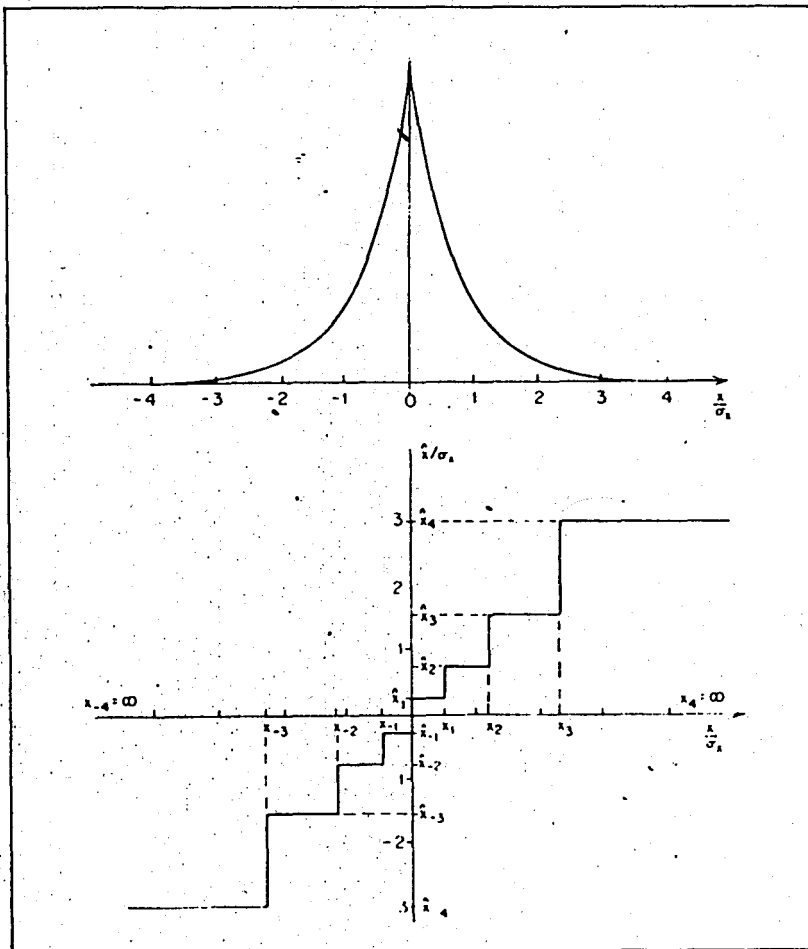


Fig 2.6 Density function and quantizer characteristic for Laplacian density function and 3-bit quantizer

N	2		4		8		16		32	
	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i
1	∞	0.798	0.962	0.453	0.501	0.245	0.258	0.128	0.132	0.066
2			∞	1.510	1.050	0.756	0.522	0.388	0.265	0.198
3					1.748	1.344	0.800	0.657	0.399	0.334
4					∞	2.152	1.099	0.942	0.536	0.467
5							1.437	1.256	0.676	0.605
6							1.844	1.618	0.621	0.742
7							2.4	2.069	0.972	0.892
8							∞	2.733	1.130	1.049
9									1.299	1.212
10									1.462	1.367
11									1.662	1.577
12									1.908	1.768
13									2.174	2.029
14									2.505	2.319
15									2.977	2.692
16									∞	3.263

Table 2.2a Optimum Quantizers with Gaussian Density

$$(m_x = 0, \sigma_x^2 = 1)$$

N	2		4		8		16		32	
i	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i
1	∞	0.707	1.102	0.395	0.504	0.222	0.266	0.126	0.147	0.072
2			∞	1.810	1.181	0.785	0.566	0.407	0.302	0.222
3					2.285	1.576	0.910	0.726	0.467	0.382
4					∞	2.994	1.317	1.095	0.642	0.551
5							1.821	1.540	0.829	0.732
6							2.499	2.103	1.031	0.926
7							3.605	2.895	1.250	1.136
8							∞	4.316	1.490	1.365
9									1.756	1.616
10									2.055	1.896
11									2.398	2.214
12									2.804	2.583
13									3.305	3.025
14									3.978	3.586
15									5.069	4.371
16									∞	5.768

Table 2.2b Optimum Quantizers with Laplacian Density

$$(m_x = 0, \sigma_x^2 = 1)$$

N	2		4		8		16		32	
	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i	x_i	x'_i
1	∞	0.577	1.205	0.302	0.504	0.149	0.229	0.072	0.101	0.033
2			∞	2.108	1.401	0.859	0.588	0.386	0.252	0.169
3					2.872	1.944	1.045	0.791	0.429	0.334
4					∞	3.799	1.623	1.300	0.630	0.523
5							2.372	1.945	0.857	0.737
6							3.407	3.798	1.111	0.976
7							5.050	4.015	1.397	1.245
8							∞	6.085	1.720	1.548
9									2.089	1.692
10									2.517	2.287
11									3.022	2.747
12									3.633	3.296
13									4.404	3.970
14									5.444	4.838
15									7.046	6.050
16									∞	8.043

Table 2.2c. Optimum Quantizers with Gamma Density

$$(m_x = 0, \sigma_x^2 = 1)$$

Fig 2.6 shows a 3-bit quantizer for a Laplacian density. It is clear from this figure that the quantization levels get further apart as the probability density decreases. This is consistent with intuition which would suggest that the largest quantization errors should be reversed for the least frequently occurring samples.

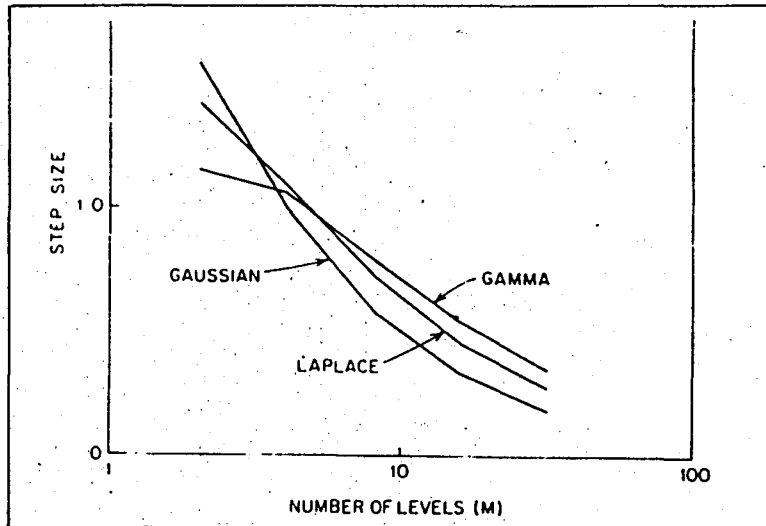


Fig 2.7 Optimum stepsizes for a uniform quantizer for Laplace, Gamma and Gaussian density functions

Fig 2.7 which is taken from the study of Max [48], shows the optimum step size for uniform quantizers for Gamma, Laplacian and Gauss densities. It is clear that, the step size decreases roughly exponentially with increasing number of bits.

In transmission systems during periods when there is no signal, i.e., the idle channel condition, the input to the quantizer is very small (assuming low noise) so that the output of the quantizer will jump back and forth between the lowest magnitude quantization levels. For a symmetric quantizer, if the lowest quantization levels are greater than the amplitude of the background noise, the output noise of the quantizer will be greater than the input noise.

2.1.2 ADAPTIVE QUANTIZATION

Better results are obtain if the quantization step size is chosen large enough to accomodate the maximum peak-to-peak range of the signal. On the other hand it is also desirable to make the quantization steps small enough so as to minimize the quantization noise. The idea is to work with a basic quantizer which is very simple (uniform, if necessary) but to modify its step size by a factor depending on the knowledge of which quantizer slots were occupied by the previous samples.

In this section, some general principles of adaptive quantization are discussed and in later sections, some examples of adaptive quantization schemes in conjunction with linear prediction will be explained.

The basic idea of adaptive quantization is to let the step size Δ (or in general the quantizer levels and ranges) vary so as to match the variance of the input signal.

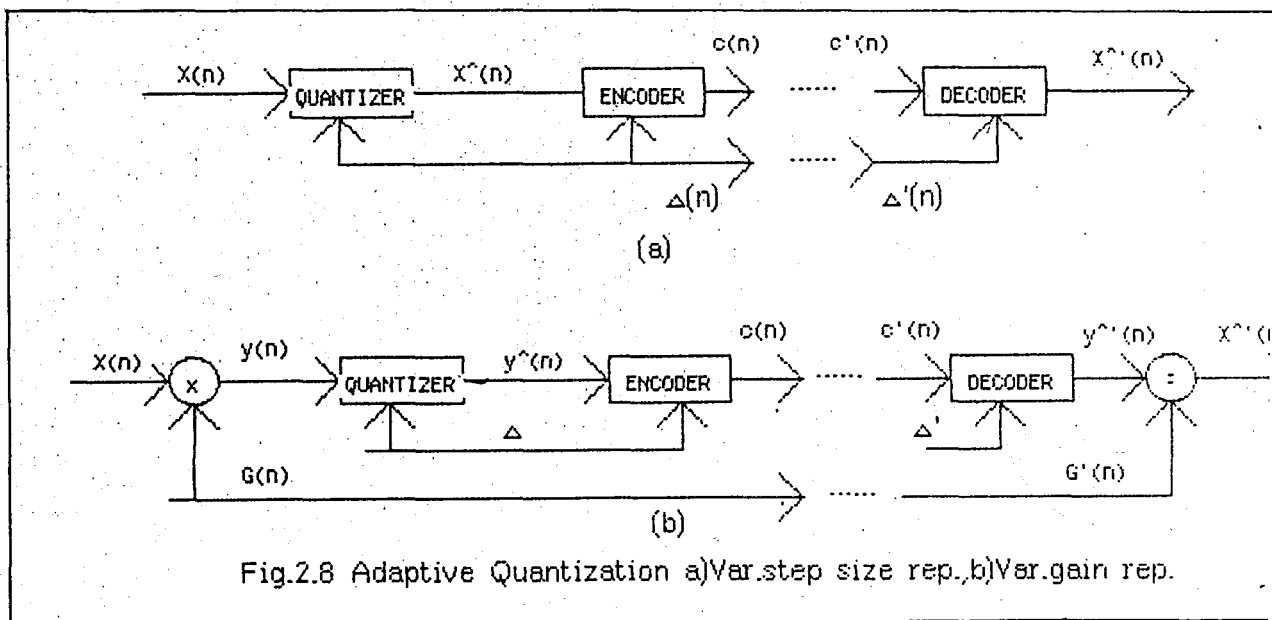
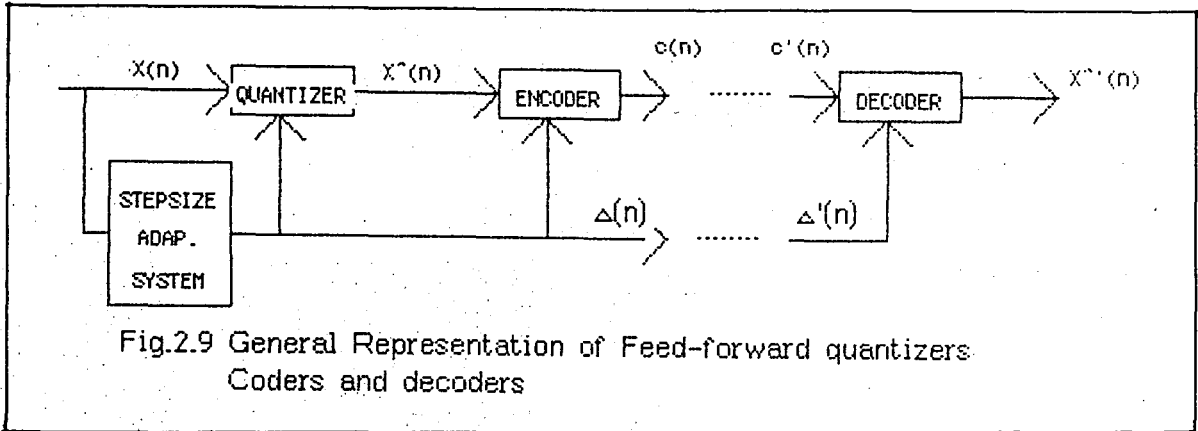


Fig.2.8 Adaptive Quantization a)Var.step size rep.,b)Var.gain rep.

This is depicted in Fig (2.8a). An alternative point of view (in Fig (2.8b)) is to consider a fixed quantizer characteristic preceeded by a time varying gain which tends to keep the variance constant. In the first case the step size should increase and decrease with increases and decreases of the variance of the input. In the case of a nonuniform quantizer this would imply that the quantization levels and ranges would be scaled linearly to match the variance of the signal. In the second point of view, which applies without modification to both uniform and nonuniform quantizers, the gain changes inversely with changes in the variance of the input so as to keep the variance of the quantizer input relatively constant. In either case it is necessary to obtain an estimate of the time-varying amplitude properties of the input signal.

In discussing adaptive quantizing schemes, it will likewise be convenient to classify them according to whether they are adapting slowly or rapidly. In one class of schemes the amplitude or variance of the input is estimated from the input itself. Such schemes are called " Feed-forward adaptive quantizers". In the other class of adaptive quantizers the step size is adapted on the basis of the output of the quantizer, $\hat{x}(n)$, or equivalently, on the basis of the output code words, $c(n)$. These are called feedback quantizers.

FEED-FORWARD ADAPTIVE QUANTIZATION



The above figure depicts a general representation of the class of feed-forward quantizers. For convenience, we assume that the quantizer is uniform so that it is sufficient to vary a single step size parameter. It is straightforward to generalize this discussion to the case of nonuniform quantizers. The step-size $\Delta(n)$, used to quantize the sample $x(n)$, must be available at the receiver. Thus, the code words $c(n)$ and the step-size $\Delta(n)$ together represent the sample $x(n)$. If $c'(n) = c(n)$ and $\Delta'(n) = \Delta(n)$ then $\hat{x}'(n) = \hat{x}(n)$, but, if $c'(n) \neq c(n)$ or $\Delta'(n) \neq \Delta(n)$, i.e., if there are errors in transmission then $\hat{x}'(n) \neq \hat{x}(n)$. The effect of errors will depend upon the details of the adaptation scheme.

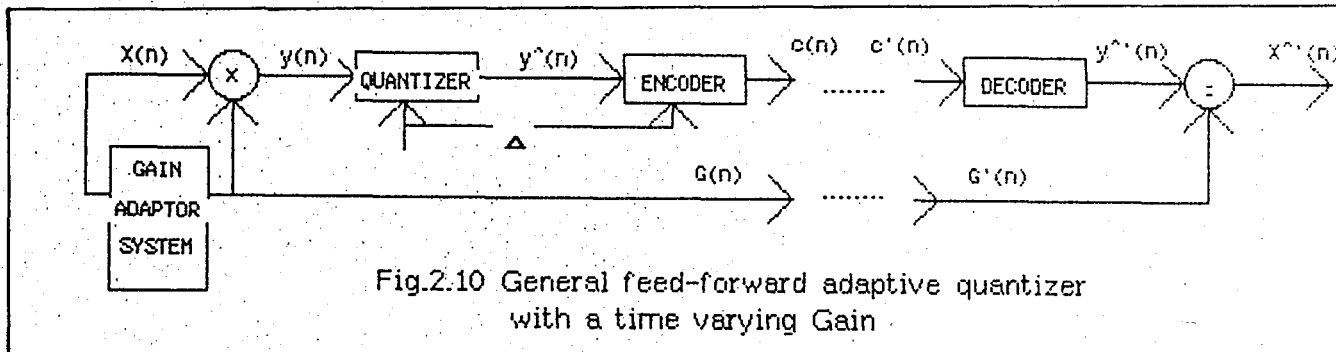


Fig 2.10 shows the general feed-forward adaptive quantizer represented in

terms of a time-varying gain. In this case, the code words $c(n)$ and the gain $G(n)$ together represent the sample.

Most systems of this type attempt to obtain an estimate of the time-varying variance. A common approach is to assume that the variance is proportional to the short-time energy, which, as we have seen, is defined as the output of a low-pass filter with input $x^2(n)$. That is,

$$\sigma^2(n) = \sum_{m=-\infty}^{\infty} x^2(m) h(n-m) \quad (2.33)$$

where $h(n)$ is the impulse response of the low-pass filter. (For a stationary input signal, it can be easily shown that the expected value of $\sigma^2(n)$ is proportional to the variance σ_x^2 .)

A simple example is

$$h(n) = \begin{cases} \alpha^{n-1} & n \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.34)$$

using this in equation (2.33) gives

$$\sigma^2(n) = \sum_{m=-\infty}^{n-1} x^2(m) \alpha^{n-m-1} \quad (2.35)$$

It can be shown that the $\sigma^2(n)$ in equation (2.35) also satisfy the difference equation $\sigma^2(n) = \alpha \sigma^2(n-1) + x^2(n-1)$ (past samples). For stability we require $0 < \alpha < 1$. The step size in Fig (2.9) would therefore be of the form

$$\Delta(n) = \Delta_0 \sigma(n) \quad (2.36)$$

or the time-varying gain in Fig (2.10) would be of the form

$$G(n) = \frac{G_0}{\sigma(n)} \quad (2.37)$$

The choice of the parameter α controls the effective interval that contributes to the variance estimate. It is important to consider the lowest possible sampling rate for the gain, since the information rate of the digital representation is the sum of the information rate of the gain function. The gain function (or step size) as used in Fig (2.9) or (2.10) must be sampled and quantized before transmission.

To permit quantizing and because of constraints of physical implementations, it is common to limit the variation of the gain function or the stepsize. That is, we define limits on $G(n)$ and $\Delta(n)$ of the form

$$G_{\min} \leq G(n) \leq G_{\max} \quad (2.38)$$

$$\Delta_{\min} \leq \Delta(n) \leq \Delta_{\max} \quad (2.39)$$

It is the ratio of these limits that determines the "dynamic range" of the system. Thus, to obtain a relatively constant SNR over a range 40 dB, requires $G_{\max}/G_{\min} = 100$ or $\Delta_{\max}/\Delta_{\min} = 100$.

An example of the improvement in SNR that can be achieved by adaptive quantization is given in a comparative study by Noll [50]. A feed-forward scheme is considered in which the variance estimate was

$$\sigma^2(n) = \frac{1}{M} \sum_{m=n}^{n+M-1} x^2(m) \quad (2.40)$$

The gain or stepsize is evaluated and transmitted for every M samples. In this case the system requires a buffer of M samples to permit the quantizer gain or stepsize to be determined in terms of the samples that are to be quantized rather than in terms of past samples as in the previous example.

Table 2.3, which is taken from the study of Noll [50], shows a comparison of various 3-bit quantizers with a speech input of known variance. The first column lists the various quantizer types. The second column gives the SNR ratios with no adaptation. The third and fourth columns give the SNR ratios for stepsize adaptation based upon the variance estimate of Eq.(2.40) with M=128 and M=1024, respectively.

Nonuniform Quan.	Nonadaptive SNR (dB)	Adaptive, M=128 SNR (dB)	Adaptive, M=1024 SNR (dB)
Gaussian	7.3	15.0	12.1
Laplace	9.9	13.3	12.8
Uniform Quan.			
Gaussian	6.7	14.7	11.3
Laplace	7.4	13.4	11.5

Table 2.3 Adaptive 3-bit Quantization with Feedforward Adap.
(After Noll[50])

It can be readily seen that the adaptive quantizer achieves up to 8.0 dB better SNR. Thus, it is evident that adaptive quantization achieves a definite advantage over fixed nonuniform quantizers. An additional advantage is, that by appropriately choosing Δ_{\min} and Δ_{\max} it is possible to achieve the improvement in SNR while maintaining low idle channel noise and wide dynamic range.

Examples of feed-forward adaptation will be given in section 2.5

FEEDBACK ADAPTATION

The second class of adaptive quantizers is depicted in Fig (2.11) and (2.12) where it is noted that the variance of the input is estimated from the quantizer output or equivalently from the codewords.

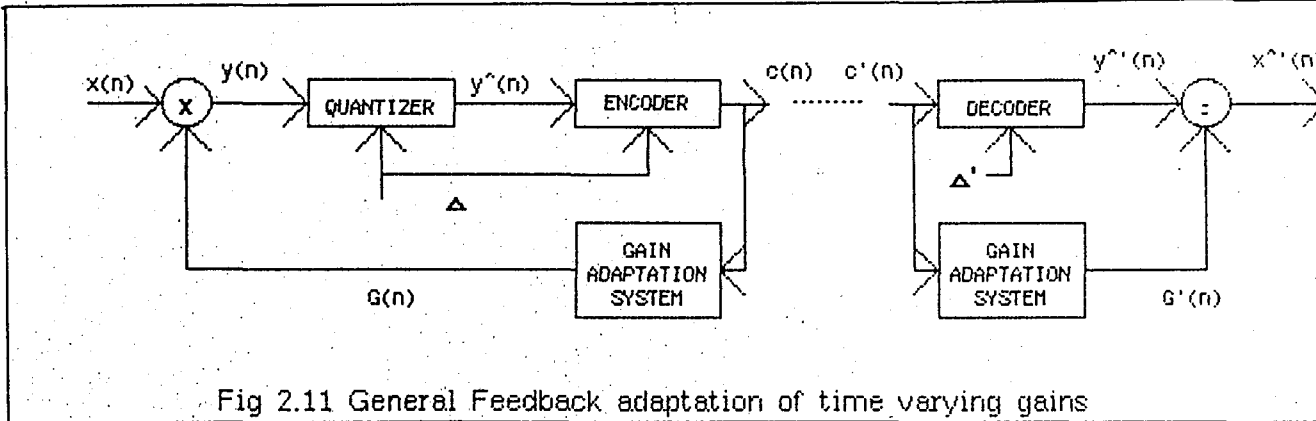


Fig 2.11 General Feedback adaptation of time varying gains

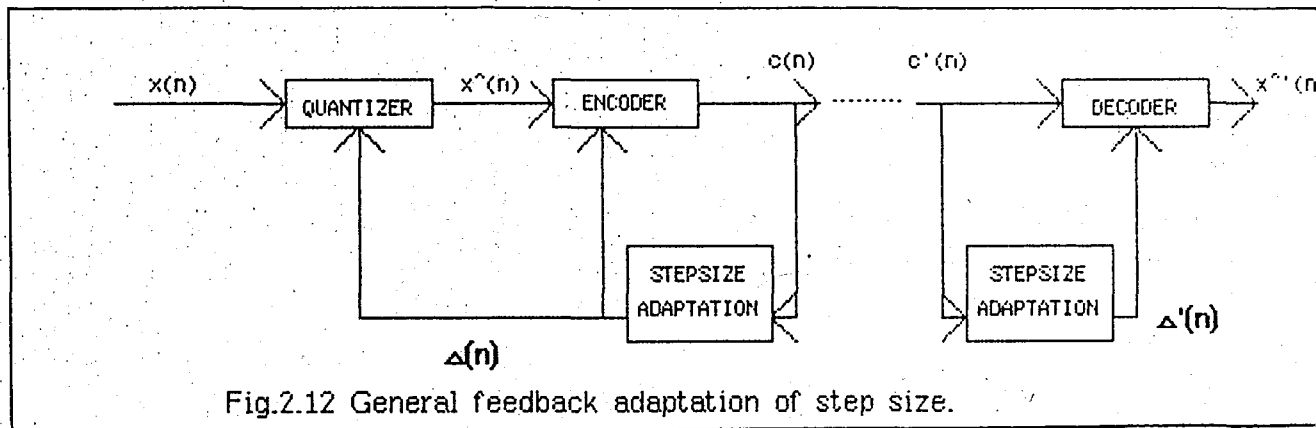


Fig.2.12 General feedback adaptation of step size.

As in the case of feedforward systems, the stepsize and gain are proportional and inversely proportional respectively to an estimate of the standard deviation of the input as in Eq (2.36) and (2.37). Such schemes have the distinct advantage that the stepsize or gain need not be explicitly retained or transmitted since they can be derived from the sequence of codewords. The disadvantage of such systems is the increased sensitivity to

errors in the codewords, since such errors imply not only an error in the quantizer levels but also in the stepsize.

One simple approach is to apply Eq (2.33) directly to the quantizer output, i.e.,

$$\sigma^2(n) = \sum_{m=-\infty}^{\infty} \hat{x}^2(m) h(n-m) \quad (2.41)$$

In this case, however, it will not be possible to use buffering to implement a noncausal filter. That is, the variance estimate must be based only on past values of $\hat{x}(n)$ since the present value of $\hat{x}(n)$ will not be available until the quantization has occurred, which in turn must be after the variance has been estimated. For example, we could use a filter whose impulse response is

$$h(n) = \begin{cases} \alpha^{n-1} & n \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.42)$$

as in Eq (2.35). Alternatively the filter might have an impulse response

$$h(n) = \begin{cases} 1/M & 1 \leq n \leq M \\ 0 & \text{otherwise} \end{cases} \quad (2.43)$$

so that

$$\sigma^2(n) = 1/M \sum_{m=n-M}^{n-1} x^2(m) \quad (2.44)$$

This system is studied by Noll, who found out that with suitable adjustments of the constants Δ_0 and G_0 in Eq (2.36) and (2.37) a SNR ratio of the order of 12 dB could be obtained for a 3-bit quantizer with a window length of only two samples. Larger values of M produced only slightly better results.

A different approach has been studied extensively by Jayant. In this method the stepsize of a uniform quantizer is adapted at each sample.

JAYANT ALGORITHM (Quantization with one word memory)

More recently studied, and presumably more flexible, means of matching quantizer step size to signal variance is the use of step size adaptation based on quantizer memory. The idea is to work with a basic quantizer that is very simple (uniform if necessary), but to modify its step size (for every new input sample, in general) by a factor depending on the knowledge of which quantizer slots were occupied by the previous samples.

In its simplest form, the scheme operates with a one word memory. Let the output of a B-bit (uniform) quantizer be

$$y_n = H_n \frac{\Delta_n}{2} \quad \text{where } H_n = 1, 3, 5, \dots, 2^{B-1} \quad (2.45)$$

and $(\Delta_n > 0, B \geq 2)$

The step size Δ_{n+1} is chosen to be the previous step size multiplied by a time-invariant function of the code word magnitude $|H_n|$

$$\Delta_{n+1} = \Delta_n M(|H_n|) \quad (2.46)$$

When the multiplier function is properly designed, the adaptation logic serves to match the step size, at every sample, to an updated estimate of signal variance. Fig (2.13) explains a 3-bit adaptive quantizer.

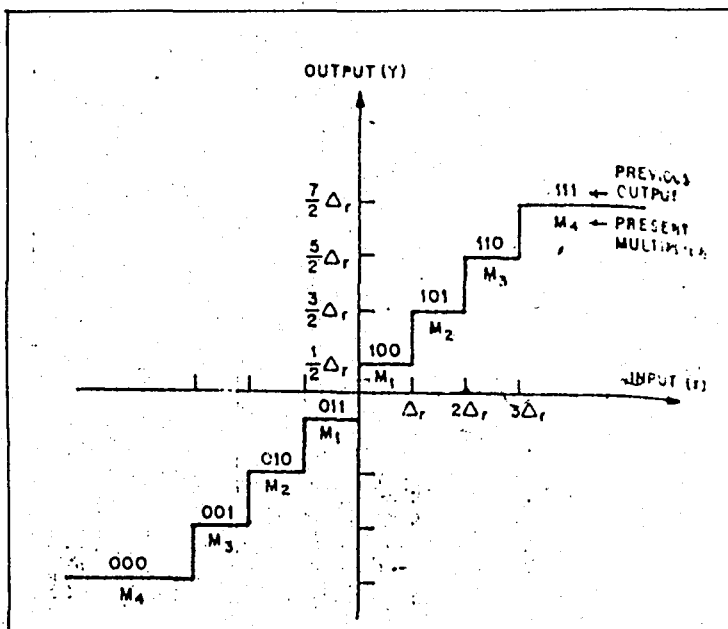


Fig 2.13 Adaptive quantization-I/O characteristics
of Jayant's Quantization Scheme

Shown in the figure are the eight possible values of (the latest) coder output (000 through 111) and the corresponding step size multipliers M . Note that the value of M depends on the magnitude of (the latest) coder output - equivalently on the magnitude $|H_n|$ in Eq.(2.45) - and not on the sign of the output. (This strategy is a simple consequence of the observation that the input probability density function $p(x)$ is expected to be symmetric about a mean value of zero).

Fig (2.14) shows histograms of stepsizes encountered in the simulation of a 4-bit adaptive quantizer with a Gaussian signal at the input which is taken from the study of Jayant [51]. The step size multipliers were selected to maximize an appropriately defined SNR and Fig (2.14) shows how these multipliers indeed maintain the variable Δ in a region centered on Δ_{opt} , the optimal (constant) step size for a nonadaptive quantizer (even when the starting step size is severely suboptimal).

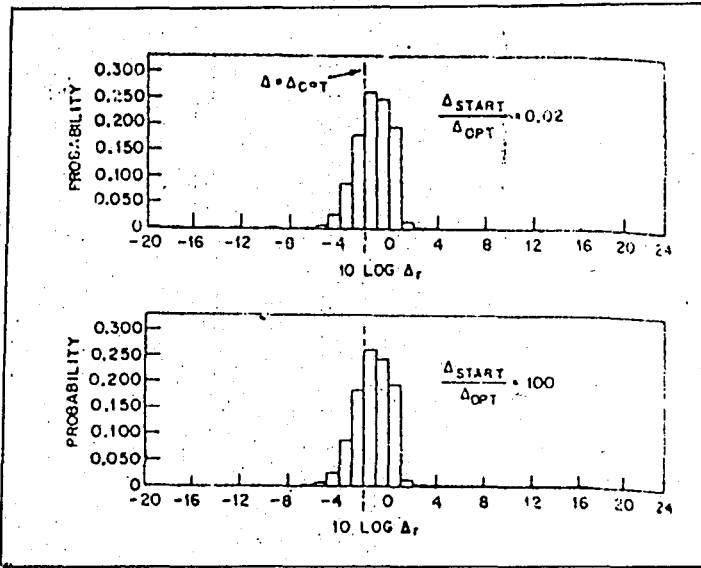


Fig 2.14 Histogram of step sizes in adaptive quantization of Gauss-Markov input [51]

CODER	PCM			DPCM		
	2	3	4	2	3	4
M ₁	0.60	0.85	0.80	0.80	0.90	0.90
M ₂	2.00	1.00	0.80	1.60	0.90	0.90
M ₃		1.00	0.85		1.25	0.90
M ₄		1.50	0.80		1.75	0.90
M ₅			1.20			1.20
M ₆			1.60			1.60
M ₇			2.00			2.00
M ₈			2.40			2.40

Table 2.4 Stepsize Multipliers for B=2,3,4 (After Jayent [51])

Table (2.4) lists step size multipliers found to be optimal for adaptive PCM coding of a low-pass filtered speech sample. The recommended step size multipliers do not in general constitute overly critical target values.

It is possible to get away, for instance, with trivial values of unity for

some of the central multipliers. For example

$$M_1 < 1 ; M_2 = M_3 = 1 ; M_4 > 1 \quad \text{for } B=3 \quad (2.47)$$

However, the step size increases should, in general, be more rapid than step size decreases. This has to do with the following comparison of two basic types of quantization errors: "overload" errors that occur when Δ_n is too small, and a signal sample falls outside the quantizer range, and "granular" errors that are inherent in quantization even when the input falls within a quantizer slot or step. Granular errors tend to be less harmful to SNR than overload errors. To mitigate the contribution of the latter to the total noise power, one seeks to correct the occurrence of overload errors more expeditiously. The end result is an optimal multiplier function that has the general form depicted in Fig (2.15)

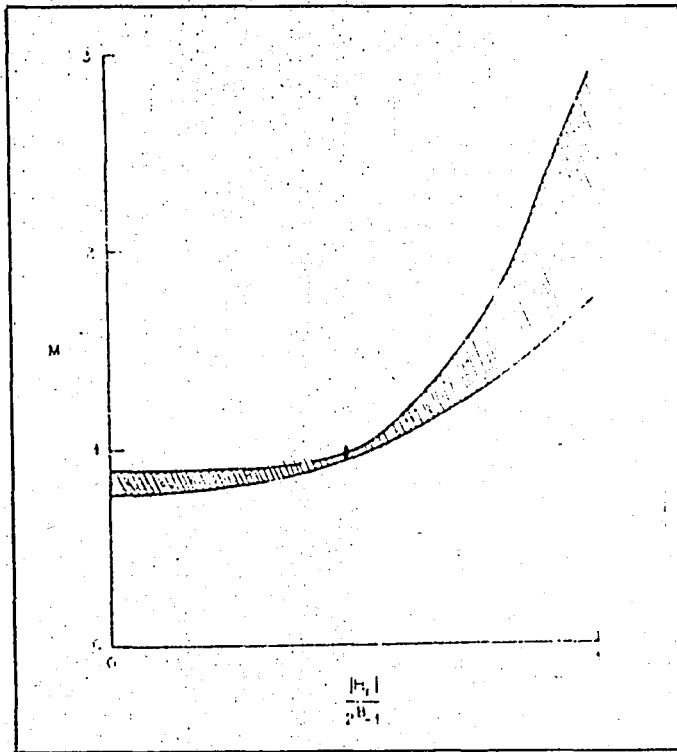


Fig 2.15 General shape of optimal multiplier function; $B > 2$

Fig (2.15) shows that

$$\Delta_{nm} = \Delta_n M(|H_n|) \quad (2.48)$$

steadily increases with $|H_n|$ during step size increases ($M > 1$; to correct for overload). On the other hand, during step size decreases ($M < 1$) the optimal M function is remarkably close to unity. The shaded region in Fig (2.15) expresses variations in the Jayant multiplier function when there are changes in B and the input statistics. The exclusion of B=2 from the adaptation rule in the figure has to do with the fact that when the quantization gets to be crude enough, the distinction between expected magnitudes of granular and overload errors diminishes and so does the disparity between desired rates of step size decrease and increase.

BLOCK QUANTIZATION (Quantization with memory)

In block quantization, more commonly considered for image digitization, a block of k input samples $(x_1, x_2, \dots, x_k) = x$ (which may be regarded as a vector in k dimensions) is simultaneously quantized, producing an output vector $(y_{n1}, y_{n2}, \dots, y_{nk}) = y_n$ approximating x. Thus the output y_{nm} is an approximation to x_m for each $m=1, 2, \dots, k$. An N-point quantizer selects one of N output "points" y_1, y_2, \dots, y_n to approximate x. Unlike zero-memory quantization, the value y_{nm} depends not only on the corresponding input sample x_m , but also on the values of all other samples x_m in the block. Even if the input samples are statistically independent, an advantage can be gained by quantizing a block at a time, rather than a sample at a time. A convenient measure of the distortion of the block quantizer is

$$D = 1/k \sum_{n=1}^k e_n^2 \quad (2.49)$$

where \bar{e}_n^2 is the mean-square error in the n th sample. Such an approach was analyzed by Berger [53], Kramer and Mathews [54] and Huang and Schultheiss [55]. The basic scheme for image transmission is as follows. A block of N data samples x_n are linearly transformed into y_n by an $N \times N$ matrix A . The y_n are quantized and transmitted. At the receiver, the quantized y_n are transformed by another $N \times N$ matrix B into z_n . For a given bit rate, the matrices A and B are chosen to minimize the mean-square error between z_n and x_n . It turns out that the optimum matrix A consists of the eigenvector of the correlation matrix of the samples x_n and the optimum matrix B is the inverse of A . It appears however that a much simpler type of matrices, the so-called Hadamard matrices work almost as well as the optimum [56]. A Hadamard matrix contains only $+1$ and -1 as its elements and is orthogonal. The performance of block quantization could be compared with zero-memory quantization by examining how the bit rate or average number of bits per sample, $B = \log_2 N/K$, depends on D , the distortion per sample. Clearly, as the block length k increases, the minimum bit rate needed for a given distortion will decrease. In the limit as $k \rightarrow \infty$, the minimum bit rate B approaches a limiting value R depending on D .

Another class of quantizers with memory are the sequential or differential quantizers such as delta modulation, differential PCM and the various adaptive versions of these schemes. These will be discussed in the succeeding sections. Although the quantization techniques are given in the preceding sections of this chapter, most of the time, different type of quantizers are used together in the same transmission process. I.e., quantization can be performed by using uniform and adaptive, or nonuniform and adaptive quantizers. One can even use uniform and adaptive quantizers in block quantization. Some examples will be given in section 2.5.

2.2 GENERAL THEORY OF SEQUENTIAL OR DIFFERENTIAL QUANTIZATION

If we consider image samples, there is a considerable correlation between adjacent image samples, and indeed the correlation is significant even between samples that are several sampling intervals apart. The meaning of this high correlation is that, the signal does not change rapidly from sample to sample so that the difference between adjacent samples should have a lower variance than the variance of the signal itself. This fact provides the motivation for the general differential quantization scheme. In this system, the output to the quantizer is a signal

$$d(n) = x(n) - \tilde{x}(n) \quad (2.50)$$

which is the difference between the unquantized input sample $x(n)$, and an estimate or prediction of the input sample which is denoted by $\tilde{x}(n)$. This predicted value is the output of a predictor system P , whose input is a quantized version of the input signal, $x(n)$. This difference signal can be called as the prediction error signal, since it is the amount by which the predictor fails to exactly predict the input value. Temporarily leaving aside the question of how the estimate $\tilde{x}(n)$ is obtained, we note that it is the difference signal that is quantized rather than the input. The quantizer could be either fixed or adaptive, uniform or nonuniform, but in any case its parameters should be adjusted to match the variance of $d(n)$. The quantized difference signal can be represented as

$$\hat{d}(n) = d(n) + e(n) \quad (2.51)$$

where $e(n)$ is the quantization error. According to Fig (2.16a) the quantized difference signal is added to the predicted value $\hat{x}(n)$ to produce a quantized version of the input, i.e.,

$$\hat{x}(n) = \hat{x}(n) + \hat{d}(n) \quad (2.52)$$

Substituting Eq (2.50) and (2.51) into Eq (2.52), it can be seen that

$$\hat{x}(n) = x(n) + e(n) \quad (2.53)$$

That is, independent of the properties of the system labeled P, the quantized image sample differs from the input only by the quantization error of the difference signal. Thus, if the prediction is good, the variance of $d(n)$ will be smaller than the variance of $x(n)$ so that a quantizer with a given number of levels can be adjusted to give a smaller quantization error than would be possible when quantizing the input directly.

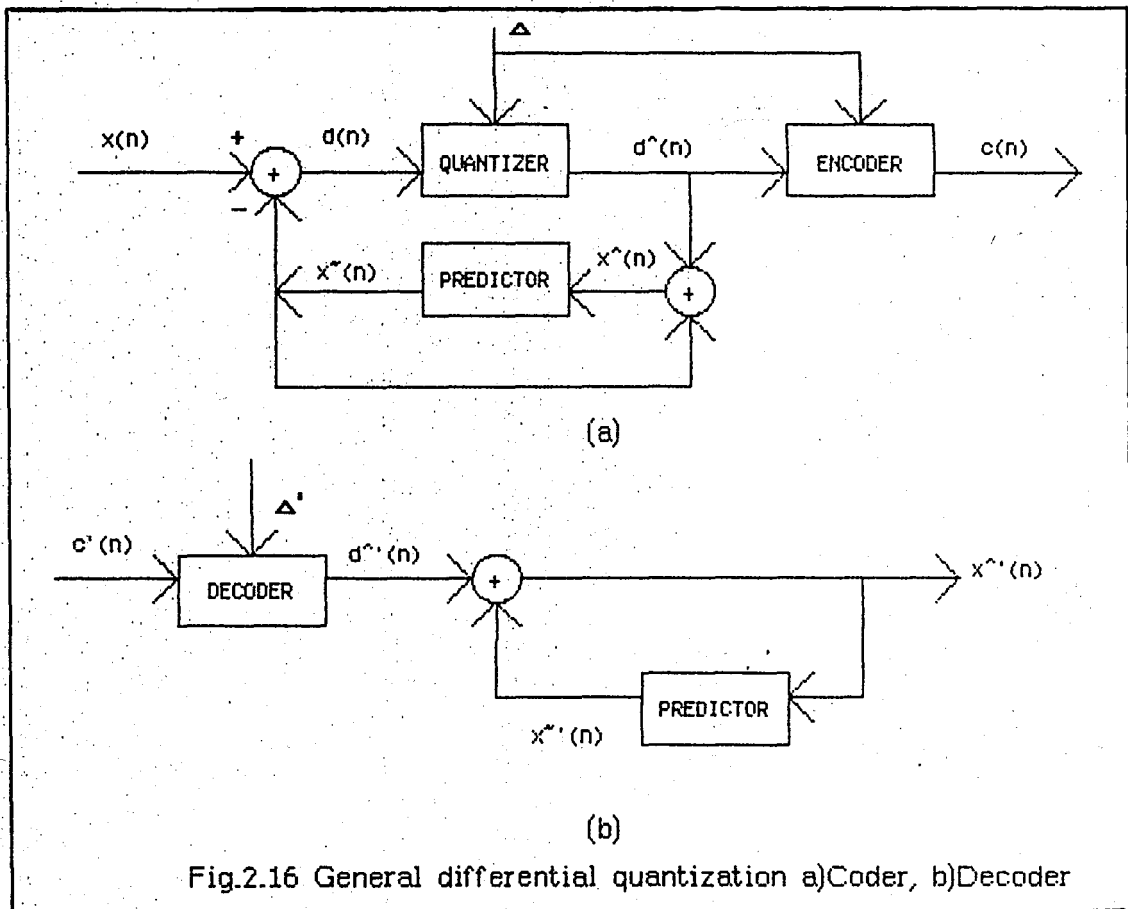


Fig.2.16 General differential quantization a)Coder, b)Decoder

It should be noted that it is the quantized difference signal that is coded for transmission or storage. The system for reconstructing the quantized input from the code words is implicit in Fig (2.16a). This system, depicted in picture in Fig (2.16b), involves a decoder which reconstructs the quantized difference signal from which the quantized input is reconstructed using the same predictor as used in Fig (2.16a). Clearly, if $c'(n)$ is identical to $c(n)$ then $\hat{x}'(n)=\hat{x}(n)$, which differs from $x(n)$ only by the quantization error incurred in quantizing $d(n)$.

The signal-to-quantization noise ratio of the system of Fig (2.16) is, by definition,

$$\text{SNR} = \frac{E[x^2(n)]}{E[e^2(n)]} = \frac{\sigma_x^2}{\sigma_e^2} \quad (2.54)$$

which can be written as

$$\text{SNR} = \frac{\sigma_x^2}{\sigma_d^2} \cdot \frac{\sigma_d^2}{\sigma_e^2} = G_p \text{SNR}_q \quad (2.55)$$

where

$$\text{SNR}_q = \frac{\sigma_d^2}{\sigma_e^2} \quad (2.56)$$

is the signal-to-quantizing noise ratio of the quantizer, and the quantity

$$G_p = \frac{\sigma_x^2}{\sigma_d^2} \quad (2.57)$$

is defined as the gain due to the differential configuration. The quantity SNR_q is dependent upon the particular quantizer that is used, and, given knowledge of the properties of $d(n)$, SNR_q can be maximized by using the

techniques of the previous section. The quantity G_p , if greater than unity, represents the gain in SNR that is due to the differential scheme. Clearly, the objective should be to maximize G_p by appropriate choice of the predictor system P . For a given signal, σ_x^2 is a fixed quantity so that G_p can be maximized by minimizing the denominator of Eq (2.57), i.e., by minimizing the variance of the predictor error.

In the following section, the concept of linear prediction will be explained before entering Differential Pulse Code Modulation, DPCM, in more detail.

DELTA MODULATION

A few words should be said about Delta Modulation even though it is not included in the subject of this study [67]. Delta Modulation, DM, is the special case of DPCM to be discussed in Sec.2.4, in which the difference level is 1. This has been of interest because of the simplicity of the equipment. Fairly extensive testing has shown that the sampling rate must be substantially higher than twice the bandwidth for acceptable quality. For a given data rate, the quality is almost always inferior to three-bit DPCM except for some low SNR systems, and is even inferior to ordinary PCM in some low SNR systems. Of course, exceptions can be found which favour any particular system.

The remarks above concerning the trade-off between channel noise performance and quality by means of varying the integrator time constant apply to the one-bit case with at least equal force. The reason why delta modulation is generally less efficient than PCM and DPCM for image signals is not hard to see. These signals, although they have little energy at the high end of

their spectrum, do have sharp transitions at the edge of objects in the scene. Even though the total energy per frame in these transitions is small, it is on their proper rendition that the respective sharpness of the picture depends. The attempt to reproduce these sharp transitions in a delta modulation system, for example by increasing the pulse height, introduces granular noise (and contouring if the integrator time constant is short). Further more, progress in electronic technology, in particular, digital integrated circuits, has greatly reduced the cost and reliability advantages of delta modulation over PCM.

2.3 LINEAR PREDICTION

Here, the nature of the predictor is specified. The predicted value $\tilde{x}(n)$ is a linear combination of past quantized values.

$$\tilde{x}(n) = \sum_{k=1}^N \alpha_k \hat{x}(n-k) \quad (2.58)$$

where α_k are the k th order predictor coefficients. The predictor coefficients can be found from the autocorrelation values of the system, which can be calculated from the equation:

$$R(k) = E\{x(n) x(n+k)\} \quad (2.59)$$

If we consider the first order predictor, we must calculate $R(0)$ and $R(1)$ which are

$$R(0) = 1/N \sum_{i=1}^N x(i) x(i) \quad (2.60)$$

$$R(1) = 1/(N-1) \sum_{i=1}^{N-1} x(i) x(i+1) \quad (2.61)$$

$$\alpha_1 = R(1)/R(0) \quad (2.62)$$

But if a second or higher order predictor is required, more complicated arithmetic work has to be done. We denote prediction error as $d(n)$ and the variance of the prediction error as σ_d^2 . Then one can write:

$$\begin{aligned} \sigma_d^2 &= E\{d^2(n)\} = E\{(x(n) - \tilde{x}(n))^2\} \\ &= E\{(x(n) - \sum_{k=1}^N \alpha_k \hat{x}(n-k))^2\} \\ &= E\{(x(n) - \sum_{k=1}^N \alpha_k x(n-k) - \sum_{k=1}^N \alpha_k e(n-k))^2\} \quad (2.63) \end{aligned}$$

In order to choose a set of predictor coefficients $\{\alpha_j\}$ where $1 < j < N$, that minimize σ_d^2 , we must differentiate σ_d^2 with respect to each parameter and set the derivative equal to zero. Thereby a set of N equations is obtained.

$$\frac{\partial \sigma_d^2}{\partial \alpha_j} = -2E\{(x(n) - \sum_{k=1}^N \alpha_k (x(n-k) + e(n-k))) - [x(n-j) + e(n-j)]\} = 0$$

where $1 < j < N$

$$\frac{\partial \sigma_d^2}{\partial \alpha_j} = E\{(x(n) - \tilde{x}(n)) \hat{x}(n-j)\} = E\{d(n) \hat{x}(n-j)\} = 0 \quad (2.64)$$

The above equation can be expanded into the set of N equations:

$$\begin{aligned} E\{x(n-j)x(n)\} + E\{e(n-j)x(n)\} &= \sum_{k=1}^N \alpha_k E\{x(n-j)x(n-k)\} \\ &+ \sum_{k=1}^N \alpha_k E\{e(n-j)x(n-k)\} \\ &+ \sum_{k=1}^N \alpha_k E\{x(n-j)e(n-k)\} \\ &+ \sum_{k=1}^N \alpha_k E\{e(n-j)e(n-k)\} \quad (2.65) \end{aligned}$$

where $1 \leq j \leq N$. We assume that $e(n)$ is uncorrelated to $x(n)$ and $e(n)$ is a stationary white noise sequence, then the above equation can be simplified to:

$$E\{x(n-j)x(n)\} = R(j) = \sum_{k=1}^N \alpha_k [E\{x(n-j)x(n-k)\} + \sigma_e^2 \delta(j-k)] \quad (2.66a)$$

where $E\{x(n-j)x(n)\} = R(j) \quad (2.66b)$

$$E\{x(n-j)x(n-k)\} = R(j-k) \quad (2.66c)$$

$$E\{e(n-j)e(n-k)\} = \sigma_e^2 \delta(j-k) \quad 1 \leq j \leq N \quad (2.66d)$$

Then the equation becomes:

$$R(j) = \sum_{k=1}^N \alpha_k [R(j-k) + \sigma_e^2 \delta(j-k)] \quad 1 \leq j \leq N \quad (2.67)$$

where $R(j)$ is the autocorrelation function of $x(n)$. The normalized autocorrelation is

$$\rho(j) = R(j) / \sigma_x^2 \quad (2.68)$$

the Eq (2.67) can be expressed in matrix form

$$\underline{\rho} = \underline{C} \underline{\alpha} \quad (2.69a)$$

where

$$\underline{\rho} = \begin{bmatrix} \rho(1) \\ \rho(2) \\ \cdot \\ \cdot \\ \cdot \\ \rho(N) \end{bmatrix} \quad (2.69b)$$

$$\underline{\underline{C}} = \begin{bmatrix} 1+1/\text{SNR} & \rho(1) & \cdot & \cdot & \cdot & \rho(N-1) \\ \rho(1) & 1+1/\text{SNR} & \cdot & \cdot & \cdot & \rho(N-2) \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \rho(N-1) & \rho(N-2) & \cdot & \cdot & \cdot & 1+1/\text{SNR} \end{bmatrix} \quad (2.69c)$$

$$\underline{\underline{\alpha}} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \cdot \\ \cdot \\ \alpha_N \end{bmatrix} \quad (2.69d)$$

where

$$\text{SNR} = \frac{\sigma_x^2}{\sigma_e^2} \quad (2.70)$$

Then the predictor coefficients of the Nth order predictor can be found from

$$\underline{\underline{\alpha}} = \underline{\underline{C}}^{-1} \underline{\underline{\rho}} \quad (2.71)$$

In general, the matrix $\underline{\underline{C}}^{-1}$ can be computed by a variety of numerical methods, because there is an advantage of the fact that $\underline{\underline{C}}$ is a Toeplitz matrix.

However, Eq. (2.69a) can not be solved in the most general case since the matrix $\underline{\underline{C}}$ contains terms which depend on the signal-to-noise ratio, $\text{SNR} = \frac{\sigma_x^2}{\sigma_e^2}$ (see Eq (2.69c)); but SNR depends on the coefficients of the linear predictor, which in turn depends on SNR through Eq (2.69a). One possibility is to neglect the term $1/\text{SNR}$ in Eq (2.69) in order to obtain a solution. For the case $N=1$, however, such an assumption is unnecessary since Eq (2.71) can be directly solved to give

$$\alpha_1 = \frac{\rho(1)}{1+1/\text{SNR}} \quad (2.72)$$

Eq (2.72) shows that $\alpha_1 < \rho(1)$. But, for $N=4$, the predictor coefficients can be solved by

$$\underline{\underline{C}} = \begin{bmatrix} 1 & \rho(1) & \rho(2) & \rho(3) \\ \rho(1) & 1 & \rho(1) & \rho(2) \\ \rho(2) & \rho(1) & 1 & \rho(1) \\ \rho(3) & \rho(2) & \rho(1) & 1 \end{bmatrix} \quad \underline{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} \quad (2.73)$$

In spite of the difficulties in solving explicitly for the predictor coefficients, it is possible to obtain an expression for the optimum gain, G_p in terms of the α_k 's. To do this we solve for σ_d^2 by rewriting Eq (2.63) in the form

$$\begin{aligned} \sigma_d^2 &= E[(x(n) - \tilde{x}(n))]^2 \\ \sigma_d^2 &= E[(x(n) - \tilde{x}(n))x(n)] - E[(x(n) - \tilde{x}(n))\tilde{x}(n)] \end{aligned} \quad (2.74)$$

Using Eq (2.64) it is straightforward to show that for the optimum predictor coefficients, the second term in the above equation is zero, i.e., the predicted value is also uncorrelated with the prediction error. Thus, it can be written.

$$\begin{aligned} \sigma_d^2 &= E[(x(n) - \tilde{x}(n))x(n)] \\ \sigma_d^2 &= E[x^2(n)] - E\left[\sum_{k=1}^N \alpha_k (x(n-k) + e(n-k))x(n)\right] \end{aligned} \quad (2.75)$$

Using the assumptions of uncorrelated signal to noise, we obtain

$$\sigma_d^2 = \sigma_x^2 - \sum_{k=1}^N \alpha_k R(k) = \sigma_x^2 \left[1 - \sum_{k=1}^N \alpha_k \rho(k)\right] \quad (2.76)$$

Thus, from Eq (2.57)

$$(G_p)_{\text{opt}} = 1 / (1 - \sum_{k=1}^N \alpha_k \rho(k)) \quad (2.77)$$

where the α_k 's satisfy Eq (2.69a). For the case $N=1$ we can examine the effects of using a suboptimum value of α_1 on the quantity $G_p = \sigma_x^2 / \sigma_d^2$. From Eq (2.77) we get

$$(G_p)_{\text{opt}} = \frac{1}{1 - \alpha_1 \rho(1)} \quad (2.78)$$

If an arbitrary value for α_1 is chosen, then by repeating the derivation leading to Eq (2.76), we get

$$\sigma_d^2 = \sigma_x^2 [1 - 2\alpha_1 \rho(1) + \alpha_1^2] + \alpha_1^2 \sigma_e^2 \quad (2.79)$$

$$(G_p)_{\text{opt}} = 1 / (1 - 2\alpha_1 \rho(1) + \alpha_1^2 (1 + 1/\text{SNR})) \quad (2.80)$$

The term α_1^2 / SNR represents the increase in variance of $d(n)$ due to the feedback of the error signal $e(n)$. Eq (2.80) can be written in the form

$$(G_p)_{\text{arb}} = (1 - \alpha_1^2 / \text{SNR} \rho) / (1 - 2\alpha_1 \rho(1) + \alpha_1^2) \quad (2.81)$$

for any value of α_1 (including the optimum value). To obtain the maximum gain, Eq (2.81) can be differentiated with respect to α_1 to give

$$\frac{d(G_p)}{d\alpha_1} = 0 \quad (2.82)$$

which can be solved directly for the optimum value of α_1 .

For illustrative purposes, the term $1/\text{SNR}$ is neglected in Eq (2.69). Thus, for a first order predictor, Eq (2.72) becomes $\alpha_1 = \rho(1)$, and the gain due to prediction is

$$(G_p)_{\text{opt}} = 1(1 - \rho^2(1)) \quad (2.83)$$

Thus so long as $\rho(1) \neq 0$ there will be some improvements due to prediction. It is clear that, even with the simplest predictor, it is possible to realize about a 6dB improvement in SNR. This is of course equivalent to adding an extra bit to the quantizer. However, since this bit is not actually added, the bit rate remains the same. The price paid, of course, is increased complexity in the quantization system.

Some basic principles of application of the differential quantization scheme can be summarized. First, it is clear that differential quantization can yield improvement over direct quantization. Second, the amount of improvement is dependent upon the amount of correlation. Third, a fixed predictor can not be optimum for all communication systems. These facts have led to a variety of schemes that are based upon the basic configuration of Fig (2.16). These schemes combine a variety of fixed and adaptive quantizers with a variety of fixed and adaptive predictors to achieve improved quality or lowered bit rate.

2.4 DIFFERENTIAL PULSE CODE MODULATION (DPCM)

2.4.1 PULSE CODE MODULATION (PCM)

Conceptually we may think of a PCM transmission system as shown in Fig (2.17).

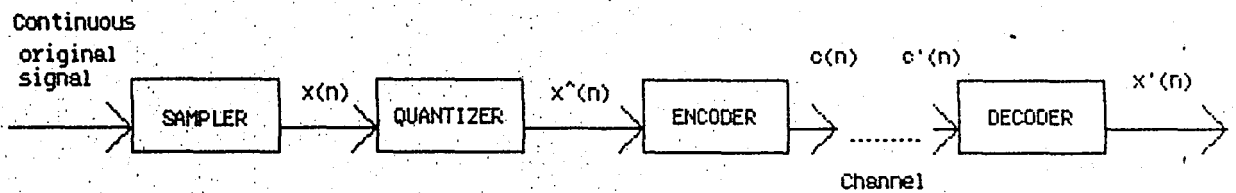


Fig.2.17 A PCM Transmission system: Coder and Decoder

If the signal is an image then the system in Fig (2.18) is used for picture transmission [58], [59].

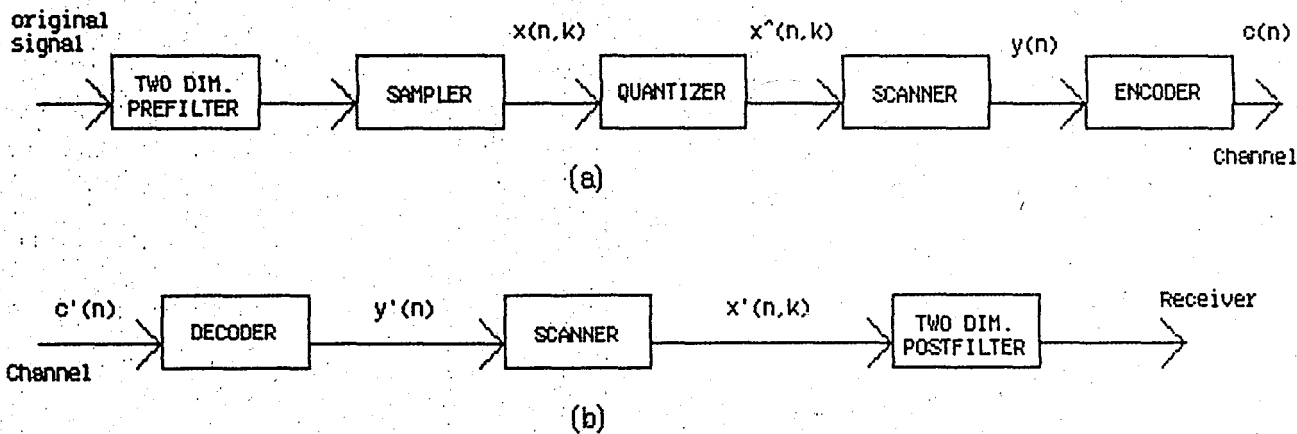


Fig.2.18 A PCM picture transmission system a)coder,b)decoder

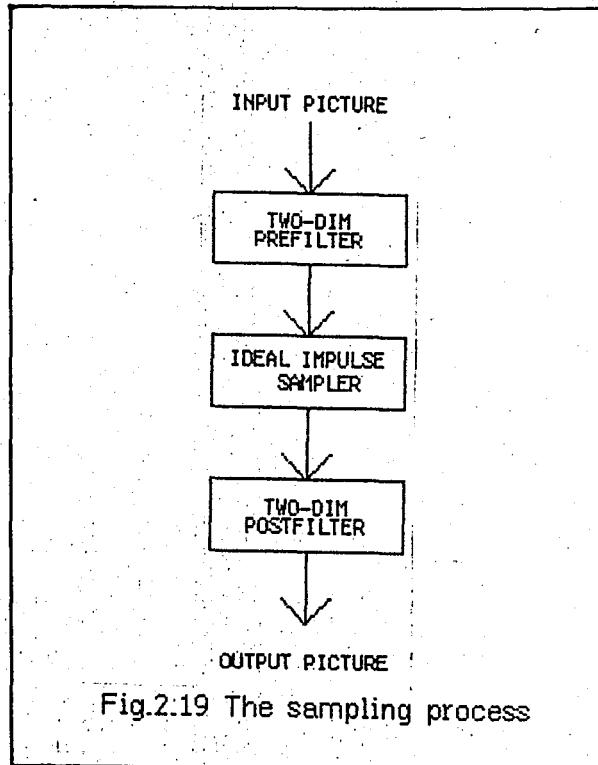
The continuous two-dimensional picture is prefiltered and then sampled in space and then quantized in brightness. A scanner transforms this

two-dimensional array of quantized samples into a one-dimensional sequence which goes into an encoder, which in turn puts out a sequence of binary numbers for transmission over the channel. At the receiver, a decoder tries to recover from the received binary sequence the original sequence of quantized samples. A scanner then transforms this reconstructed sequence into a two-dimensional array. Finally a two-dimensional post-filter is used to obtain a continuous picture. By comparing the figures given in section 2.1 and Fig (2.18) one can easily see that each of the quantization systems explained in section 2.1 is one choice of PCM transmission. When adaptive quantization is used directly on samples of the input the system is called adaptive PCM or simply APCM. If it is a picture transmission then only scanners and two-dimensional filters must be added to those systems.

SAMPLING

To concentrate on the sampling process, let us consider the simplified subsystem depicted in Fig (2.19). The basic question is, for a fixed number of samples per frame, how should one choose the prefilter, the postfilter, and the sampling pattern to optimize the output picture quality? Let the picture be sampled at a square array of points. Peterson and Middleton [60] showed that, for a fixed number of samples per frame, pre- and postfiltering with one-dimensional ideal low-pass filters (whose cut-off frequencies are to avoid aliasing) give the least mean square difference between the output and the input. Subjective tests [61] indicated that these same filters also give reconstructed pictures with the best subjective quality in the case of very low-resolution 64×64 samples per frame) systems. For higher resolution systems (256×256 samples per frame), high frequency accentuation at the postfilter seems to improve the output picture quality; however, no

extensive subjective tests have been done to substantiate this.



That the quality of the output picture depends on the shapes of the pre- and the postfilters was demonstrated by Huang and Tretiak [61]. The sampling process is basic to all PCM picture transmission systems and deserves further investigation.

QUANTIZATION

In section 2.1 the quantization process is explained in detail. Quantization noise can be reduced by putting a pre- and a postfilter around the quantizer. The human eye objects much more to noise with strong structure, such as quantization noise, than to random noise. Therefore, a smaller number of quantization levels can be tolerated, if means can be found to transform quantization noise to random noise. An example is the method of Roberts [62], who added pseudo-random noise to a picture before

quantization, and later at the receiver subtracted the same noise from the quantized picture. It can be shown that by this maneuver the quantization noise is transformed into random noise with the same rms value. This method gives acceptable pictures with only four bits per sample.

SCANNING

In the PCM picture transmission system, the transmitter scanner converts the two-dimensional digitalized picture to a one-dimensional sequence of points, and the receiver scanner reconstructs a two-dimensional picture from the received one-dimensional sequence. Usually, the scanners scan the picture line by line sequentially. However, many other scanning patterns have been suggested. The so-called pseudorandom scanning, for example can be used for secrecy transmission [63].

In pseudorandom scanning, the scanning beam hops from point to point in a seemingly random fashion. However, the transmitter and receiver scanners are synchronous, so the receiver scanner can reconstruct the picture. The coordinates of the successive scanning points are specified by a sequence of pseudorandom numbers which can be generated, for example, by a maximum-length shift register generator. Any one who does not know the particular pseudorandom sequence will not be able to reconstruct the correct picture even if he should intercept the one-dimensional signal being transmitted. Assuming the transmitter and receiver scanners are synchronous, then the appearance of the received picture will be independent of the scanning pattern, if the channel is noiseless or if it is noisy but treats each incoming bit independently. In many particular channels, however, the noise tends to occur in bursts. For such channels,

sequential scanning will yield received pictures containing errors that last over many successive picture points along the scanning direction, whereas pseudo-random scanning will randomize the noise so that the errors will scatter more or less uniformly over the entire picture.

CODING AND CHANNEL NOISE

Assume that fixed-length binary code is used, so that each of the 2^B brightness levels $(0, 1, 2, \dots, 2^B - 1)$ has a B-bit codeword. When the channel is noisy, the amount of noise in the received picture depends on the particular code one chooses to use. Two codes are often used in practice: the straight binary code, in which the codeword for each integer is just the binary representation of that integer, and the reflected binary gray code, in which the codewords for any two successive integers differ in one and only one bit. It has been proven that for transmission through a binary symmetric channel, the straight binary code yields less noise power than the gray code, with the assumption that the input brightness has a uniform distribution and that the channel error probability p is less than $1/2$.

In fact, it was shown that with these assumptions, the average noise power for a n -bit straight binary code is [64]

$$N_n = (4^n - 1)p/3 \quad (2.84)$$

and that for a n -bit reflected-binary gray code is

$$G_n = (4^n - 1)/6 = \frac{1-2p}{2} \cdot \frac{4^n - (1-2p)^n}{4 - (1-2p)} \quad (2.85)$$

so that
$$G_n - N_n = (1-2p)G_{n-1} \quad (2.86)$$

hence
$$G_n \gg N_n \quad \text{for } p \ll 1/2 \quad (2.87)$$

In most practical cases, the channel error probability p is very small. Then, the noise power is essentially due to single-bit errors in the codewords.

In summary the advantages of PCM, as compared with analog transmission methods, are as follows.

- 1 - By the use of repeaters, PCM can be employed to transmit signals over long distances without deterioration in signal-to-noise ratio.
- 2 - It lends itself to time-division multiplexing.
- 3 - It simplifies switching problems in central stations.
- 4 - It can be adapted easily to secrecy transmission.
- 5 - PCM systems are most suitable for transmitting digital data and are easily coupled with digital computers.

The disadvantages are:

- 1 - The transmitter and the receiver (or coder and decoder) for PCM are somewhat complicated.
- 2 - PCM requires more bandwidth.

2.4.2 DIFFERENTIAL PULSE CODE MODULATION (DPCM)

As mentioned in section 2.2, the statistical relationship between nearby pels and the greater sensitivity of the eye to the differences than to absolute values, both spatial and temporal, has led to many suggestions for

"difference" transmission [66]. Direct analog difference transmission has few advantages, since the bandwidth required is unchanged, the peak power is greater (through the average power is less), and the noise level is higher than in the original signal.

Any system of the form shown in Fig (2.16) could be called a differential PCM (DPCM) system. If this is a picture transmission system as mentioned earlier about PCM, the system may have two-dimensional filters, predictor and quantizer. It may have a scanner in order to transform this two-dimensional array of quantized samples into a one-dimensional sequence. The output of the scanner must go into an encoder, which in turn puts out a sequence of binary numbers for transmissions over the channel. The remaining is the same both for one-dimensional and two-dimensional transmission processes (see Fig (2.18) for PCM).

Delta modulators, as briefly discussed in section 2.2, for example, could also be called a one-bit DPCM system. Generally, however, the term differential PCM is reserved for differential quantization systems in which the quantizer has more than two levels.

In DPCM systems, what is transmitted is the quantized difference between the input signal and a second signal which is the same as that which can be derived at the receiver by integration of the received signal. Differential quantizing, since it uses the transmitted data primarily to represent edges to which the observer is more sensitive, produces better quality pictures than PCM for the same number of bits per pel.

As it is clear from Fig (2.20) which is taken from the study of Noll [65],

DPCM systems with fixed predictors can provide from 4 to 11 dB improvement over direct quantization (PCM). The greatest improvement occurs in going from no prediction to first order prediction with somewhat smaller additional gains resulting from increasing the predictor order upto 4 or 5, after which little additional gain results.

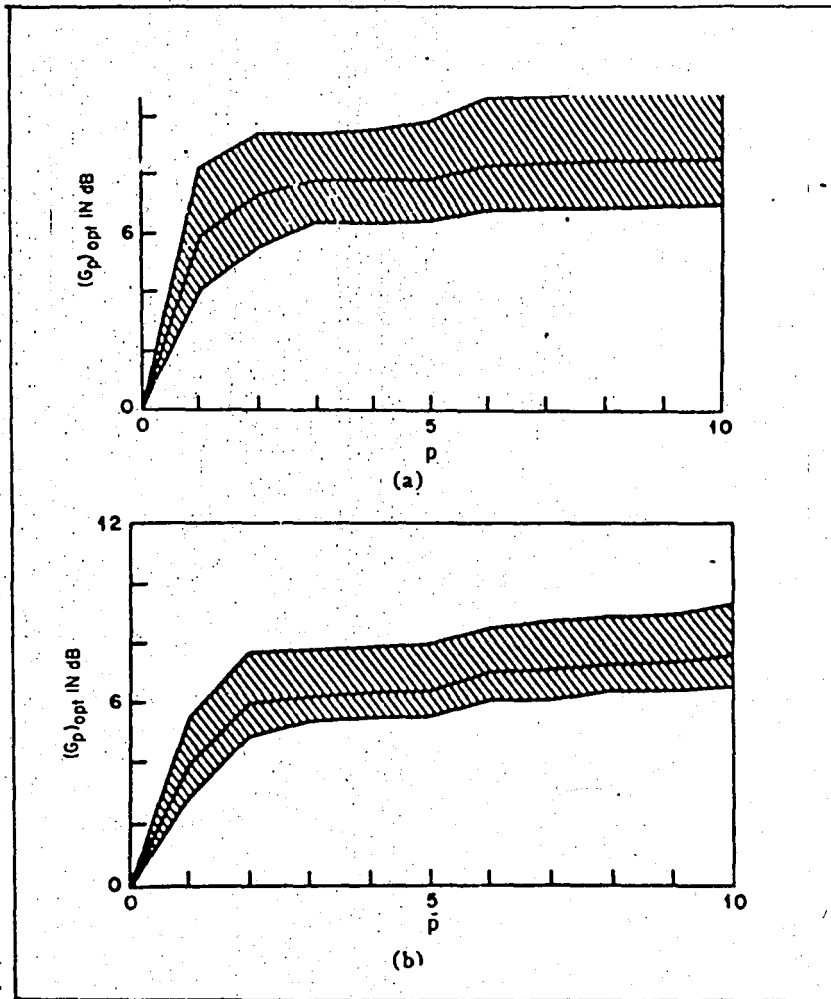


Fig 2.20 Optimal SNR gain G vs. number of predictor coefficients a) Low-pass filtered speech signal
b) Bandpass filtered speech signal

This gain in SNR implies that a DPCM system can achieve a given SNR using

one less bit than would be required when using the same quantizer directly. Thus, a reasonable estimate of the performance that can be obtained for a particular quantizer used in a differential configuration can be obtained. For example, for a differential PCM system with a uniform fixed quantizer the SNR would be approximately 6 dB greater than the SNR for a quantizer with the same number of levels acting directly on the input. The differential scheme would behave in much the same manner as the direct PCM scheme, i.e., the SNR would increase 6 dB for each bit added to the code words, and the SNR would show the same dependence upon signal level. Some examples will be given in section 2.5.7. Similarly the SNR of a logarithmic quantizer would be improved by about 6 dB by use in a differential configuration and at the same time its characteristic insensitivity to input signal level would be maintained. Fig (2.20) displays a wide variation of prediction gain with source and with bandwidth. This variation of performance with the input source and picture material, together with variations in signal level inherent in the picture transmission process, may make adaptive prediction and adaptive quantization necessary to achieve best performance over a wide range of inputs. Such systems are called adaptive DPCM systems (ADPCM).

2.4.3 DPCM WITH ADAPTIVE QUANTIZATION (DPCM-AQ)

The discussion of adaptive quantization in Sec.(2.1.2) can be applied directly to the case of DPCM. As indicated in Sec.(2.1.2), there are two basic approaches for the control of adaptive quantizers.

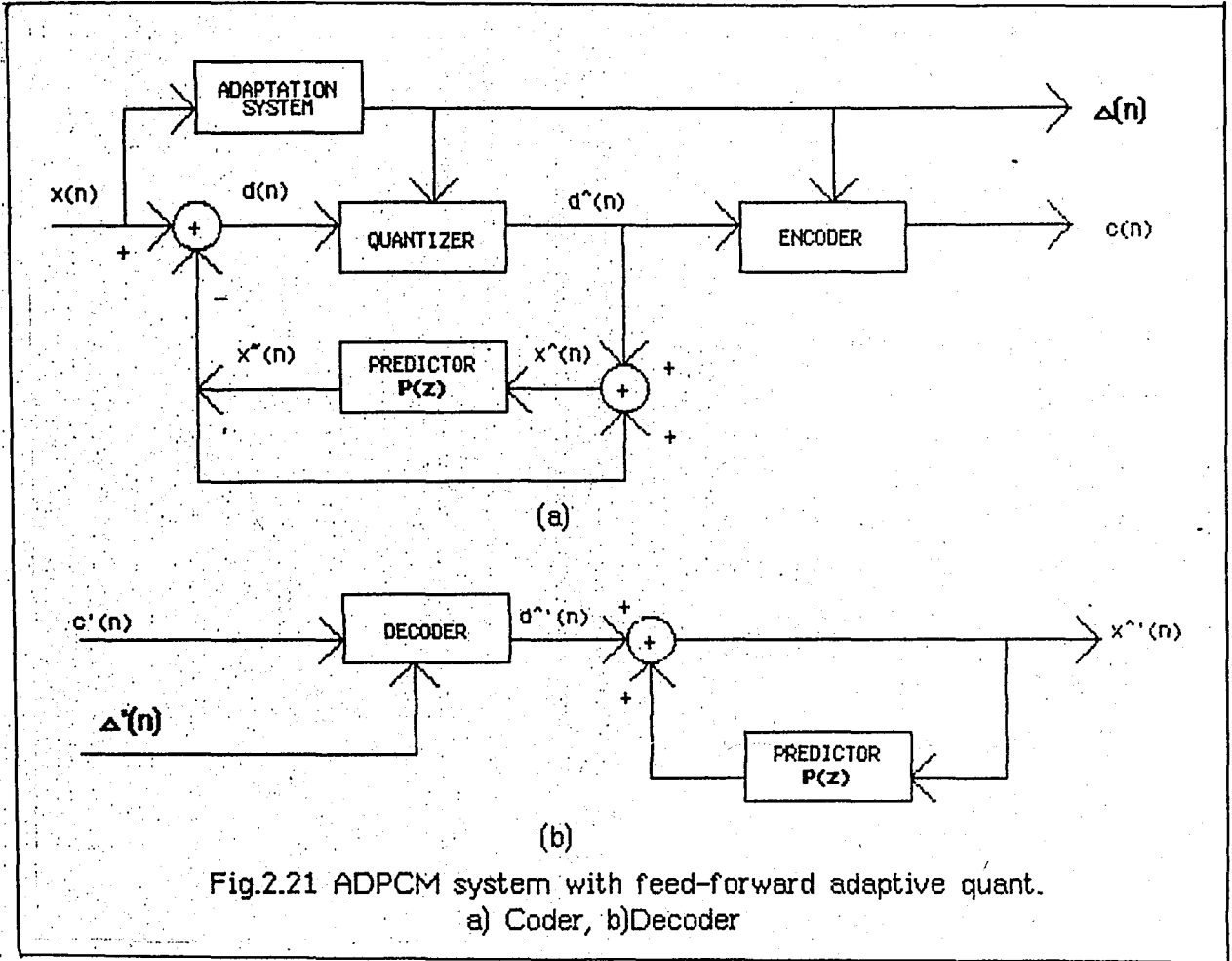


Fig.2.21 ADPCM system with feed-forward adaptive quant.
a) Coder, b) Decoder

Fig (2.21) shows how a feed-forward-type adaptive quantizer is used in an ADPCM system [65]. In schemes of this type, the quantizer step size is proportional to the variance of the input to the quantizer. However, since the difference signal $d(n)$ will be proportional to the input, it is reasonable to control the step size either from $d(n)$, or as depicted in Fig (2.21), from the input $x(n)$. Several algorithms for adjusting the step size

are given in section 2.1.2. The discussion of section 2.1.2 indicates that such adaptation procedures can provide about 5dB improvement in SNR over standard logarithmic non-adaptive PCM. This improvement coupled with the 6dB that can be obtained from the differential configuration with fixed prediction, means that ADPCM with feed-forward adaptive prediction should achieve an SNR that is 10-11dB greater than could be obtained with a fixed quantizer with the same number of levels. Simulation results will be given in section 2.5.

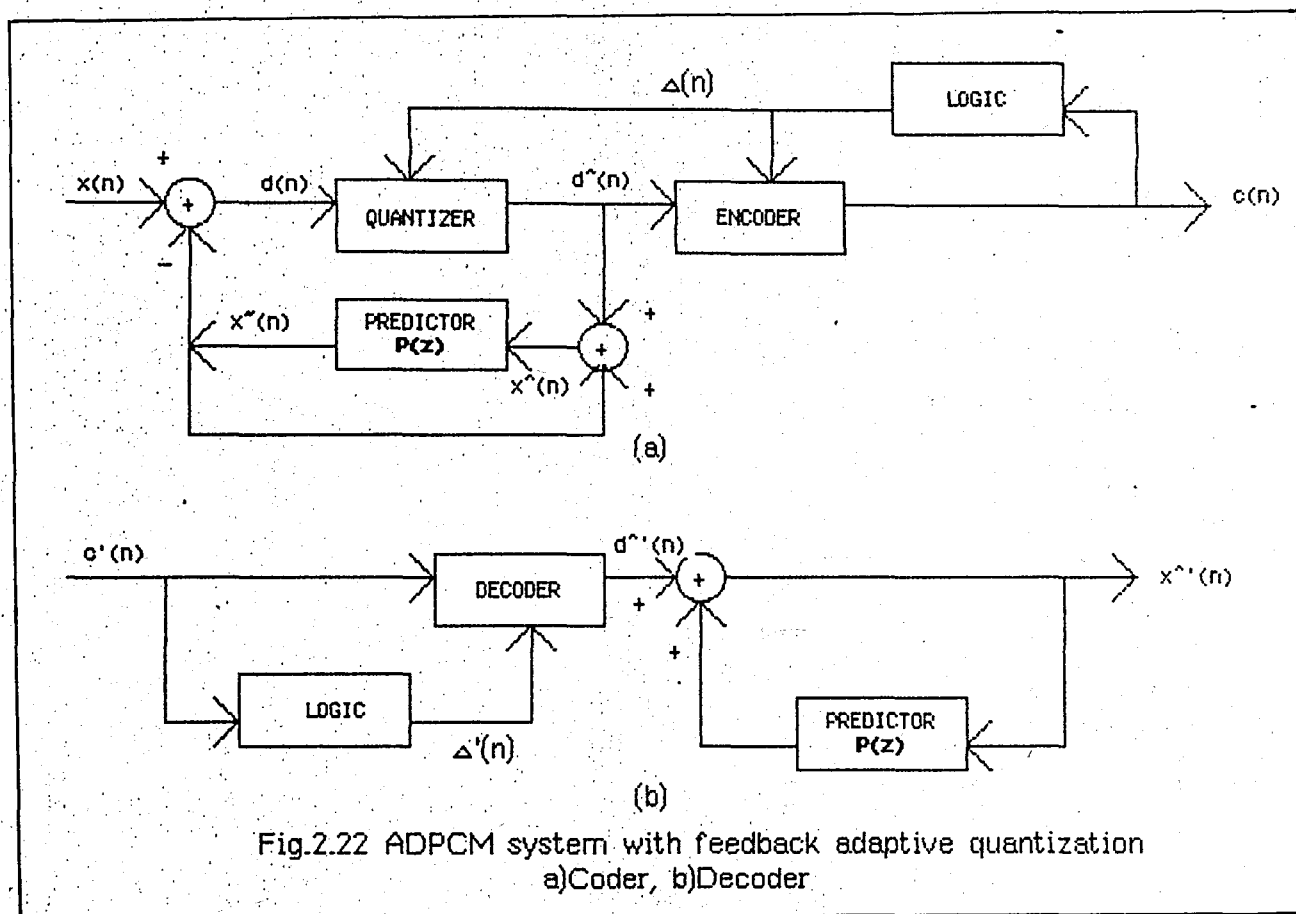


Fig (2.22) shows how a feedback-type adaptive quantizer can be used in an ADPCM system. Thus, both the feed-forward and the feedback adaptive quantizers can be expected to achieve about 10-12dB improvement over a fixed quantizer with the same number of levels.

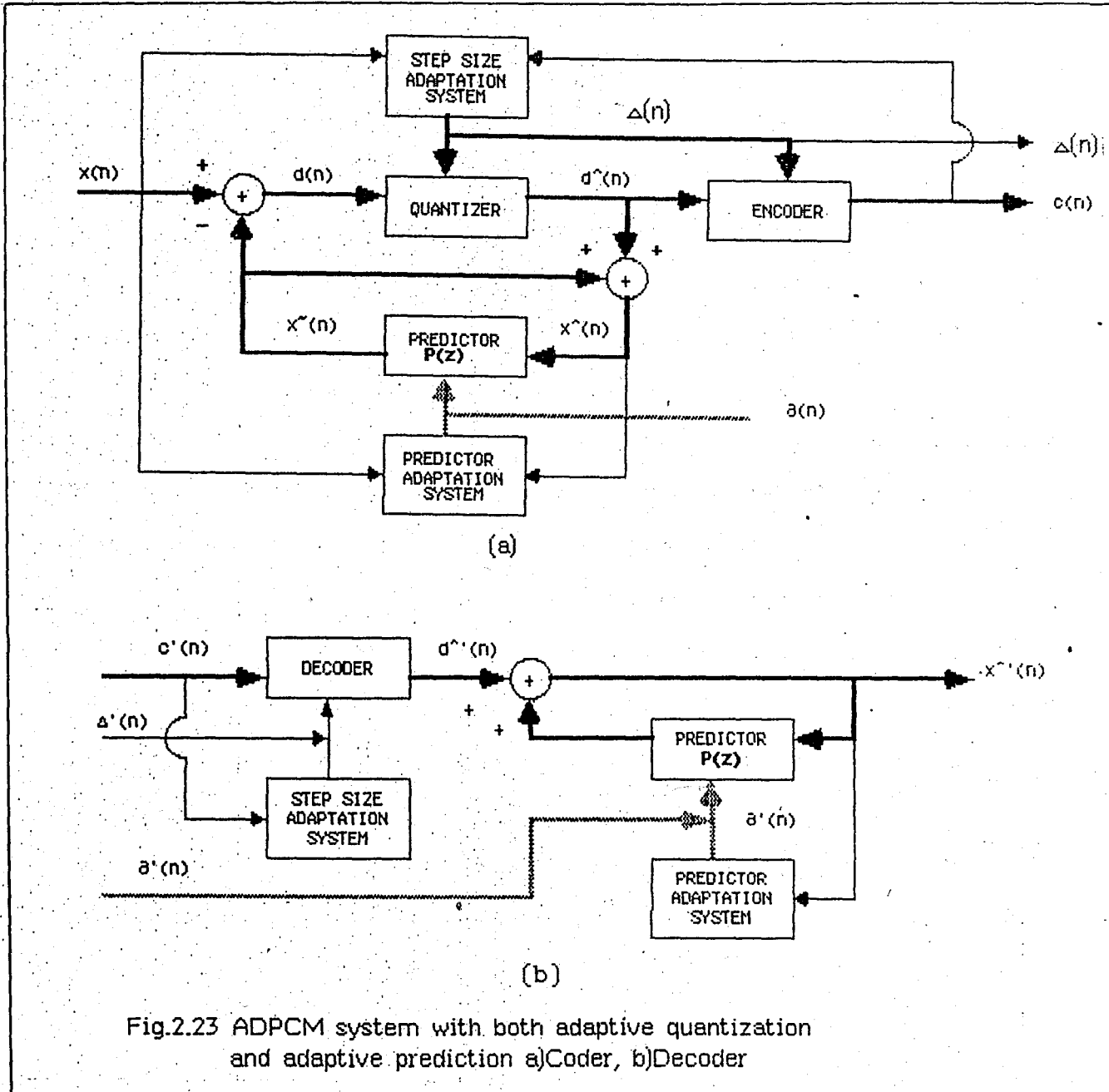
In either case the quantizer adaptation provides improved "dynamic range" as well as improved SNR. The main advantage of the feedback control is that the step size information is derived from the code word sequence, so that no additional step size information need be transmitted or stored. This however, makes the quality of the reconstructed output more sensitive to errors in transmission. With feed-forward control, the code words and the step size together serve as the representation of the signal. Although this increases the complexity of the representation, there is the possibility of transmitting the step size with error protection, thereby significantly improving the output quality for high error rate transmission.

2.4.4 DPCM WITH ADAPTIVE PREDICTION (ADPCM-APAQ)

So far, only fixed predictors have been considered and it has been found that with higher order predictors, we can expect that differential quantization will provide, about 10-12dB improvement. Furthermore the amount of improvement is a function of the input. In order to effectively cope with the non-stationarity of the image transmission process, it is natural to consider adapting the predictor as well as the quantizer to match the temporal variation of the image samples. A general adaptive DPCM system with both adaptive quantization and adaptive prediction is depicted in Fig (2.23). The dotted lines indicate that both the quantizer adaptation and the predictor adaptation algorithms can be either of the feed-forward or the feedback type. If feed-forward control is used for the quantizer or the predictor, then $\Delta(n)$ or the predictor coefficients, $\alpha(n) = \{\alpha_k(n)\}$, (or both) are also required in addition to the code words, $c(n)$, to complete the representation of the image samples.

The predictor coefficients are assumed to be time dependent so that the predicted value is

$$\tilde{x}(n) = \sum_{k=1}^N \alpha_k(n) \hat{x}(n-k) \quad (2.88)$$



In adapting the predictor coefficients $\alpha_k(n)$ it is common to assume that the properties of the image samples remain fixed over short time intervals. The predictor coefficients are therefore chosen to minimize the average squared prediction error over a short time interval. For feed-forward control, the

predictor adaptation is based upon measurements on the input signal. Using the same type of manipulations that were used to derive Eq (2.67) and (2.69) and neglecting the effect of quantization errors, it can be shown that the optimum predictor coefficients satisfy the equations,

$$R_n(j) = \sum_{k=1}^N \alpha_k(n) R_n(j-k) \quad j=1,2,\dots,N \quad (2.89)$$

where $R_n(j)$ is the short-time autocorrelation function

$$R_n(j) = \sum_{m=-\infty}^{\infty} x(m)w(n-m)x(j+m)w(n-m-j) \quad 0 \leq j \leq p \quad (2.90)$$

and $w(n-m)$ is a window function that is positioned at sample n of the input sequence. Since the parameters of the image vary rather slowly, it is reasonable to adjust the predictor parameters $\alpha(n)$ infrequently. As defined by Eq (2.90), the computation of the correlation estimates required in Eq (2.89) would require the accumulation of N samples of $x(n)$ in a buffer before computing $R_n(j)$. The set of coefficients $\alpha(n)$ satisfying Eq (2.89) are used in the configuration of Fig (2.23a) to quantize the input during the interval of N samples beginning at sample n . Thus, to reconstruct the input from the quantizer code words we also need the predictor coefficients (and possibly the quantizer step size) as depicted in Fig (2.23b). The details of computing the time-varying predictor parameters is beyond the purpose of this study.

In order to quantitatively express the benefits of adaptive prediction, Noll [65] has examined the dependence of the prediction gain, G_p , upon predictor order for both fixed and adaptive predictors.

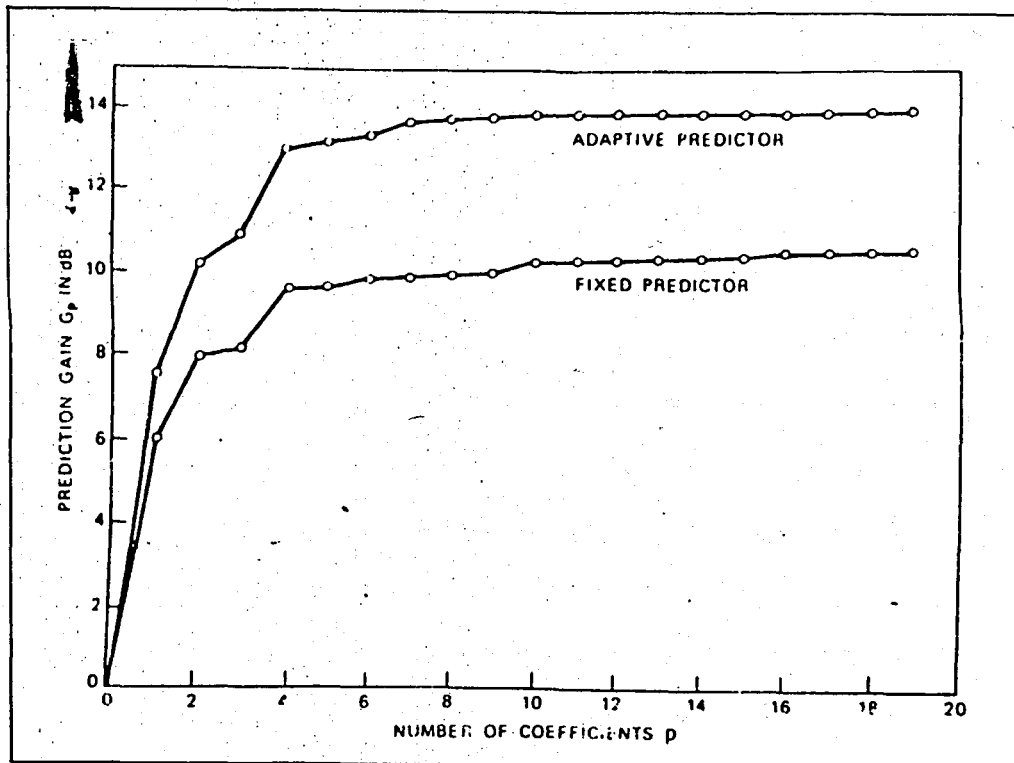


Fig 2.24 Predictor gains vs. number of predictor coefficients, which is taken from the study of Noll [65]

Fig (2.24) shows the quantity G_p , as a function of predictor order, p , for both fixed and adaptive prediction. The lower curve, obtained by computing a long term estimate of the autocorrelation for a given sample and solving for the set of predictor coefficients satisfying Eq (2.69), shows a maximum gain of about 10.5dB. The upper curve was obtained by finding the value of window length, L , and the predictor coefficients $\alpha(n)$ that maximized G_p across the entire utterance for a fixed value of N (number of predictor coefficients). That maximum value is plotted for each value of N . In this case, the maximum gain is about 14dB. Thus, Noll [65] suggests that reasonable upper bounds on the performance of DPCM systems with fixed and adaptive prediction are 10.5 and 14dB, respectively for speech signals.

2.5 RESULTS OF THE COMPUTER SIMULATION

In this section the simulation results of PCM and DPCM techniques are examined. A package program is developed which consists of different types of quantizers, multilevel predictors and different types of signals. By using menus on the screen one can choose the type of signal or quantizer or decide to use a predictor or not. The structure of the program is designed such that it can serve for both two-dimensional or one-dimensional signal processing. But because of the storage capacity of the computer, test images could not be generated and the program could not be tested on images. Instead, one-dimensional AR model waveforms have been generated and the proposed picture coding techniques were tested on this models. The program has been developed for just finding the performances of several PCM and DPCM techniques but not for the comparison and evaluation of those. At the beginning, the structure of the program will be explained in detail and then some outputs will be given.

2.5.1 A PACKAGE PROGRAM FOR QUANTIZATION, PCM AND DPCM TECHNIQUES

This program has been developed by using the APPLE IIe microcomputer, which has 64 kB main memory and 140 kB diskette capacity. The program is written in the APPLESOFT BASIC programming language. The memory allocation of Apple does not allow the use of graphic mode, large variable strings and long programs at the same time. That's why, if we want to see the graphic or the spectrum of the input or the input itself, we have very limited area for variables. Then if the graphics are required, one can work only with 64 samples of data at a time. A subprogram has been developed to solve this problem. A long string of input can be given at the beginning. This

subprogram divides the whole signal into pieces in such a way that each piece consists of 64 samples at most. In order to use the graphics mode of the computer, it has been avoided to use long programs; instead it has been divided into many little programs which are CHAINED together without losing the variables which must be shared by all programs.

As mentioned above test images could not be generated because of limitations of the capacity and the package program was tested on some one-dimensional waveforms such as AR model, sine, random sine or square waveforms. As explained in Chapter 1, autoregressive sequences are one of the good representations of images. Thus, all analysis and performance evaluations were performed on autoregressive sequences. The examples of input waveforms are shown in Fig (2.25). In section 1.5 detailed information about AR sequences has been given.

The names and the functions of the programs are the following:

MAIN.: The package begins with this program and after choosing the type of signal, the respective signal generation program is called.

KARPROG, SINPROG, RASPROG and GAUPROG are the names of the respective waveform generation programs. As mentioned earlier, the number of samples can be much more than 64. The generated samples are stored in a textfile and recalled as blocks of required size (maximum 64).

QUANPROG: After generation of input samples, signal generation programs are CHAINED to a program called QUANPROG. here, the type of quantization and coding techniques, number of quantizer levels and types of quantizers are chosen.

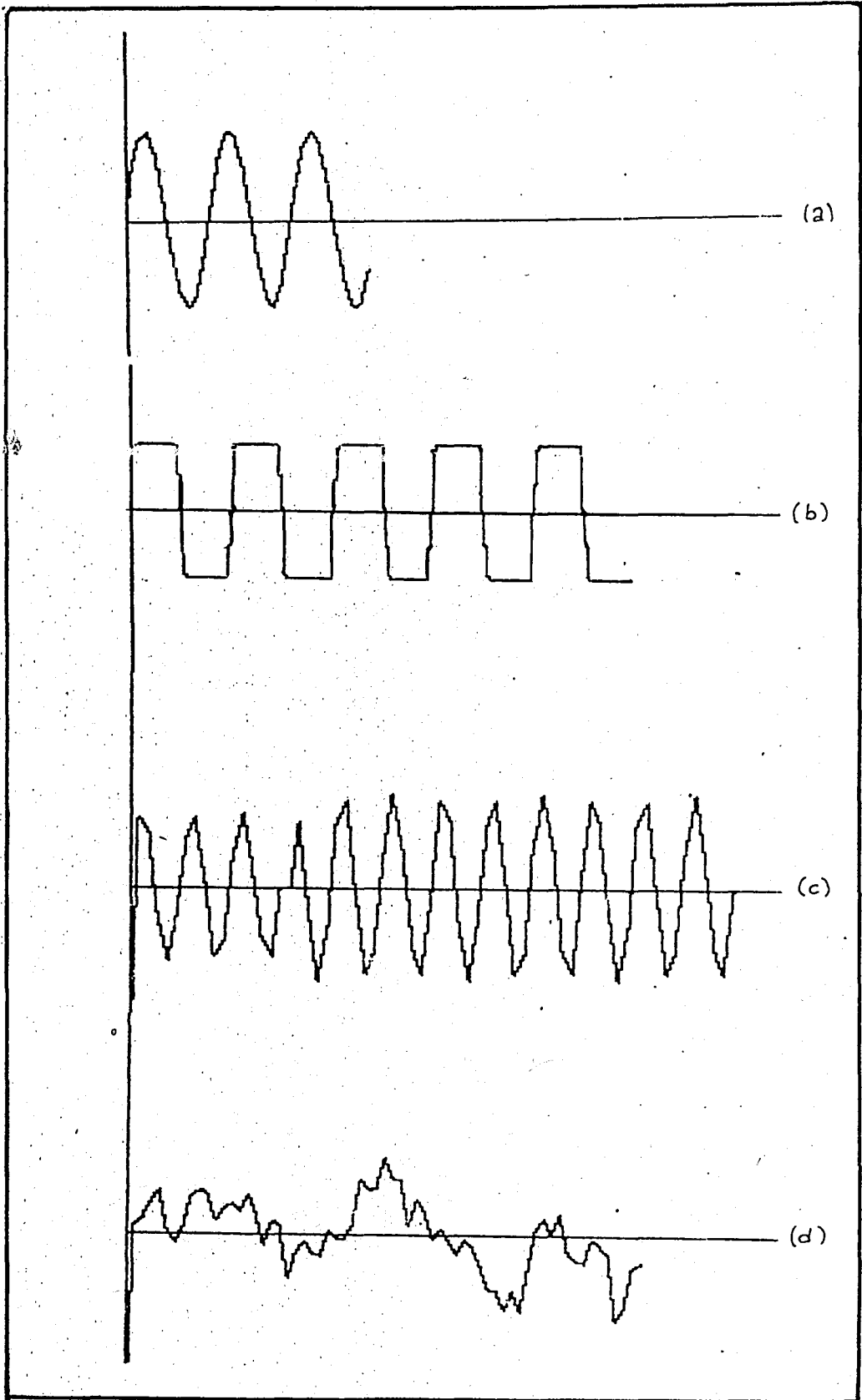


Fig 2.25 Four types of input sequences used in the program each of which consists of 64 samples
a)Sine-wave b)Square-wave c)Random Sine-wave d)AR Sequence

The quantization and coding techniques in the package are classified as:

- 1 - Uniform quantization with signal variance adaptation
- 2 - Nonuniform optimum quantization with signal variance adaptation
- 3 - Differential quantization with a summer instead of a predictor (See Sec.2.5.4)
- 4 - Logarithmic quantization
- 5 - Feed-forward adaptive quantization
- 6 - Differential pulse code modulation (DPCM)

The program has been constructed in such a way that each quantization technique can choose several types of quantizers or predictors.

Choices for quantizer types in the simulation program are as follows:

- 1 - Quantizers with uniform midriser values
- 2 - Quantizers with uniform midtread values
- 3 - Quantizers with Gamma densities (optimal quantizer)
- 4 - Quantizers with Laplace densities (optimal quantizer)
- 5 - Quantizers with Gauss densities (optimal quantizer)

The quantizer levels can vary from 2 to 32. This program is chained to another program without losing the variables according to the type of quantization.

Q.SNRPROG, DIFADAPQU, FFADAPQU, DPCM: If optimum, logarithmic or uniform quantization is chosen, then Q.SNRPROG is chained to QUANPROG. DIFADAPQU contains the algorithm of a type of differential quantization which has a summer, which finds the cumulative error. the details of the algorithm will be given later in this chapter. FFADAPQU performs feed-forward adaptive

quantization. DPCM has the algorithm of DPCM with usual predictor. DPCM can find upto the 6th coefficient of the predictor.

OZILINTI: In order to get correct decisions on error waveform, autocorrelation coefficients of input and error might be required. This program can be chosen optionally.

FFTPROG: Finds the spectrum of input, error, quantized signal or autocorrelation functions of input and error. In order to get smooth results the same quantization technique is applied on the same type of waveform several times, and each time the results are saved on a textfile. At the end the average values of the spectrums are calculated. The listings of the programs will be given in Appendix A. Fig (2.26) shows an example

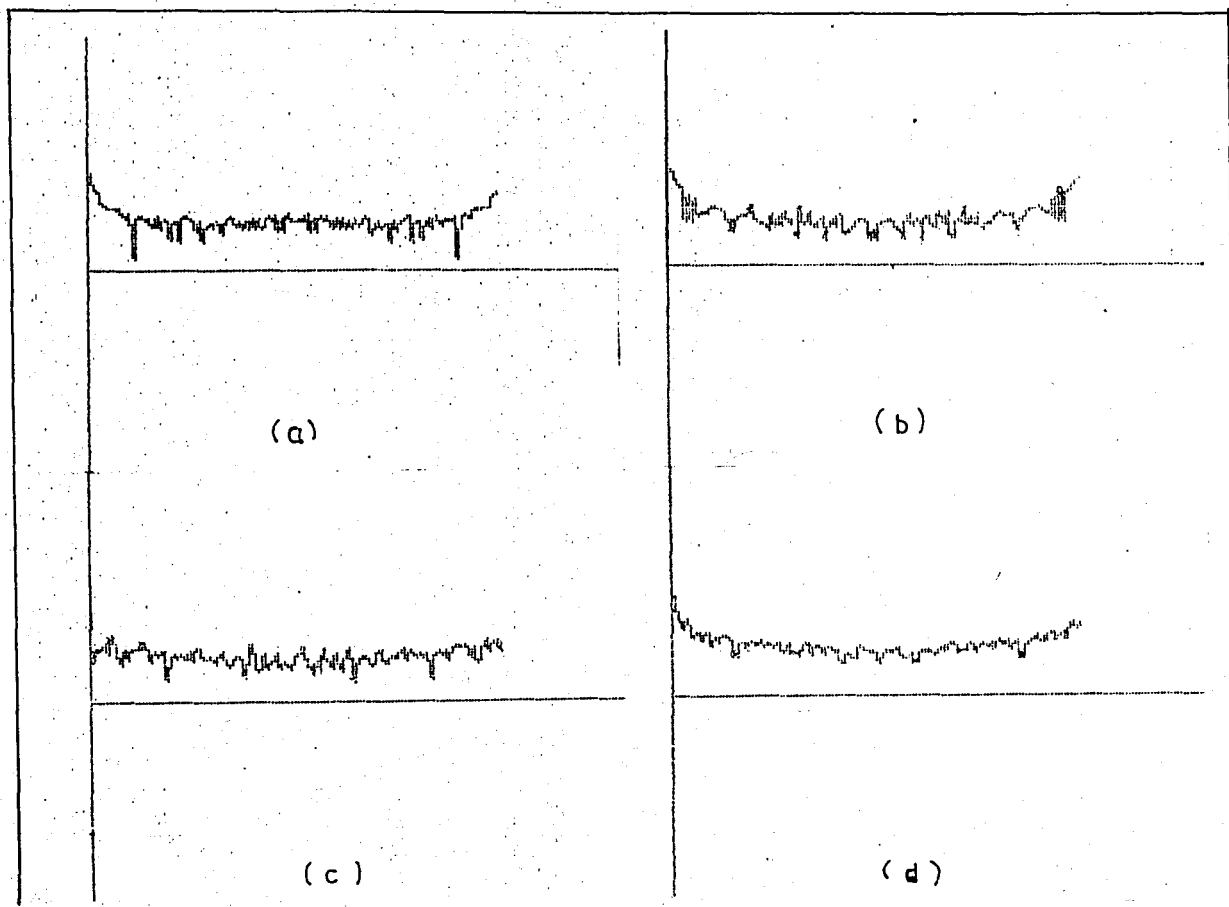


Fig 2.26 Spectra of three quantized signals individually and their averages in a), b), c) and d), respectively

2.5.2 UNIFORM QUANTIZATION WITH SIGNAL VARIANCE ADAPTATION

The algorithm of this technique is the same as the algorithm which has been explained in section 2.1.1. Quantizer levels are calculated by the formula;

$$2x_{\max} = \Delta 2^B \quad (2.91)$$

where $x_{\max} = 4\sigma_x$ assuming $\sigma_x = 1$ (2.92)

Two types of quantizers can be chosen as uniform midriser or uniform midtread values. Quantization levels and stepsizes are shown in Fig (2.3) and Table (2.1) where $\sigma_x = 1$. All tests are done on AR model sequences and the variance of the sequences are calculated. The quantizer stepsizes are then multiplied with this variance, in order to adapt the quantizer to the signal variance. The outputs are taken for 2-, 3-, 4- and 5-bit quantizers. The signal to noise ratio for all outputs are calculated. Fig (2.27) shows the result as SNR vs. number of bits used in quantization. As expected, each bit in the code word contributes 5-7db to the signal to noise ratio.

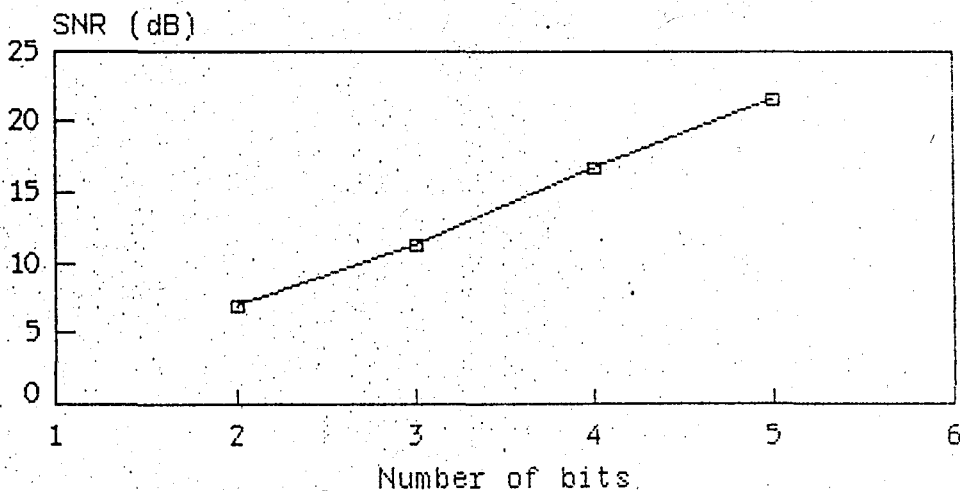


Fig 2.27 SNR vs. quantizer stepsizes for uniform quantization with midriser values

In Fig (2.28) AR model sequence and the quantized waveform using 3-bit quantizer with midriser values are shown.

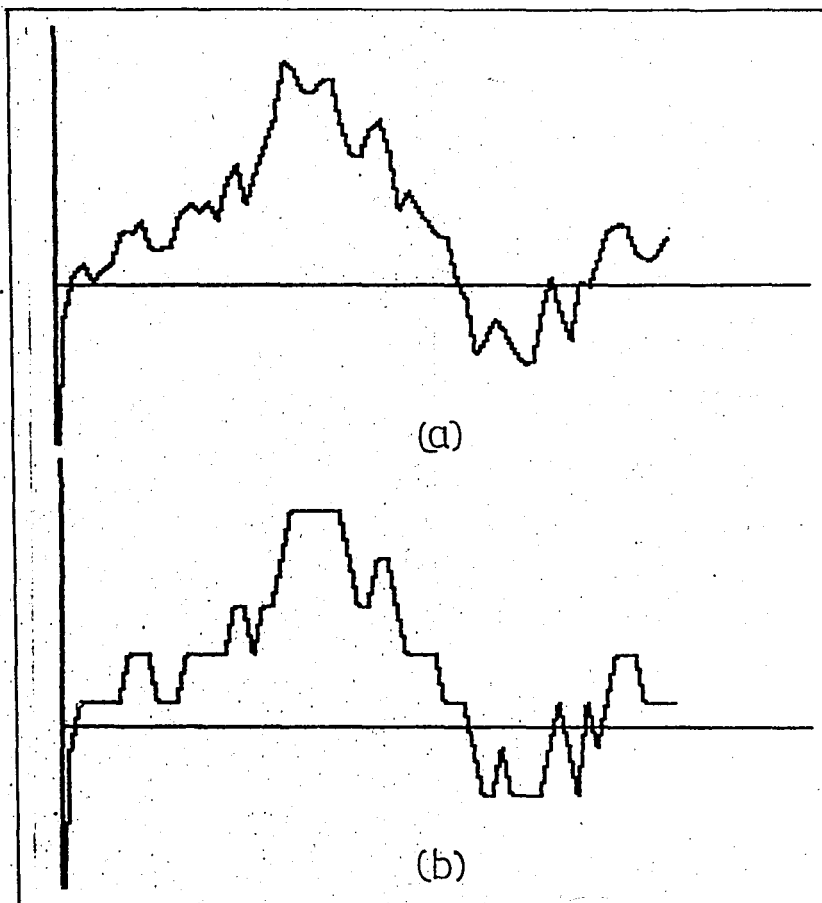


Fig 2.28 Uniform quantization with 3-bit uniform midriser values a) original sequence b) quantized sequence

Table (2.5) shows the numerical values of original sequence, quantized sequence and the quantization error.

-8.92055506	-11.085461	2.16490596
8.92055506	3.986995	4.93356005
8.92055506	7.42654005	1.494015
8.92055506	.692417845	8.22813721
8.92055506	5.69473617	3.22581889
8.92055506	8.31073588	.609819174
26.7616652	19.0337001	7.72796511
26.7616652	18.7343386	8.02732657
26.7616652	23.8826766	2.87898855
8.92055506	12.9852335	-4.0646784
8.92055506	13.4542391	-4.53368408
8.92055506	13.9327125	-5.01215744
26.7616652	25.1561224	1.60554273
26.7616652	29.9233732	-3.16170801
26.7616652	26.8993281	-1.137662925
26.7616652	30.4152624	-3.65359718
26.7616652	23.5951349	3.16653032
44.6027753	37.0518421	7.55093322
44.6027753	44.5754203	.0273549557
26.7616652	29.307357	-2.54569179
44.6027753	41.7063302	2.89644508
44.6027753	51.5632128	-6.96043746
62.4438854	61.1167958	1.32708962
80.2849955	83.215813	-2.93081749
80.2849955	79.7233809	.561614573
80.2849955	71.4237629	8.86123261
80.2849955	71.6846523	8.60034323
80.2849955	76.362395	3.92260054
80.2849955	75.1164471	5.16854841
62.4438854	58.9782724	3.46561299
44.6027753	47.4337443	-2.83096898
44.6027753	48.0509672	-3.44819194
62.4438854	58.2236434	4.22024202
62.4438854	61.7161116	.727773756
44.6027753	48.066164	-3.46338876
26.7616652	27.1495916	-3.387926385
26.7616652	34.648726	-7.88706081
26.7616652	27.8207233	-1.0590581
26.7616652	22.3020987	4.4595665
8.92055506	17.3732828	-8.45272777
8.92055506	17.2803016	-8.35974657
8.92055506	1.16822306	7.752332
-8.92055506	-6.75839463	-2.16216043
-26.7616652	-26.2981267	-4.463538475
-26.7616652	-20.6467806	-6.11488457
-8.92055506	-13.6382651	4.71771006
-26.7616652	-18.610363	-8.15130222
-26.7616652	-25.6235242	-1.13814098
-26.7616652	-30.7584914	3.9968262
-26.7616652	-28.6287382	1.86707299
-8.92055506	-10.1981113	1.27755621
8.92055506	2.77184754	6.14870752
-8.92055506	-11.8103456	2.88979053
-26.7616652	-21.8002898	-4.96137539
8.92055506	.307671398	8.61288367
-8.92055506	-1.79998509	-7.12056997
8.92055506	9.23455772	-.314002667
26.7616652	20.9796945	5.78197069
26.7616652	22.1704024	4.59126273
26.7616652	20.5754134	6.1862518
8.92055506	10.3906544	-1.47009933
8.92055506	8.45769168	.462863382
8.92055506	11.937767	-3.01721195
8.92055506	17.7492316	-8.82867659

(a)

(b)

(c)

Table 2.5 Numerical outputs of the 3-bit uniform quantization
a) Quantized signal b) Original signal c) Quantization error

2.5.3 NONUNIFORM OPTIMUM QUANTIZATION WITH SIGNAL VARIANCE ADAPTATION

As mentioned in section 2.1.1, in cases where the signal variance is known, it is possible to choose the quantizer levels so as to minimize the quantization error. Optimum quantizer parameters which are used in the simulation are given in table (2.2) for Laplace, Gamma and Gaussian probability density functions. As in the case of uniform quantization the values in the table are calculated assuming that the signal variance is 1. AR model sequence is used for all tests and the variance of the sequence is calculated in order to adapt the quantization stepsizes to this variance. The simulations are done using three different types of quantizer values. Table (2.6) gives the SNR values of three different types of 2-bit and 4-bit quantizers

QUANTIZER	2-BIT SNR (dB)	4-BIT SNR (dB)
Gaussian	9.82	20.80
Laplace	7.81	18.82
Gamma	5.82	17.63

Table 2.6 SNR values for 3 different type of 2-bit & 3-bit Optimum Quantizers

As one can see from table (2.6) SNR is closely dependent on the signal variance. The AR model sequence with gaussian distribution gives the best results. Fig (2.29) gives the results as SNR vs. the number of quantizer stepsizes by using gaussian distribution values.

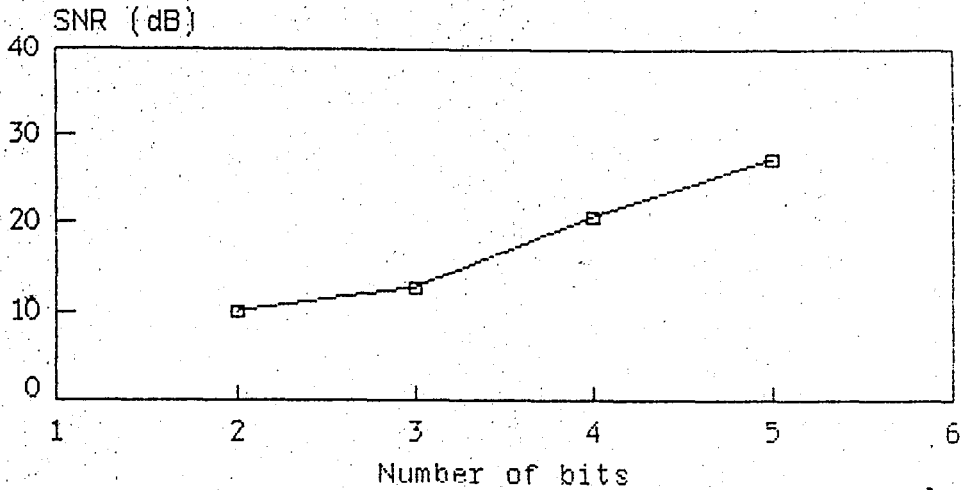


Fig 2.29 SNR vs. the number of bits for optimum quantization with gaussian densities

In Fig (2.30) AR model sequence and quantized waveform using 3-bit optimum quantizer with gaussian densities is shown. The numerical values are listed in Table (2.7), as the original sequence, the quantized sequence and the quantization error.

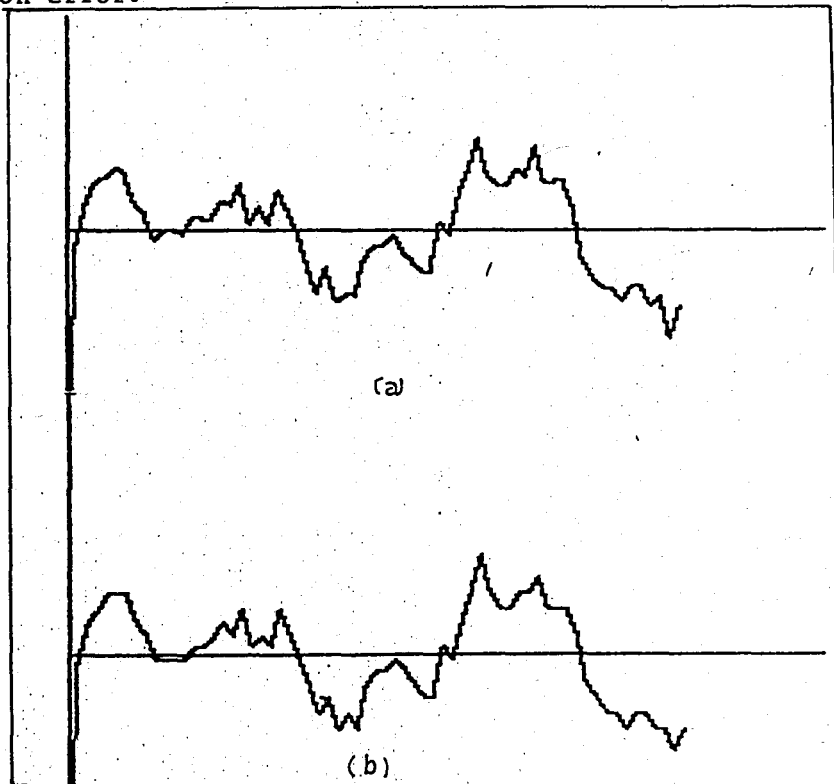


Fig 2.30 Optimum quantization using 3-bit optimum quantizer with gaussian densities a)Original signal b)Quantized signal

-2.25421947	-4.23566588	1.98144641
11.5704859	10.9881664	.582319438
16.5896464	18.376632	-1.78698556
22.1195285	20.559106	1.56042251
22.1195285	23.3584897	-1.2359612
22.1195285	20.6877105	1.43181802
11.5704859	9.9057652	1.66472067
6.83310277	5.39076426	1.44233851
-2.25421947	-4.49254574	2.23832627
-2.25421947	-.982525193	-1.27169428
-2.25421947	-1.27301328	-.981206193
-2.25421947	-2.9388102	.684590728
2.25421947	4.02964543	-1.77542596
2.25421947	3.61687023	-1.36265076
6.83310277	4.57861038	2.25449239
11.5704859	10.7292623	.841223557
6.83310277	8.92709037	-2.0939876
16.5896464	17.0619604	-.472313993
2.25421947	1.99505849	.259160982
6.83310277	8.66964447	-1.8365417
2.25421947	1.96793795	.286281518
16.5896464	14.3969127	2.19273367
6.83310277	6.22884205	.604260717
-2.25421947	-1.17304633	-1.08117314
-11.5704859	-13.5541967	1.98371088
-22.1195285	-24.0902994	1.9707709
-16.5896464	-14.1172403	-2.47240614
-28.494743	-27.577541	-.917201966
-22.1195285	-24.5804439	2.46091536
-28.494743	-25.5463371	-2.9484059
-11.5704859	-11.0545575	-.515928414
-6.83310277	-6.83029795	-2.80482322E-03
-6.83310277	-5.3487488	-1.48435397
-2.25421947	-2.71345257	.459233098
-6.83310277	-8.59670702	1.76360425
-11.5704859	-12.8616444	1.29115852
-16.5896464	-16.9493518	.359705359
-16.5896464	-16.1559856	-.433660768
2.25421947	2.41660757	-.162388104
-2.25421947	-2.76399195	.50977248
11.5704859	12.2292327	-.658746842
22.1195285	22.2889063	-.169377767
36.4373444	34.9569628	1.48038156
22.1195285	20.0284598	2.09106874
16.5896464	16.523792	.0658543855
16.5896464	16.1565853	.433061145
22.1195285	22.0253181	.0942104682
22.1195285	19.7496876	2.36984092
28.494743	31.4008813	-2.90613828
16.5896464	17.0170933	-.42744685
16.5896464	18.4473921	-1.85774573
16.5896464	17.2986646	-.709018178
6.83310277	7.98309976	-1.14999699
-11.5704859	-12.1967261	.626240183
-16.5896464	-18.7353184	2.14567199
-22.1195285	-22.7035625	.584033944
-22.1195285	-22.552017	.432488449
-28.494743	-27.5724289	-.922314055
-22.1195285	-21.2822886	-.837239981
-22.1195285	-21.5067347	-.61279387
-28.494743	-29.5738026	1.07905966
-28.494743	-25.3383382	-3.15640475
-36.4373444	-41.8102241	5.37287972
-28.494743	-29.0409968	.546253838

(a)

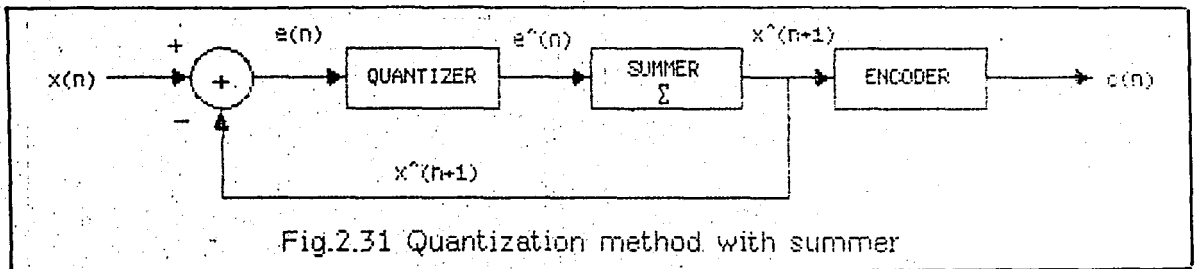
(b)

(c)

Table 2.7 Numerical output of the 3-bit optimum quantizer with gaussian densities a) Quantized sequence b) Original sequence c) Quantization error

2.5.4 DIFFERENTIAL QUANTIZATION WITH A SUMMER

An alternative technique of quantization can be obtained by adding quantization errors. This quantization method must contain a summer in addition to the quantizer. Fig (2.31) shows the block diagram of the quantization method



Here, the error is quantized and then added to the preceding errors in order to find the quantized value of the input. The formulation of the method is as follows:

$$\hat{x}(n+1) = \sum_{i=1}^n e(i) \quad \text{where } 1 \leq n \leq N \quad (2.93)$$

$$\text{and} \quad e(i) = x(i) - \hat{x}(i+1) \quad (2.94)$$

Five different types of quantizers can be chosen. Quantizer stepsizes are adapted to the error variance according to the formula:

$$\hat{x}(n) = \sum_{i=1}^M \alpha(i) \cdot \hat{x}(n-i) \quad (2.95)$$

where $\hat{x}(n)$ is the expected value and the $\alpha(i)$'s are the predictor

coefficients. For $i=1$ then,

$$\hat{x}(n) = \alpha(1) \hat{x}(n-1) \quad (2.96)$$

and one can approximate the above equation with the original values

$$x(n) = \alpha(1) x(n-1) \quad (2.97)$$

Then the error will be

$$e(n) = x(n) - \alpha(1) x(n-1) \quad \text{where } 1 \leq n \leq N \quad (2.98)$$

$$\text{and } \alpha(1) = R(1)/R(0) \quad (2.99)$$

where $R(1)$ and $R(0)$ are the autocorrelation coefficients and are found from:

$$R(0) = 1/N \sum_{n=1}^N x(n)^2 \quad (2.100)$$

$$R(1) = 1/(N-1) \sum_{n=1}^{N-1} x(n) x(n+1) \quad (2.101)$$

Then the error variance can be found by the following formula:

$$\sigma^2 = 1/(N-1) \sum_{n=1}^N e(n)^2 \quad (2.102)$$

The disadvantage of this technique is that the performance criterion, SNR is closely dependent on the signal variance. This can be seen from the outputs shown in Fig (2.32). An example of the original signal and the quantized sequence using differential quantization with summer and a 3-bit uniform midriser quantizer is shown in the figure.

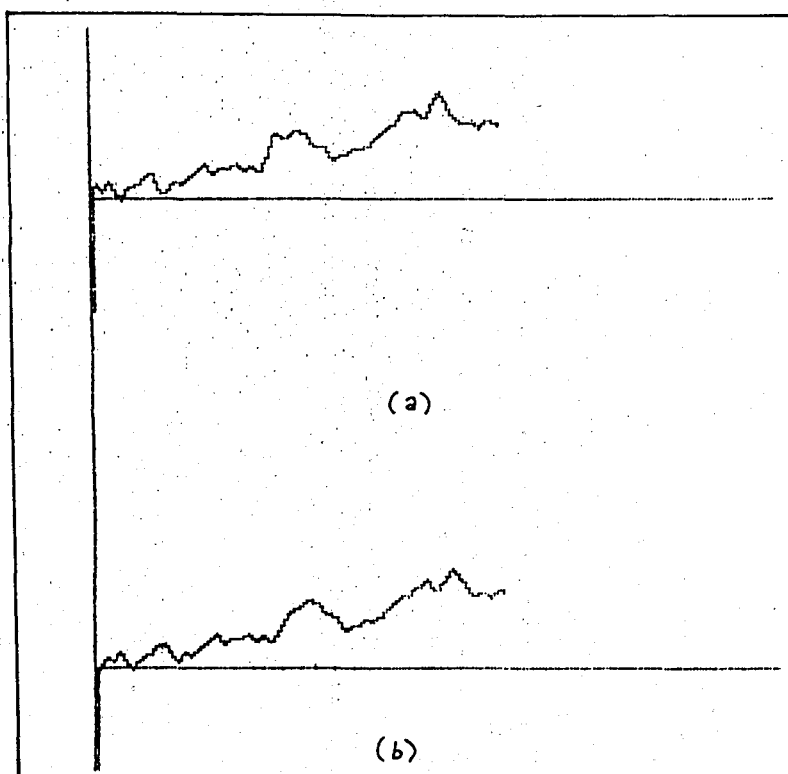


Fig 2.32 Differential quantization with summer using 3-bit uniform midriser quantizer a) original signal b) quantized sequence, $x(n+1)$.

Table (2.8) shows the values of $x(n)$, $\hat{x}(n)$, $e(n)$. It can be seen from Fig (2.32) and Table (2.8) that the first samples of the original signal are missing, but the error between input and output is decreasing while the number of samples is increasing. The value of any one input sample let say n , matches up the $(n+1)$ th value of the output sample. The performance criterion, SNR is relatively better than other quantization techniques. Fig (2.33) shows the SNR value vs. the number of quantization stepsizes. But DPCM with a predictor is a more reliable method, because it is less dependent on the signal variance.

4.81290478	0	1.06290478
1.89932313	3.75	-.10067687
5.31237873	2	.0623787306
1.32638222	5.25	-.17361778
-1.4739646	1.5	-.223964601
3.84953364	-1.25	1.34953364
4.16573442	2.5	-.0842655785
5.32529826	4.25	-.174701741
8.6187932	5.5	-.131206803
8.66511722	8.75	.165117219
1.90382203	8.5	-2.84617797
1.54731673	4.75	.0473167291
5.97469997	1.5	.72469997
4.16089677	5.25	.150896771
6.50488156	4	-.245118443
8.38657098	6.75	-.113429017
10.4592687	8.5	.209268697
12.6189783	10.25	.118978299
8.42489337	12.5	-.325106636
10.2950483	8.75	-.204951696
10.9237542	10.5	.1737542
11.3125493	10.75	-.1874507
12.6450591	11.5	-.104940899
9.90937394	12.75	-.090626061
11.9793363	10	.229336295
9.48287271	11.75	-.0171272978
10.7622502	9.5	.0122502036
15.7284275	10.75	1.2284275
23.0696705	14.5	4.8196705
21.7475631	18.25	.217563108
22.3638416	21.5	.113841601
24.2704586	22.25	-.229541399
23.5020794	24.5	-.24792061
23.3081416	23.75	-.191858396
18.8878625	23.5	-.862137496
18.6916329	19.75	.191632897
17.551596	18.5	-.198403999
13.8736503	17.75	-.126349695
15.6609925	14	-.0890075043
15.522347	15.75	.0223470032
17.2030273	15.5	-.0469727069
17.0205803	17.25	.0205803066
18.5221522	17	-.2278478
19.3393308	18.75	-.1606692
22.1014807	19.5	-.148519307
24.2573632	22.25	-.242636792
26.058911	24.5	-.191088997
26.1410339	26.25	.141033903
31.0560074	26	1.30600741
31.1082493	29.75	.108249292
32.0083784	31	-.241621614
29.0922039	32.25	.0922038928
29.1430945	29	-.106905498
35.0017043	29.25	2.00170431
39.4039687	33	2.65396871
34.1007173	36.75	.100717291
29.3677364	34	-.882263586
27.0727329	30.25	.0727329031
27.735221	27	-.0147790089
28.1432571	27.75	.143257104
25.1696186	28	-.0803913934
28.3281558	25.25	-.171844199
27.0662981	28.5	-.183701888
26.4947237	27.25	-5.27628511E-03

(a)

(b)

(c)

Table 2.8 Numerical output of differential quantization with a summer and a 3-bit uniform quantizer a)Original seq. b)Quant. seq. c)Error

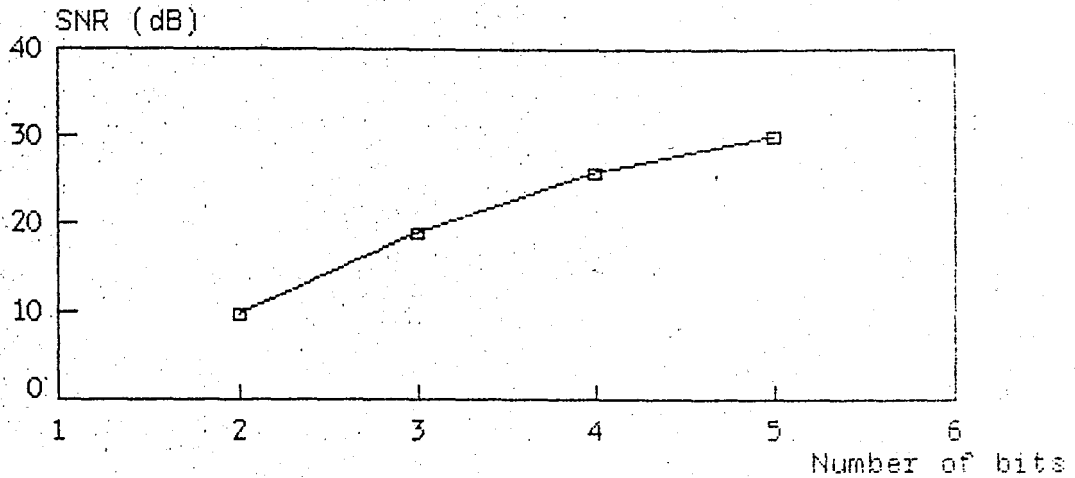


Fig 2.33 SNR vs. the number of bits for differential quantization with a summer and uniform midriser quantizer

2.5.5. LOGARITHMIC QUANTIZATION

As mentioned in section 2.1.1, in order that the percentage error be a constant, the quantization levels must be logarithmically spaced or, equivalently, the logarithm of the input must be quantized.

In this study the logarithm values of the input are taken and then quantized, as shown in Fig (2.5). Five different types of quantizer values, listed in Table (2.1) and (2.3), can be used as quantizer values. Adaptation to signal variance is not performed. AR model sequences are used for all tests. Fig (2.34) shows an example for logarithmic quantization by using a 3-bit uniform midriser quantizer. Table (2.9) gives the numerical values of input sequence, output sequence and the error.

12.0673877	10.6672148	-1.40017287
7.77621378	10.6672148	2.89100106
-6.51000177	-5.21214503	1.29785674
-22.6231155	-23.4819805	-1.858865015
-25.2278535	-23.4819805	1.74587296
-18.202564	-23.4819805	-5.27941652
-20.1028189	-23.4819805	-3.37916157
-27.6754566	-23.4819805	4.19347606
-40.300312	-58.3180736	-18.0177617
-29.7420264	-23.4819805	5.26004584
-21.2055822	-23.4819805	-2.27639828
-9.24936983	-10.6672148	-1.41784501
-19.7206689	-23.4819805	-3.76131165
-10.9481944	-10.6672148	.280979518
-15.908261	-23.4819805	-7.57371955
-26.4764477	-23.4819805	2.99446724
-1.540951667	-1.724948723	-1.183997056
-4.38534775	-5.21214503	-8.82679728
-7.6748213	-10.6672148	-2.99239354
-3.75850827	-5.21214503	-1.45363676
-7.41353526	-5.21214503	2.20139023
-10.345583	-10.6672148	-3.21631808
-22.7298954	-23.4819805	-7.752085112
-19.2235244	-23.4819805	-4.25845611
-19.6726523	-23.4819805	-3.80932821
-9.61514765	-10.6672148	-1.05206719
2.42656085	2.65120634	.224645491
5.95206626	5.21214503	-7.739921231
4.65351815	5.21214503	.558626877
14.5423791	10.6672148	-3.8751643
5.50974765	3.21214503	-1.29760262
.616836361	.724948723	.108112362
2.67210901	2.65120634	-0.0209026681
-6.22420064	-5.21214503	1.01205561
-6.12312445	-5.21214503	.910979422
18.4132255	23.4819805	5.068755
15.1274991	10.6672148	-4.46028431
3.09257551	2.65120634	-1.441369177
-1.363629314	-1.377186787	-0.135574726
-11.3596628	-10.6672148	.692447998
-30.0979135	-23.4819805	6.61593299
-31.5460751	-23.4819805	8.06409464
-32.0919405	-23.4819805	8.60995999
-36.0273022	-23.4819805	12.5453217
-36.8683594	-23.4819805	13.3863789
-25.0132685	-23.4819805	1.53128799
-27.9174936	-23.4819805	4.4355131
-27.2528027	-23.4819805	3.77082222
-14.827935	-10.6672148	4.16072012
3.21833441	2.65120634	-5.567128077
11.1407927	10.6672148	-1.473577827
24.7527185	23.4819805	-1.27073801
18.7856585	23.4819805	4.696322
3.66186058	2.65120634	-1.01065424
-6.47838548	-5.21214503	1.26624445
5.74879539	5.21214503	-5.536650367
-5.44446578	-5.21214503	.232320752
3.18827486	2.65120634	-5.573068526
2.23820514	2.65120634	.413001202
-12.108194	-10.6672148	1.44097915
-23.8133189	-23.4819805	.331338435
-15.3506677	-10.6672148	4.68345282
-3.61946002	-2.65120634	.968253679
-9.48809243	-10.6672148	-1.17912241

(a)

(b)

(c)

Table 2.9 Numerical output of logarithmic quantization with 3-uniform quantizer a)Original seq. b)Quantized seq. c)Quant.er

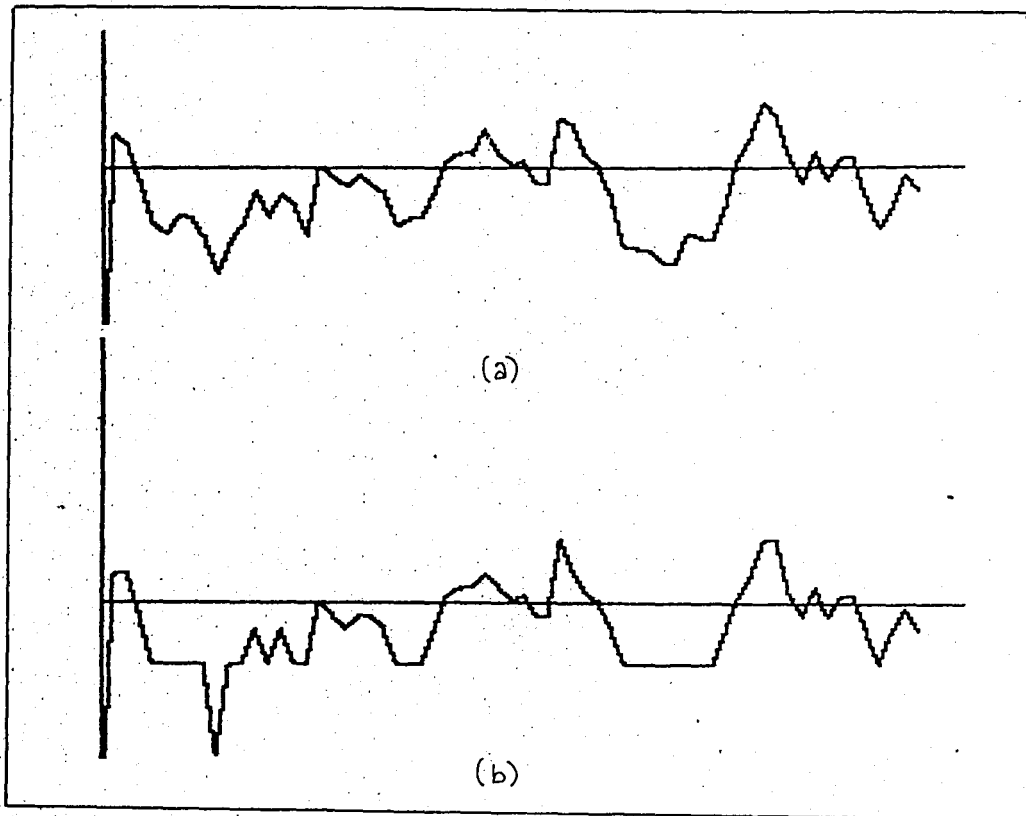


Fig 2.34 Logarithmic quantization with a 3-bit uniform midriser quantizer a) input sequence b) quantized signal

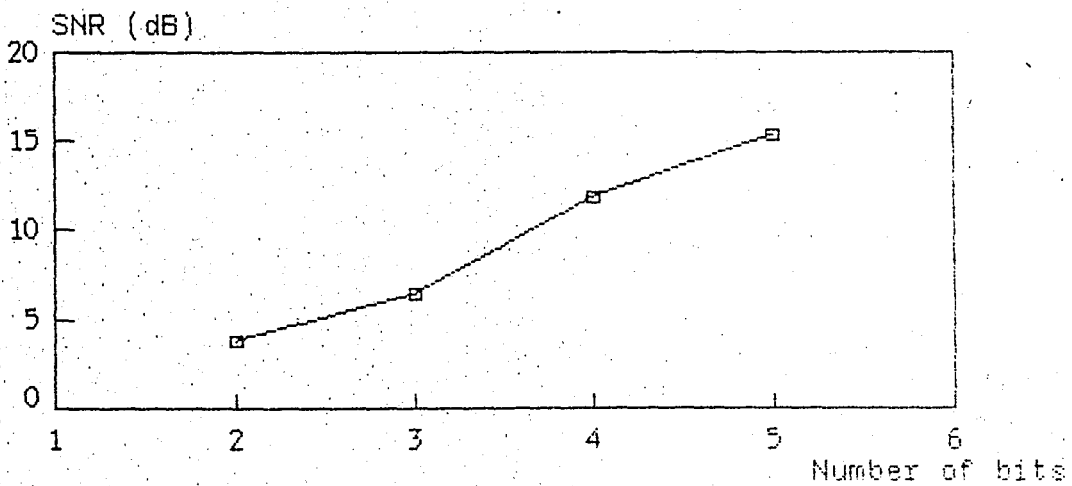


Fig 2.35 SNR vs. the number of bits for logarithmic quantization

AR model sequences are used for all tests. The outputs are taken for 2-, 3-, 4- and 5-bit quantizers and SNR values are calculated for all outputs. Fig (2.35) shows the SNR value vs. the number of bits. The results are not as good as uniform quantization with signal variance adaptation.

2.5.6 FEED-FORWARD ADAPTIVE QUANTIZATION

In section 2.1.2 the theory of adaptive quantization has been discussed in detail. For the feed-forward adaptation scheme in this study the stepsize of the quantizers is evaluated and transmitted every M samples. In this case the system requires a buffer of M samples to permit the quantizer stepsize to be determined in terms of the samples that are to be quantized rather than in terms of past samples. Uniform quantization with signal variance adaptation which has been given in section 2.5.2 is a specific application of the feed-forward adaptive quantization. The buffer is chosen so that it covers all the input samples i.e., $M=N$. The variance is calculated according to the formula;

$$\sigma^2(n) = 1/M \sum_{m=n}^{n+m-1} x^2(m) \quad 1 \leq n \leq N \quad (2.103)$$

The simulations on feed-forward adaptation is done for $M=4,8,16,32$. Fig (2.36) gives the results of the simulation for a 4-bit uniform midriser quantizer. As one can see from Fig (2.36) better results are obtained for smaller M values.

For adaptive quantization five different types of quantizers can be chosen. As for other quantization methods, AR model sequences are used for simulation. The outputs are taken for 2-,3-,4- and 5-bit quantizers and the SNR values are calculated for all outputs. In Fig (2.37) an example is given for feed-forward adaptation with a 4-bit unif. midriser quantizer and $M=8$.

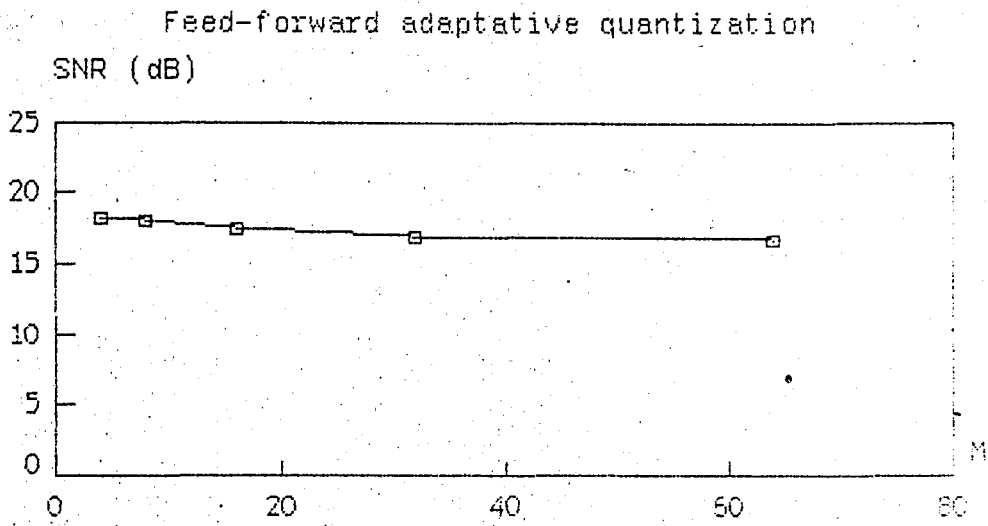


Fig 2.36 SNR vs.M value for feed-forward adaptive quantization using 4-bit uniform midriser values

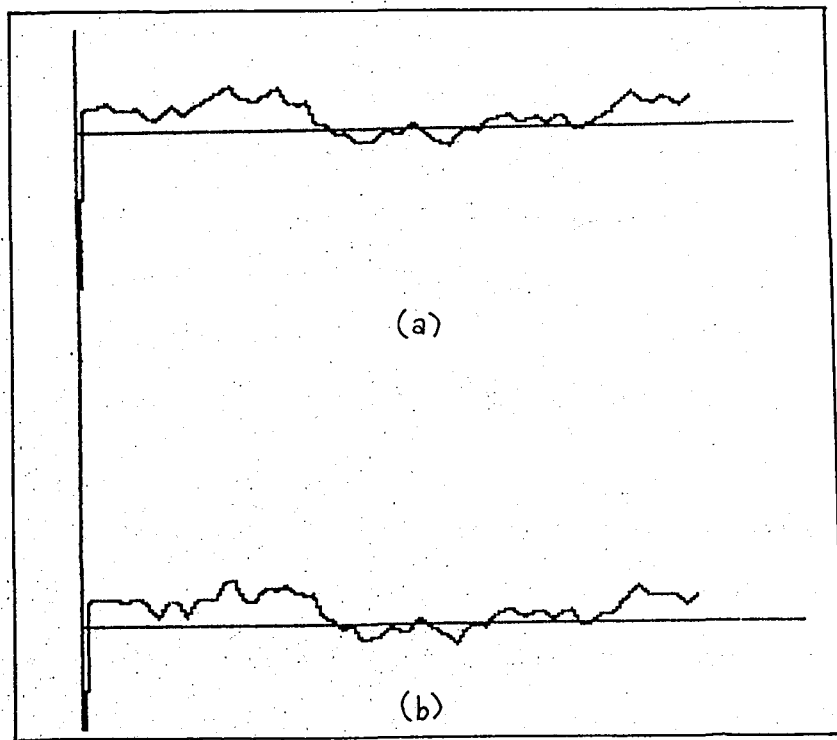


Fig 2.37 Feed-forward adaptation with 4-bit uniform midriser quantizer and M=8, a) input sequence b) quantized sequence

Table (2.10) gives the numerical values of Fig (2.37). In Fig (2.38) the performance of an adaptive quantization with M=8 and 16 is given as SNR vs. the number of bits used in quantization.

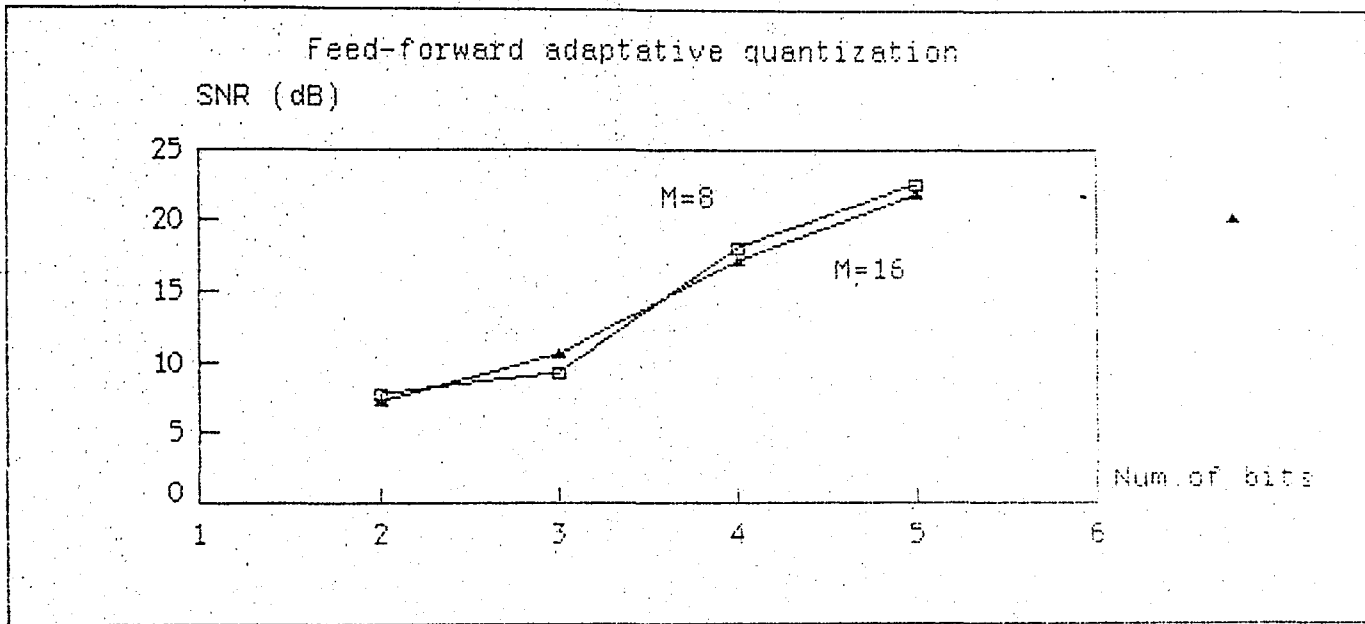


Fig 2.38 SNR vs. the number of bits for feed-forward adaptive quantization using uniform midriser quantizer while M=8 and M=16

2.5.7 DIFFERENTIAL PULSE CODE MODULATION - DPCM

The theory of DPCM has been discussed in section 2.4.2. The program is constructed so that for DPCM five different types of quantizers can be chosen. In addition the predictor can be designed as multilevel predictor upto the sixth order. Predictor coefficients are calculated for each simulation. The DPCM performance is calculated according to the equation (2.54) - (2.57), i.e., the performance criterion SNR can be written as

9.78440667	8.39451726	1.3898894
9.48943623	8.73918842	.75024781
9.60225021	10.1238999	-.521649688
8.87011188	7.38090545	1.48920643
9.11279245	7.58834232	1.52445013
9.59999659	8.89978292	.700213671
6.2572424	5.91049876	.346743643
2.37877703	3.82435652	-1.4455795
8.25352909	6.43114496	1.82238413
8.59933399	9.34884573	-.749511741
2.90343521	5.76617797	-2.86274276
9.11997686	8.76481401	.355162848
9.55964218	10.2108626	-.65122043
10.0915173	12.8137079	-2.7221906
16.4350175	14.2263282	2.20868933
15.6617216	16.0983161	-.436594516
8.85938075	11.1464761	-2.28709536
8.36820934	10.7107463	-2.34253696
13.127072	11.7104993	1.41657271
12.1234876	13.9149544	-1.79146678
14.6432038	15.9032026	-1.25999877
11.3159962	9.93883118	1.37716504
10.0135829	8.67805232	1.33553058
11.5653706	10.9341722	.631198402
2.58472835	1.97477946	.609948889
.869264811	1.02959406	-.160329249
-2.66825491	-2.7183688	.0501138913
-.863906341	-1.1046273	.240720959
-6.06179233	-5.2237875	-.838004829
-5.72409213	-5.1198037	-.604288423
-4.41483956	-4.9743673	.559527744
-1.01585336	-1.789383	.773529645
-3.09736636	-2.3671008	-.730265558
-3.0350962	-2.3657653	-.669330895
1.01016122	1.28995847	-.279797246
-1.02768296	-1.2950749	.267391942
-3.22971716	-4.1098627	.880145536
-5.39280726	-6.3572417	.964434445
-8.42044954	-7.55261681	-.867832743
-2.10356274	-2.7485792	.645016463
-.731840171	-.41489054	-.316949631
-2.19391969	-2.2801188	.0861991123
2.27879809	2.49688466	-.218086566
4.02966639	3.85296339	.176702996
3.70142078	4.17608295	-.474662178
1.92837973	1.88641664	.041963086
3.15862131	2.81588967	.342731639
3.35470381	3.63004374	-.27533993
.772121984	.268697706	.503424278
3.37331412	3.25549163	.117822491
4.68258482	3.93944176	.743143057
-1.69157659	-1.3610019	-.330574686
-1.81651508	-.42481949	-1.39169559
2.02193532	1.32393123	.698004093
2.16377517	3.80371371	-1.63993854
5.67633317	5.64505152	.0312816463
9.46055527	9.25204475	.208510518
13.2447774	12.6730034	.57177398
9.46055527	8.36286009	1.09769517
9.46055527	7.61276387	1.84779139
9.46055527	10.0552417	-.594686434
9.46055527	8.81739546	.643159799
5.67633317	6.84414774	-1.16781457
9.46055527	10.7058495	-1.24529424

(a)

(b)

(c)

Table 2.10 Numerical output of feed-forward adaptive quantization using 4-bit uniform quantizer a) Quantized seq. b) Original seq. c) Quantization error

$$\text{SNR} = \text{SNR}_q \cdot G_p \quad (2.104)$$

where

$$\text{SNR}_q = \frac{\sigma_d^2}{\sigma_e^2},$$

$$G_p = \frac{\sigma_x^2}{\sigma_d^2}$$

where σ_x is the signal variance, σ_d is the difference variance σ_e is the quantization error variance, G_p is the predictor gain and SNR is the SNR for the quantizer. For each simulation, the predictor gain and the quantization gain is calculated. Fig (2.39) shows the variation of the predictor gain as a function of the predictor coefficient for first order AR model sequence. It can be seen from Fig (2.39) that after the fourth or fifth predictor coefficient the predictor gain decreases. Third order predictor is sufficient for DPCM.

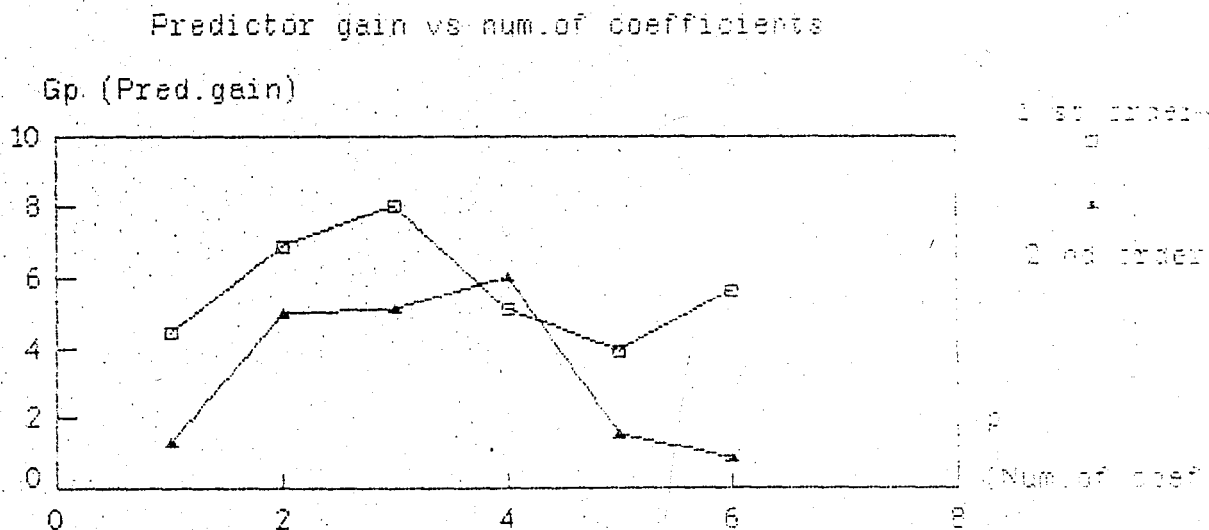


Fig 2.39 Predictor gain vs. the number of predictor coefficients for DPCM quantization of first order and second order AR model sequence

AR model waveforms are used for all tests. An example for DPCM quantization with 4-bit uniform midriser quantizer and a first order predictor output is shown in Fig (2.40).

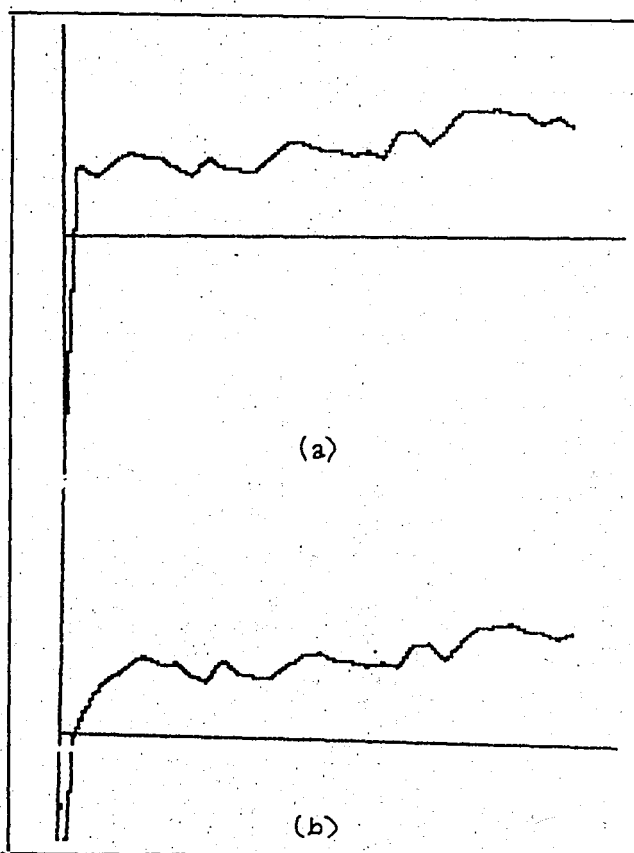


Fig 2.40 DPCM quantization with 4-bit uniform midriser quantizer and a first order predictor for a first order AR sequence a) input sequence b) quantized sequence

Numerical values of Fig (2.40) is given in Table (2.11). SNR is calculated for each quantizer level and 1st, 2nd, 3rd and 4th order predictors. Fig (2.41) shows the performance criterion as a function of the number of bits for 1st and 3rd order predictors using uniform midriser quantizer values for a first order AR seq. as an input.

DRG. SIGNAL	PRED. SIGN.	QY (I)	QUAN. ERR.	PRED. ERR.
25.1036969	0	14.722498	10.3811989	25.1036969
21.8211023	14.722498	6.87049909	.228105173	7.09860426
26.6226572	21.5051254	4.90749935	.210032467	5.11753181
30.7584699	26.2842707	4.90749935	-.433300104	4.47419924
28.1709193	31.0348914	-2.94449961	.0805274807	-2.86397213
28.3936962	27.905159	.981499869	-.492962662	.488537207
24.1609085	28.720106	-4.90749935	.348301876	-4.55919747
21.4456425	23.6411897	-2.94449961	.748952437	-2.19554717
28.9342738	20.5555868	8.83349883	-.45481183	8.378687
24.3529726	29.2663989	-4.90749935	-5.92697039E-03	-4.91342632
23.4544608	24.1842221	-.981499869	.251738619	-.72976125
24.1123313	23.0583778	.981499869	.0724536069	1.05395348
29.3491764	23.902253	4.90749935	.539424073	5.44692342
34.1373273	28.6670909	4.90749935	.562737059	5.47023641
33.5780923	33.4034897	.981499869	-.806897271	.174602598
31.3305121	34.1856197	-2.94449961	.0893920176	-2.85510759
31.5058793	31.037082	.981499869	-.51270261	.468797259
29.4453586	31.833336	-2.94449961	.556522194	-2.38797741
31.692357	28.698838	2.94449961	.0490193553	2.99351896
28.8852432	31.4720476	-2.94449961	.357695166	-2.58680444
38.1868405	28.339706	10.7964986	-.949364092	9.84713447
39.7914502	38.9670581	.981499869	-.157107729	.82439214
33.6488572	39.7159817	-6.87049909	.803374585	-6.0671245
39.2850113	32.6084364	6.87049909	-.19392417	6.67657492
46.0283704	39.2843109	6.87049909	-.126439583	6.7440595
46.1416907	45.9203403	.981499869	-.760149393	.221350476
47.2294794	46.627763	.981499869	-.379783411	.601716459
45.27436	47.3309634	-2.94449961	.887896236	-2.05660337
45.2937406	44.1039673	.981499869	.208273437	1.18977331
41.5868852	44.8222311	-2.94449961	-.290846277	-3.23534589
44.0392643	41.6102084	2.94449961	-.515443753	2.42905586
40.8567129	44.3063561	0		

Table 2.11 Numerical output of DPCM quantization using 4-bit uniform quantizer and first order linear predictor

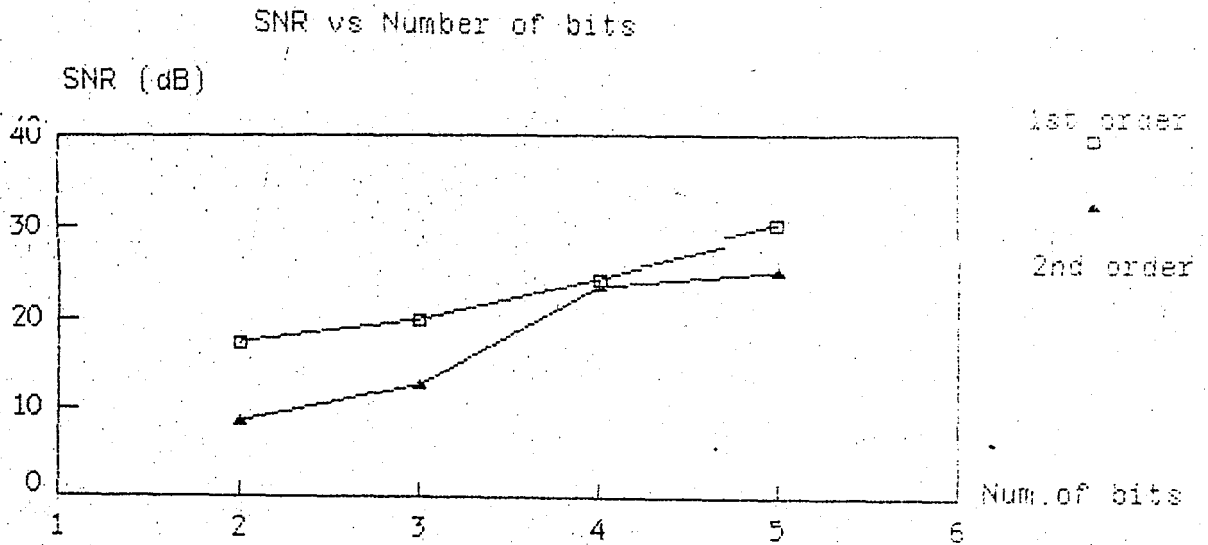


Fig 2.41 SNR vs. number of bits for DPCM quantization with uniform midriser quantizer for first order AR seq. with first order & third order predictor

As expected DPCM systems can provide 5-10 dB improvement over direct quantization (PCM). The greatest improvement occurs in going from no prediction to first order prediction with somewhat smaller additional gains resulting from increasing the predictor order up to 3 or 4, after which a decrease in gain can be observed (see Fig (2.39)).

2.6 CONCLUSION AND SUMMARY

In this chapter the most commonly used picture compression techniques PCM and DPCM have been discussed in detail. Different types of quantizers have been explained and their applications in PCM and DPCM systems have been discussed. In the second part of this chapter, a package program was described and six different types of PCM and DPCM techniques were simulated

on a microcomputer. The program has been designed such that the flexibility of choosing alternative quantization methods, alternative quantizer values and multilevel predictors is given to the user. For each technique the user can find the performance criteria as SNR value and if desired the autocorrelation functions and the spectrum of each the input sequence, quantization noise or autocorrelation function. From the results of the simulation one can easily see the performance of each technique as SNR vs. number of bits used in quantization. The results can be summarized as follows. For uniform quantization each bit adds 6dB to the SNR value. If adaptation with signal variance is added to the method better results are obtained. Logarithmic quantization without adaptation to signal variance gives an SNR value which is below the uniform quantization with signal variance adaptation. The advantage of the logarithmic quantization is that the SNR value is totally independent of the signal variance. It depends only upon the stepsize. Optimum quantization can give better results if the quantization stepsize is matched with the variance of the signal, i.e., whenever the variance of the signal is known. Adaptive quantization gives better results than nonadaptive schemes. About 2-8 dB increase in gain can be obtained. Differential quantization with summer gives 3-10 dB increase in gain, but it is closely dependent on the signal variance and approximately the first ten samples of the input can not be matched. This is the disadvantage of the differential quantization with summer. Differential Pulse Code Modulation (DPCM) gives the best results and can provide from 4 to 11 dB improvement over direct quantization (PCM).

The package program is designed such that further improvements on the package are possible. Alternative PCM and DPCM methods can be added to the package. The only limitation is the storage capacity of the computer.

III. EDGE DETECTION TECHNIQUES AND THEIR COMPARISON

3.1 INTRODUCTION

In image compression discussed in the last chapter, the desired output is a picture - an approximation to, or an improved version of the input picture. Another major branch of picture processing deals with "Image Analysis", "Scene Analysis"; here the input is still pictorial, but the desired output is a description of a given picture or scene. The description refers to specific parts in the picture, or scene, to generate the description, it is necessary to "extract" the picture or "segment" it into these parts. This chapter will discuss an important approach to picture extraction which is based on the detection of discontinuity, i.e., of places where there is a more or less abrupt change in grey level, indicating the end of one region and the beginning of another. Such a discontinuity is called an "edge".

The human observer is the ultimate receiver in many image communication systems. A human observer seems to rely upon edges, boundaries and colour rather than the individual pixel amplitudes for visual recognition and interpretation. Therefore various image coding techniques based upon the detection and the coding of edges and contours have been explored.

The other advantage of edge detection is the potential for reduction of bandwidth required to transmit signals. Since the advent of television in the early part of this century it has been obvious to investigators in this

field that there should be some way of reducing the bandwidth required to transmit a television signal. The basis for the assumption has been that the high bandwidth, necessary for reproducing sharp changes of brightness, is required only at a relatively small number of points in the picture.

In section 3.2 fundamental concepts about edge detection techniques will be summarized. In section 3.3 different edge detection techniques will be discussed. Then evaluation of edge detection schemes performed by Pratt, Fram, Deutsch and Alpaslan will be summarized in order to review the evaluation criteria and techniques. In the last chapter, chapter 4, further research on evaluation and comparison of performances of edge detection techniques will be performed.

3.2 FUNDAMENTALS OF EDGE DETECTION

3.2.1 TEXTURAL AREAS

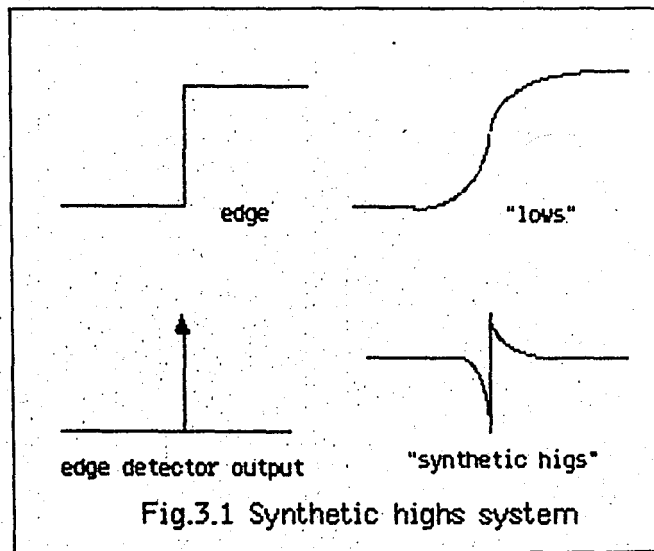
A visual scene or picture is seen as composed of regions and edges which are separate regions. But, not all of these regions can be seen as "objects" at the same time. In general, only the region on one side of an edge is seen, at a given time, as a "figure" that has a shape. If one of the two regions is brighter, or smaller in size, more symmetrical, or bounded, it is generally easier to see that one as the figure. The properties that are important in producing similarity grouping are also important in the perception of "visual textures". These are complex visual patterns composed of entities, or subpatterns, that have characteristic brightness, color, slopes, sizes, etc. Thus a "texture" can be regarded as a similarity grouping. The local subpattern properties give rise to the perceived

brightness, directionality, etc., of the texture as a whole.

3.2.2 GRADIENT AND LAPLACIAN

The detection of details in an image is not easy and is made more difficult by any noise in the signal. Any effort to reduce the noise content of the original picture before processing will greatly simplify the edge detection process. By considering the image in two dimensions the true edges of the picture which lie along connected contours can be isolated from the spatially noncorrelated noise, while "textured" areas of the picture can be eliminated because of the low rate of the brightness changes.

The operation of a "synthetic highs" system using the gradient as an edge detector can best be explained in one dimension with reference to Fig.3.1.



If a signal is unit step, the gradient will be unit impulse. The impulse response of the reconstruction filter must have the form shown in Fig.3.1, such that the original unit step will result when the "highs" are

added to the "lows". Although the dependence of the reconstruction filter on the low-pass filter is obvious in one dimension, the general formulation has been first derived by Schreiber [73] for the gradient edge detector.

A - GRADIENT

The gradient of the picture $G(x,y)$ is a vector signal and has two components.

$$\nabla A(x,y) = G(x,y) = \frac{\partial A(x,y)}{\partial x} u_x + \frac{\partial A(x,y)}{\partial y} u_y \quad (3.1)$$

where u_x and u_y are unit vectors in the x and y directions respectively. The reconstruction filter $H(x,y)$ must also have two components to operate on the gradient signal,

$$H(x,y) = H_x(x,y) u_x + H_y(x,y) u_y \quad (3.2)$$

To have the output equal to the input, the following must hold:

$$\nabla A \otimes H + A \otimes M = A \quad (3.3)$$

where \otimes indicates convolution, H is the reconstruction filter and M is the lowpass filter. $\nabla A \otimes H$ gives the synthetic highs, i.e., high frequency version, and $A \otimes M$ gives the low frequency version of the picture.

B - LAPLACIAN

Using the Laplacian as an edge detector,

$$\nabla^2 A(x,y) = C(x,y) = \frac{\partial^2 A(x,y)}{\partial x^2} + \frac{\partial^2 A(x,y)}{\partial y^2} \quad (3.4)$$

the signal is a scalar and requires a filter, $f(x,y)$, for reconstruction [73]. The $\nabla^2 A$ signal is a scalar and thus is easier to store and filter than the gradient, ∇A , but since it involves a higher derivative, the Laplacian signal will be more adversely affected by noise in the original and by any nonlinear operation such as thresholding.

3.2.3 DIRECTIONAL DERIVATIVES

Derivative operators, which give high values at points where the grey level of the picture change rapidly. Evidently any such operator can be used as an edge detector; its value at a point represents the "edge strength" at that point, and we can explicitly extract sets of edge points from the picture by thresholding these values.

The simplest derivative operators are the first partial derivatives $\partial A(x,y)/\partial x$ and $\partial A(x,y)/\partial y$, as were seen in Sec.3.2.2, which give the rates of change of gray level in the x- and the y-directions. The rate of change in any direction Θ is a linear combination of these:

$$\frac{\partial A(x,y)}{\partial x'} = \frac{\partial A(x,y)}{\partial x} \cos\Theta + \frac{\partial A(x,y)}{\partial y} \sin\Theta \quad (3.5)$$

For digital pictures, we use differences instead of derivatives:

$$\Delta_x A(i,j) = A(i,j) - A(i-1,j) \quad (3.6a)$$

$$\Delta_y A(i,j) = A(i,j) - A(i,j-1) \quad (3.6b)$$

$$\Delta_\Theta A(i,j) = \Delta_x A(i,j) \cos\Theta + \Delta_y A(i,j) \sin\Theta \quad (3.6c)$$

It should be noted that the values of these operations can be either positive or negative, depending on whether the grey level goes upward or downward as one moves in the positive x - (or $-y$, or $-\Theta$) direction. If one wants operations that always have nonnegative values at edges, absolute values of derivatives or differences can be used.

A directional derivative (or difference) measures only the component of the rate of change of grey level in one particular direction. Suppose, for example, that the grey level is given by the linear ramp function $A(x,y) = a(x\cos\varphi + y\sin\varphi) + b$ over some portion of the picture. Here the x -component of the rate of change is $\partial A(x,y)/\partial x = a\cos\varphi$; the y -component is $\partial A(x,y)/\partial y = a\sin\varphi$; the component in direction φ is

$$\frac{\partial A(x,y)}{\partial x} \cos\varphi + \frac{\partial A(x,y)}{\partial y} \sin\varphi = a(\cos^2\varphi + \sin^2\varphi) = a \quad (3.7)$$

Similarly, the components in direction $\varphi + (\pi/2)$, $\varphi + \pi$ and $\varphi + (3\pi/2)$ are 0, $-a$ and 0, respectively. Thus the partial derivatives in various direction give responses to this ramp edge that vary from 0 to a , depending on their orientations relative to the ramp direction.

For digital pictures, we can use differences in place of derivatives in the previous definitions. Thus the magnitude of A at (i,j) is $\sqrt{\Delta_x A(i,j)^2 + \Delta_y A(i,j)^2}$

It is common practice to approximate this expression, either by

$$|\Delta_x A(i,j)| + |\Delta_y A(i,j)| \quad (3.8)$$

or by $\max(|\Delta_x A(i,j)|, |\Delta_y A(i,j)|)$ (3.9)

However, these approximations are no longer equally sensitive to edges in all directions. They agree with the exact expression for horizontal or vertical edges. For a 45° edge, where $\Delta_x A(i,j) = \Delta_y A(i,j)$, we have

$$\sqrt{\Delta_x A(i,j)^2 + \Delta_y A(i,j)^2} = \Delta_x A(i,j)\sqrt{2} \quad (3.10a)$$

$$|\Delta_x A(i,j)| + |\Delta_y A(i,j)| = 2\Delta_x A(i,j) \quad (3.10b)$$

$$\max(|\Delta_x A(i,j)|, |\Delta_y A(i,j)|) = \Delta_x A(i,j) \quad (3.10c)$$

so that the approximations can give values that are too high or too low, respectively, by a factor of as much as $\sqrt{2}$.

Various other approximations to digital difference or gradient are often used. For example one can use

$$\max_{u,v} |A(i,j) - A(u,v)| \quad (3.11)$$

where the max is taken over some set of neighbors (u,v) of the point (i,j) , e.g., its four horizontal and vertical neighbors, or its eight horizontal, vertical and diagonal neighbors. The various digital gradient approximations yield different numerical "edge values" for any given picture. When however, we display these values in picture form, representing edge values by grey levels, the results all tend to look quite similar.

3.2.4 TWO DIFFERENT APPROACHES TO EDGE DETECTION

There are two distinct ways of viewing edge detection. They might be called respectively the signal processing approach and the artificial intelligence approach. The first one can be briefly described as follows. In real images the notion of edge implies a variation of brightness. It can be represented punctually by a vector whose components are continuously varying functions. If a decision is made to retain "important" variations, the result can be called edge or contour.

In the second approach, a regional and textural property is defined first. Then, all the picture points possessing the same property are identically labeled, forming thus a region. The border lines of these regions, which allow the segmentation of the image, can be called contours or edges. If no reference is made to real objects, they may correspond to edges. Here, major edge detection techniques will be reviewed briefly.

Given a picture, the purpose of edge detection is to produce a binary image, of the same size as that of the original, where each picture element (pixel) has the label "edge" or "not edge".

Edge detection methods may be classified in a number of ways at various levels like preprocessing and labeling. A key level is the one where the decision to label or not a pixel as an edge-point has to be taken. Before this labeling stage, and at the same time to make a decision as correct as possible, the pictures are usually preprocessed. The preprocessing level may be viewed as "edge enhancement". A distinction can be made between local, regional or global methods depending on the size of neighbourhood used for preprocessing. Local methods attempt to approximate differentiation within a small window of size 2 by 2 or 3 by 3. Commonly used operators are those of gradient, Sobel, Prewitt, Kirsch. Regional methods try to find the best match between a region of the image and a set of idealized edge configurations. This idea is due to Hueckel.

The general idea for regional methods can be described as template matching. Given a set of N idealized edge configurations (+1, -1 or 1 and 0) within a circular or square region, the problem is to find the mask which matches the best analyzed region surrounding every pixel of the original image.

Finally, global methods attempt to filter the entire image, linearly or non-linearly, aiming to keep the edges, while eliminating the rest as much as possible. Linear shift invariant filtering is the most commonly used technique. The operators mentioned as local are also examples of non-linear filters for edge enhancement.

These preprocessing operators may also be classified into linear or nonlinear methods. In each case the resulting picture is a grey level vector image. A decision should then be made on each pixel, according to a criterion, to decide whether it must be labeled as an edge or not. The decision can be done either by thresholding the magnitude of the variation

vector, i.e. gradient image, or by ridge riding. In the first case the contours are usually thick and it might be necessary to skeletonize them. In both cases, post processing is often needed to connect or disconnect edge segments, or to filter them.

From the implementation point of view, a distinction can be made between parallel and sequential methods. The first ones can be applied to each pixel independently whereas the second ones find a first edge pixel and proceed sequentially to "connected" neighbors. From the conceptual point of view a distinction can also be made between information dependent and information independent methods where the information source is the image being processed.

The result obtained after preprocessing for edge enhancement is a vector image. At each pixel, the magnitude of this vector indicates the "strength" of the edge and its angle the "direction" of the edge. The next step is to make a decision at each pixel whether it is an edge point or not, on the basis of some pointwise or regional information. There are two alternatives to this problem, each with several variations. The first one is thresholding and the second one is ridge riding.

Thresholding consists of comparing a parameter extracted from the enhanced edge image and representing a given pixel to a threshold, and decide for an edge point if it exceeds it. Thresholding will be discussed in section 3.2.5.

The other alternative, i.e., ridge riding, is basically a sequential method and functions as follows: Scanning the image line by line, all the pixels are tested against a Contour Start Threshold (CST). When a pixel is found

that exceeds the CST, its neighbourhood is searched for a local maximum. This maximum is the starting location for a contour, and its coordinates are stored in a contour buffer. The algorithm then selects the largest pixel in the 3x3 neighbourhood of the local maximum, and stores it into the contour buffer, initiating the ridge riding process: find the largest pixel in the direction pointed to by the previous pair of elements in the buffer, plus/minus 45° , and test it against the Contour Continuations Threshold (CCT). If the pixel found exceeds the CCT, it is stored into the contour buffer and ridge riding proceeds to the next element. If a pixel is found to be below the CCT, the search is terminated. However, since the contour may extend in both directions from the starting location, the second possible path is searched and traced similarly. In addition, the points traced out are flagged to avoid retracing. The restriction of plus or minus 45° can be modified if high curvature contours are also searched.

This method has several substantial advantages: CST may be set fairly high so that most spurious objects are ignored. On the other hand, the CCT may be set very low, which permits to track contours of very uneven contrast without breaks, eliminating the need for edge linking operations. Secondly, this algorithm produces one element wide contours which are directly suitable for feature extraction without thinning. Thirdly, the buffer length is a first measure of contour acceptance and short segments can be eliminated immediately.

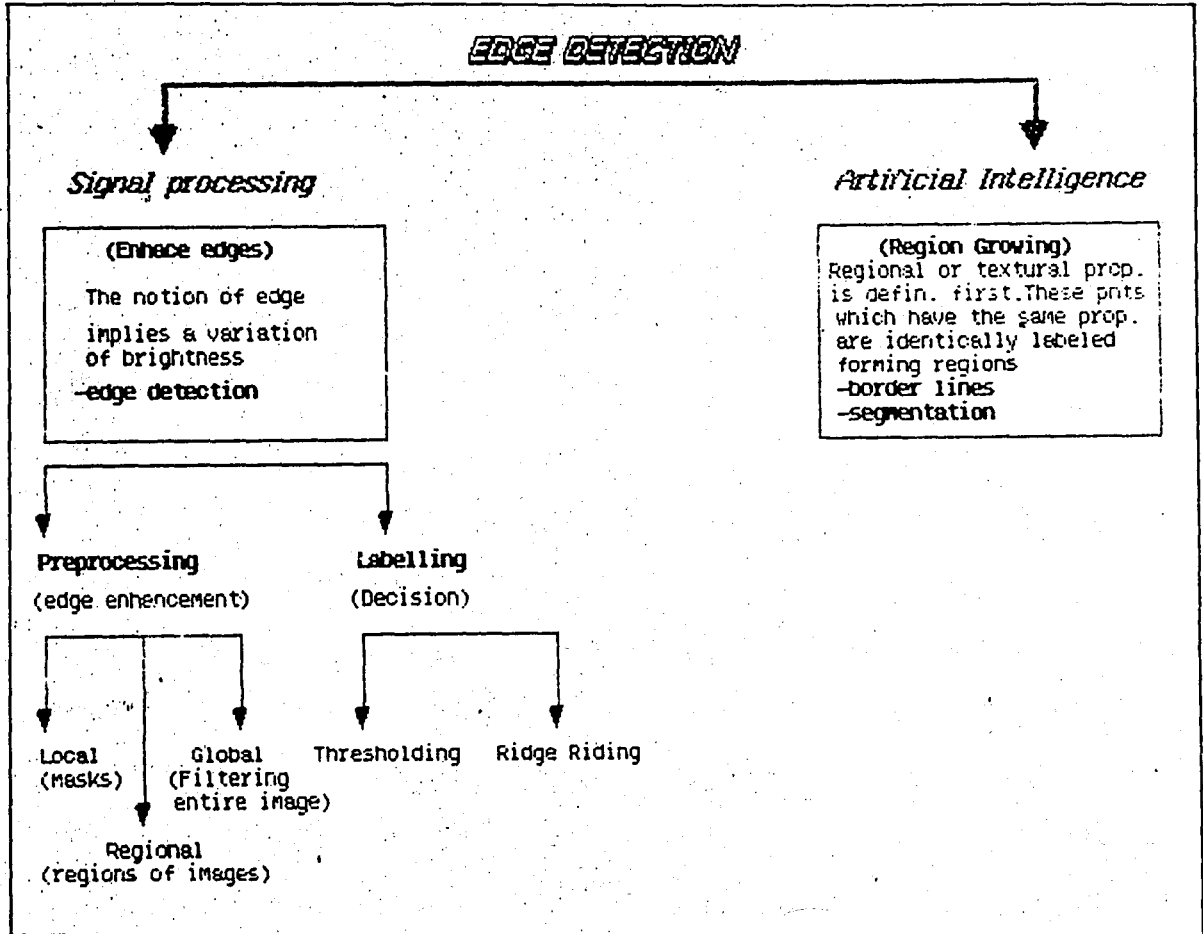
The main disadvantages are the complexity due to the sequential nature of the method and the arbitrariness in the selection of the threshold CST and CCT.

REGION GROWING

Edge detection by region growing can be viewed as the artificial intelligence approach of the problem. The general idea is the following. We assume that the image is composed of regions. A region is defined as an area in the picture whose pixels have a common property. The operation which consists of separating the image into such regions, is called segmentation. Depending on the property, the frontier points of regions can be interpreted as edge points. Region growing requires first the property to be defined, (e.g., the magnitude of the digital gradient), then each pixel of the picture must be labeled accordingly, i.e. must belong to a region. The pixels of a region are therefore connected and consequently are surrounded by a closed border. Finally the region must cover the maximum possible number of pixels.

For the border lines of the regions to correspond to edges or to estimate them, the property has to be selected very carefully. One possible estimation is to use a property of small grey level variations. Let δB be a small brightness interval compared to the full dynamic range of the image. Starting from a given pixel of grey level G , a region can be grown by searching connected points whose grey levels are within the interval $G+\delta B$ and $G-\delta B$. In order to intercept a maximum number of pixels, the interval $2\delta B$ can be moved up and down in the brightness scale. However, such displacement should be limited so that previously intercepted points remain in the region.

We can summarize the classes of edge detection techniques as follows.



3.2.5 THRESHOLDING

The thresholding consists of comparing a parameter extracted from the enhanced edge image, and representing a given pixel to a threshold, and decide for an edge point if it exceeds it. Two questions that arise are: What parameter and what threshold. The most commonly used parameter, especially for practical implementation reasons, is the magnitude of the edge vector at a given pixel. The threshold is selected from an histogram

computed over a region surrounding the pixel (adaptive threshold), or for simplicity, over the entire image (fixed threshold). In both cases there is considerable arbitraryness in selection of the threshold.

In general fixed thresholds selects the big grey level differences and label these as edges in an image. The pixels whose magnitudes are above the given threshold are selected.

Fix threshold eliminates the edges which have small variations in grey levels such as edges in sky and face and produces thick edges.

If an image is observed one can recognize that some edges are very faint, while some are stronger. If the fixed threshold is too low, too many edge points are obtained and if the threshold is too high, then some significant edges are lost. This suggests that improvements of the analog gradient image and the use of a local threshold is necessary in order to bring out most edges and boundaries in natural images.

The edges, which have small variations in grey levels can be selected and one can obtain thinner edges by using adaptive thresholding.

Several improvements for thresholding can be listed. For example the parameter can be a linear combination of edge magnitude in a small window around the pixel, weighted according to their distance to the pixel or to their own magnitude.

Another choice for a fixed threshold value in edge detection is the mean intensity of the gradient image. In the examples, edge maps generated with

fixed threshold are obtained using mean intensity as the threshold on the corresponding gradient image. The third alternative for the fixed threshold can be obtained by using the histogram of the gradient image. More and more edge points can be added by lowering the threshold on the histogram in order to attain a fixed number of edge points in the resultant binary edge map.

An example to the adaptive thresholding is suggested by Robinson [84]. It is discussed in more detail in section 3.3.10. Basically he has suggested that an adaptive threshold can be obtained by comparing the analog gradient image with a blurred version of the original image, which is obtained by a low-pass operation on the image.

Another alternative to the adaptive thresholding is taking averages of magnitudes of pixels in a 3x3 grid, surrounding the pixel of gradient image, which is the one, that must be determined as an edge or not.

E.Alpaslan [87], have taken locally adaptive threshold which is 7x7 grid surrounding the pixel the gradient image ,and chosen the maximum 13 pixels between 7x7 grid.

3.3 EDGE DETECTION TECHNIQUES

A common approach to edge detection is illustrated in Fig(3.2) in which an original image $A(x,y)$ undergoes a grey scale edge enhancement by linear or nonlinear processing to produce an image field $G(x,y)$ with accentuated spatial brightness changes. Next, a threshold operation is performed to determine the pixel location of significant edges. An edge, going in the negative direction exists if

$$G(x,y) < T_L(x,y) \quad (3.12)$$

and a positive going edge exists if

$$G(x,y) \geq T_U(x,y) \quad (3.13)$$

where $T_L(x,y)$ and $T_U(x,y)$ are lower and upper threshold values, respectively.

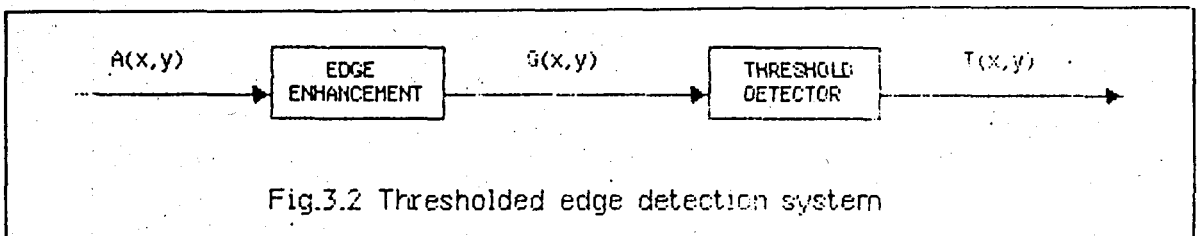


Fig.3.2 Thresholded edge detection system

Some linear and nonlinear edge detection techniques are summarized in the following sections.

3.3.1 LINEAR EDGE DETECTION

A) PRATT

A variety of edge enhancement techniques can be utilized to accentuate edges before threshold detection. One of the simplest techniques is discrete differencing analogous to continuous spatial differentiation [10].

Horizontal edge sharpening can be obtained by the running difference operation, which produces an output image according to the relation

$$G(x,y) = A(x,y) - A(x,y+1) \quad (3.14)$$

Similarly, vertical sharpening results from the operation

$$G(x,y) = A(x,y) - A(x+1,y) \quad (3.15)$$

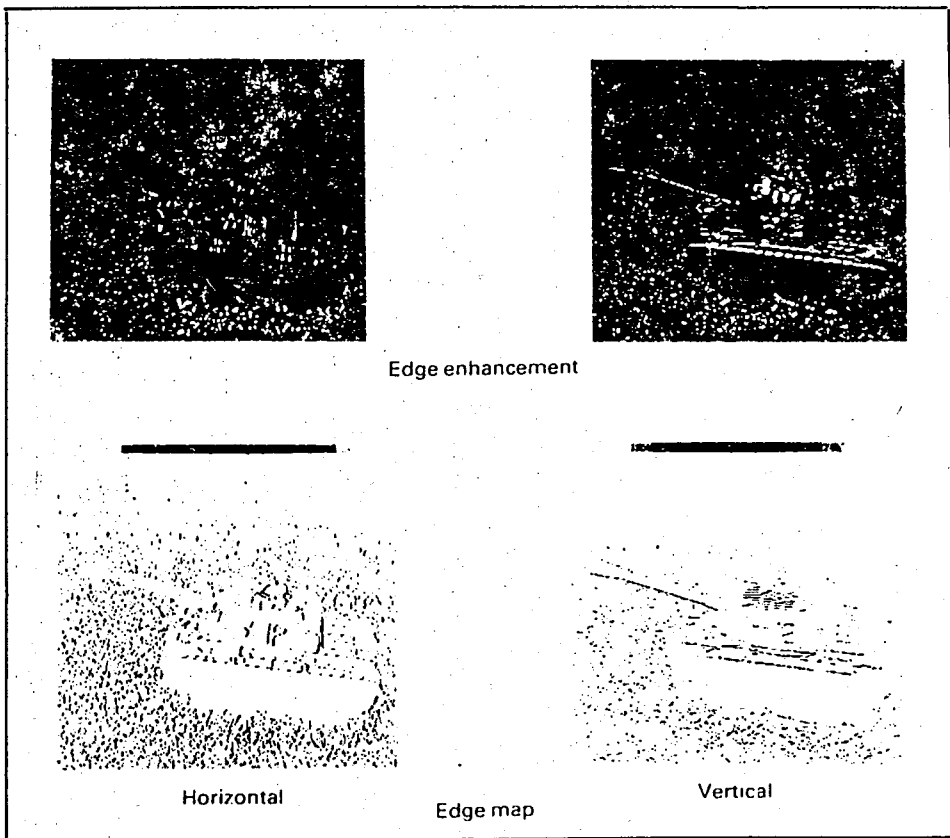


Fig.3.3 Examples of horizontal and vertical differencing edge detection

Diagonal sharpening can be obtained by subtraction of diagonal pairs of pixels. Fig.3.3 which is taken from the study of W.Pratt [10] provides examples of horizontal and vertical differencing edge detection. The edge plots have been obtained by thresholding the magnitude of the difference planes at a threshold level corresponding to the 85% level of the gradient magnitude histogram.

Horizontal edge accentuation can also be accomplished by forming the differences between the slopes of the image amplitude along a line according to the relation

$$G(x,y) = [A(x,y)-A(x,y-1)]-[A(x,y+1) - A(x,y)] \quad (3.16)$$

or equivalently

$$G(x,y) = 2 A(x,y) - A(x,y-1) - A(x,y+1) \quad (3.17)$$

Similar expressions exist for vertical and diagonal slope differences. Pratt has performed two-dimensional discrete differentiation by convolving the original image array with the compass gradient masks listed below.

North

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \quad (3.17.a)$$

Northeast

$$H = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix} \quad (3.17.b)$$

$$\text{East} \quad H = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \quad (3.17.c)$$

$$\text{Southeast} \quad H = \begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (3.17.d)$$

$$\text{South} \quad H = \begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (3.17.e)$$

$$\text{Southwest} \quad H = \begin{bmatrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & 1 \end{bmatrix} \quad (3.17.f)$$

$$\text{West} \quad H = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix} \quad (3.17.g)$$

$$\text{Northwest} \quad H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{bmatrix} \quad (3.17.h)$$

The compass names indicate the slope direction of maximum response; for example, the East gradient mask produces a maximum output for horizontal luminance changes from left to right. It should be noted that the gradient masks have zero weighting (the sum of the array elements is zero), so that

there is no output response over constant luminance regions of the image.

Edge sharpening without regard to edge direction can be obtained by convolution of an image with a Laplacian mask. Several types of Laplacian masks are listed below.

$$\text{Mask 1} \quad H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (3.18.a)$$

$$\text{Mask 2} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (3.18.b)$$

$$\text{Mask 3} \quad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad (3.18.c)$$

Fig.3.4 illustrates the performance of the Laplacian edge detector, which is performed by W.Pratt

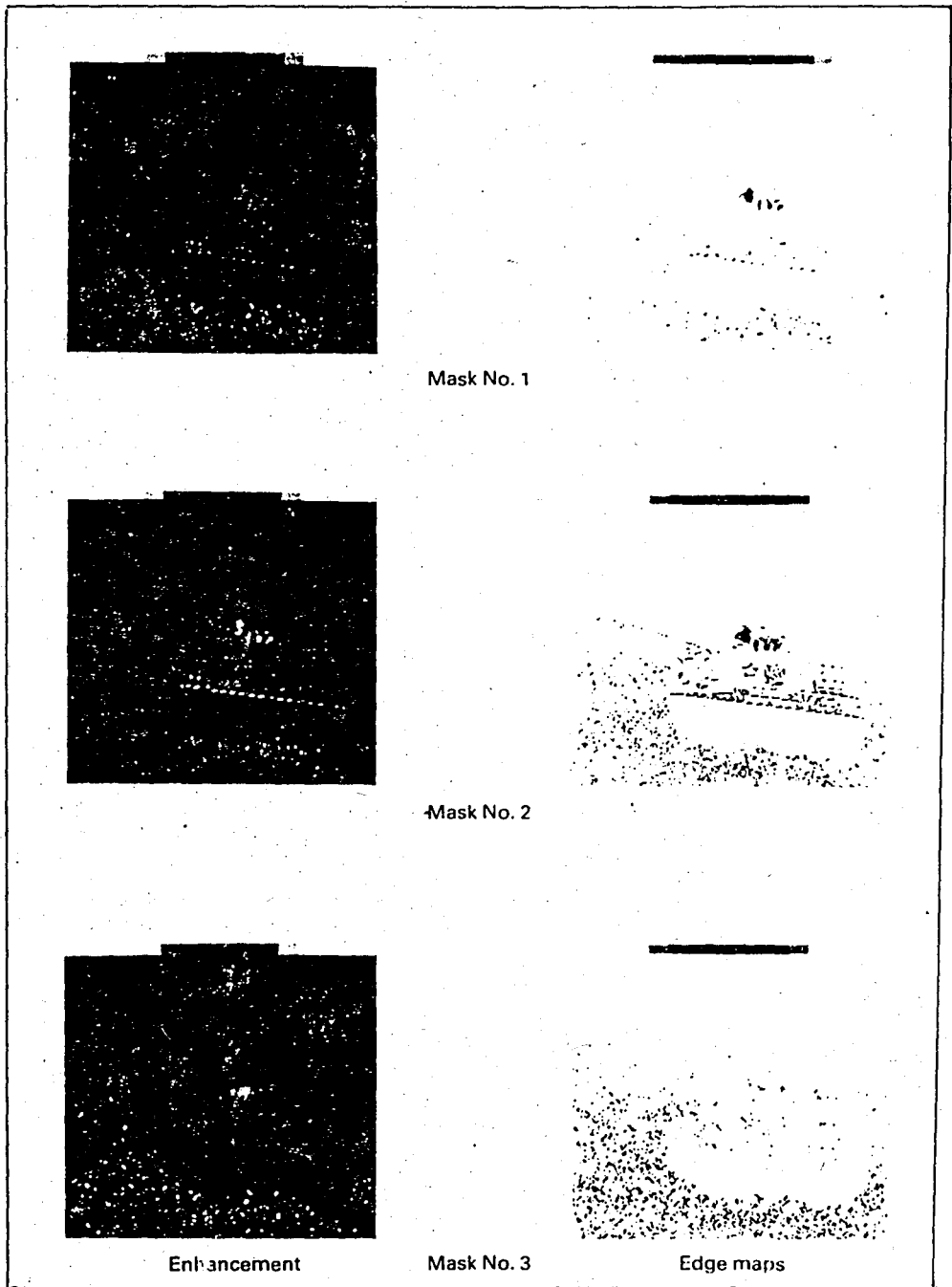


Fig.3.4 Examples of Laplacian edge detection

Edge sharpening can be made proportional to the statistical correlation of pixel values by the statistical mask.

$$H = \begin{bmatrix} \rho_C \rho_R & -\rho_C (1+\rho_R^2) & \rho_C \rho_R \\ -\rho_R (1+\rho_C^2) & (1+\rho_C^2)(1+\rho_R^2) & -\rho_C (1+\rho_R^2) \\ \rho_C \rho_R & -\rho_C (1+\rho_R^2) & \rho_C \rho_R \end{bmatrix} \quad (3.19)$$

in which ρ_R and ρ_C represent the assumed first order Markovian correlation factor between adjacent row and column pixels. If $\rho_C = \rho_R = 0$, there is no adjacent element correlation, and the statistical mask has no effect; in the extreme, if $\rho_C = \rho_R = 1$, the statistical mask reduces to the Laplacian mask of Eq.(3.18.c).

B) ARGYLE and MACLEOD

Argyle [74] and Macleod [75] have proposed Gaussian shaped weighing functions as a means of edge enhancement. The Argyle function is a split Gaussian function defined in one dimension as

$$\begin{aligned} h(x) &= \exp \{ -1/2 (x/p)^2 \} & x \geq 0 \\ h(x) &= -\exp \{ -1/2 (x/p)^2 \} & x < 0 \end{aligned} \quad (3.20)$$

where p is a spread constant.

Macleod's method consists of calculating an edge weighting, at every point of the picture by multiplying the grey level value of each point in a surrounding neighbourhood by the value of the corresponding point of a mask.

The Macleod function given by

$$H_x(x,y) = \exp\{-1/2 (y/t)^2\} [\exp\{-1/2 ((x-p)^2/p)\} - \exp\{-1/2((x+p)^2/p)\}] \quad (3.21)$$

where p and t are spread constants and suppress the effect of pixel values in the edge transition region and edges in rows above and below the edge to be detected. In the above equation x is the component of the distance of the neighbouring point from the original point in a direction perpendicular to the direction of the edge in question, and y is the parallel component. Values p and t are supplied by the user. Examples of edge detection with these masks are presented in Fig.3.5

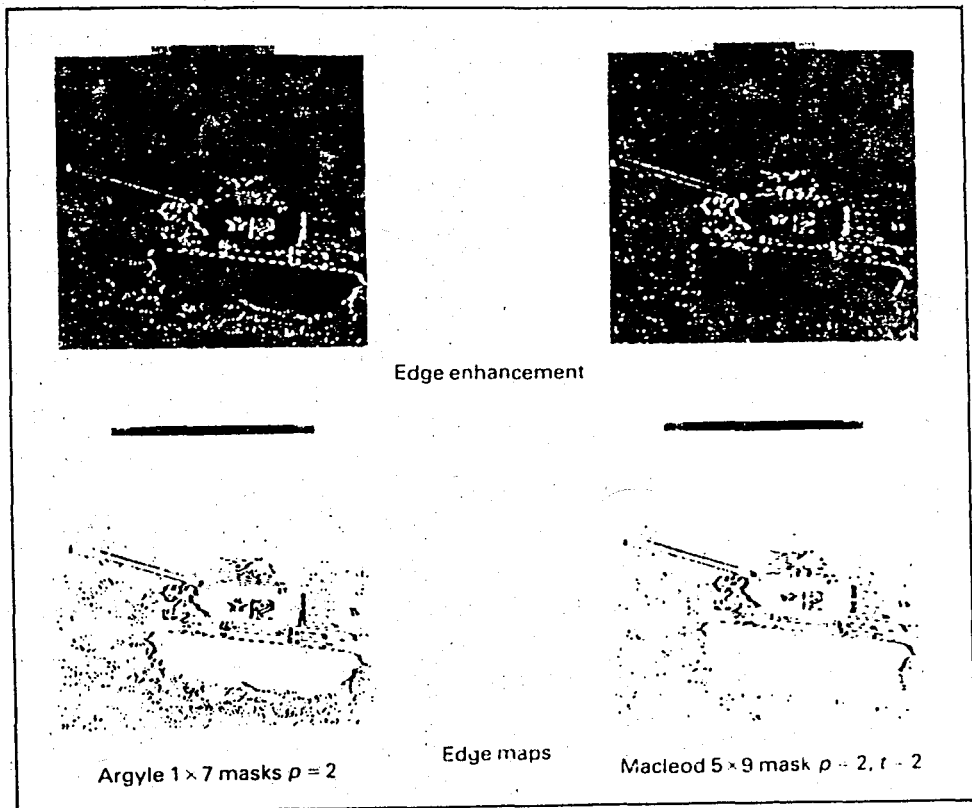


Fig.3.5 Examples of Argyle and Macleod mask edge detection

These schemes have the desirable feature that, if the mask is centered on an edge of the scale and orientation of interest, then the points most likely to indicate such an edge will be weighted most heavily. Points on the edge

itself and points most likely to lie on neighbouring edges will contribute little to the resulting edge weight.

C) HUECKEL- LOCAL VISUAL OPERATOR

The local visual operator due to Hueckel [76] under which the grey level values of the image were analytically fit to members of a set of ideal edge lines whose Gaussian error of approximation to the original image was minimum.

A common limitation of the linear edge sharpening methods previously discussed is the amplification of high spatial frequency noise and artifacts as a result of the inherent differencing operations involved. Noise smoothing can be incorporated into the linear edge sharpening method by performing the linear masking on regions of pixels rather than on individual pixels. This can be accomplished by forming a linear mask

$$H(x,y) = H_S(x,y) \otimes H_E(x,y) \quad (3.22)$$

by convolving one of the edge enhancement masks $H_E(x,y)$ previously defined with a low-pass filter averaging mask $H_S(x,y)$. Such spatial averaging, of course, leads to a smoothing of edges as well as noise. This idea will be discussed later in this chapter.

3.3.2 NONLINEAR EDGE DETECTION

A) ROBERTS

Nonlinear edge detection systems utilize nonlinear combinations of pixels as a means of edge enhancement before thresholding. Most techniques are limited to processing over 2x2 or 3x3 windows. Roberts [77] has introduced the simple nonlinear cross operation,

$$G_R(x,y) = ([A(x,y)-A(x+1,y+1)]^2 + [A(x,y+1)-A(x+1,y)]^2)^{1/2} \quad (3.23)$$

as a two-dimensional differencing method for edge sharpening and edge isolation. Another spatial differencing operation, which is of a computationally simpler form, is given by

$$G_A(x,y) = |A(x,y) - A(x+1,y+1)| + |A(x,y+1) - A(x+1,y)| \quad (3.24)$$

it can easily be shown that

$$G_R(x,y) \leq G_A(x,y) \leq \sqrt{2} G_R(x,y) \quad (3.25)$$

Crude directional information can be extracted by noting which of the four pixels is largest at a detected edge point. Fig.3.6 illustrates the operation of the Roberts square-root and magnitude cross-difference operators.

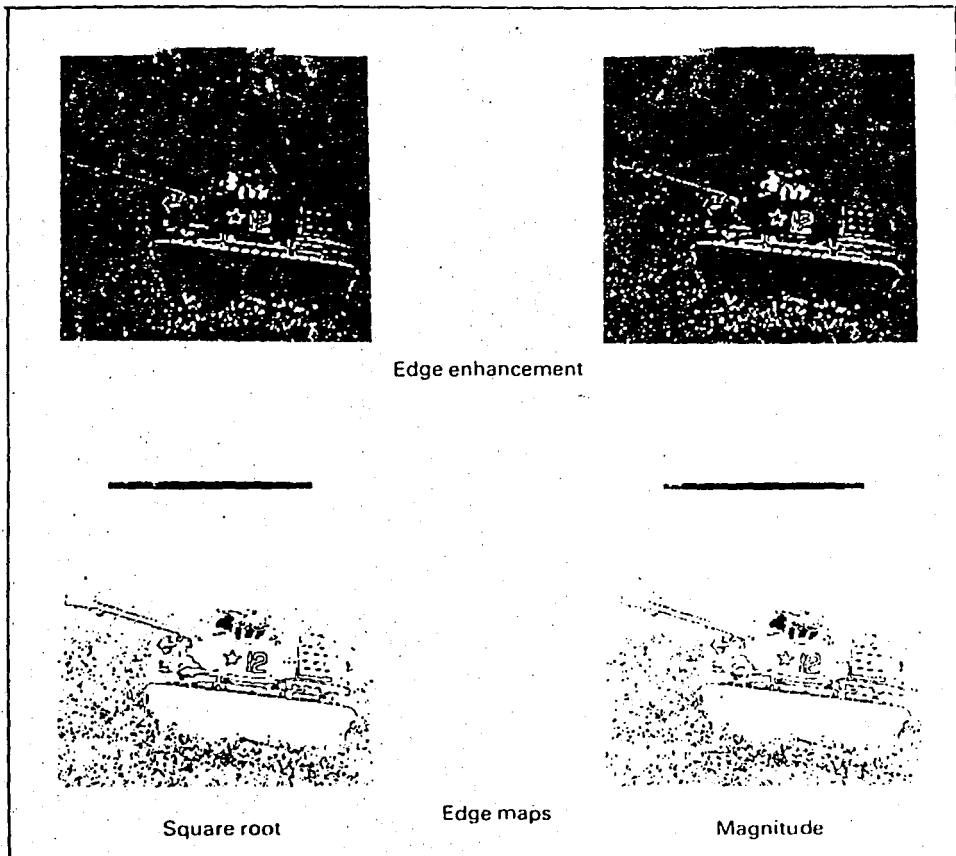


Fig.3.6 Examples of Roberts square-root and magnitude cross-difference edge detection

B) SOBEL

Sobel [79] has used an operator which effectively accomodates the high edge densities of structural scenes and establishes boundary and corner locations accurately, but at the expense of reduced noise tolerance.

He has suggested a 3x3 nonlinear edge enhancement operator described by the pixel numbering convention of Fig.3.7. The edge enhancement plane is defined as

$$G(x,y) = \sqrt{XX^2 + YY^2} \quad (3.26)$$

where

$$XX = (A_2 + 2A_3 + A_4) - (A_0 + 2A_7 + A_6) \quad (3.27.a)$$

$$YY = (A_0 + 2A_1 + A_2) - (A_6 + 2A_5 + A_4) \quad (3.27.b)$$

A_0	A_1	A_2
A_7	$A(x,y)$	A_3
A_6	A_5	A_4

Fig.3.7 Numbering for 3x3 edge detection operators

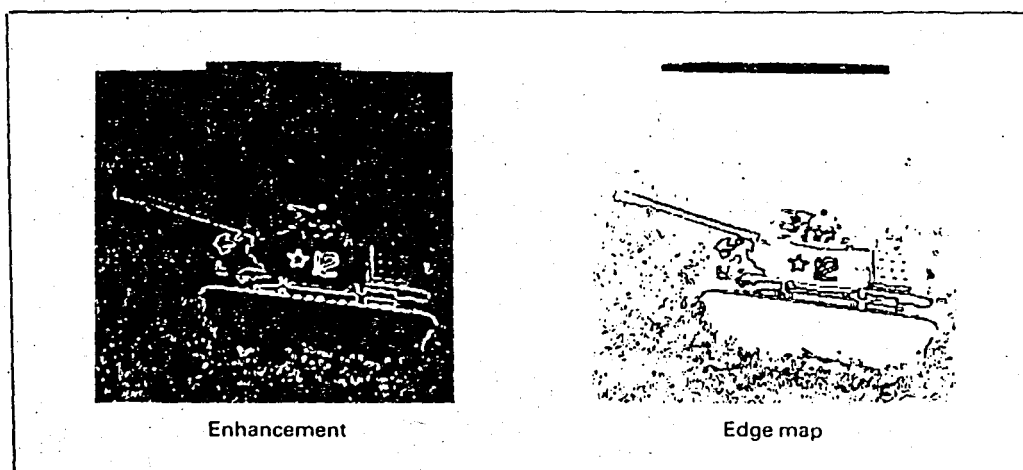


Fig.3.8 Examples of Sobel Edge Detection

After convolving each pixel of the original image by the 3x3 operator, edges are established by the threshold test of edge magnitude.

$$\text{Edge Mag.} = |XX| + |YY| \quad (3.28.a)$$

or

$$\text{Edge Mag.} = \sqrt{XX^2 + YY^2} \quad (3.28.b)$$

$$\text{if} \quad \frac{|XX| + |YY|}{T} > 1 \quad (3.29)$$

edge is established at pixel. In equation (3.29) T is the fixed or the local threshold.

C) KIRSCH

Another 3x3 nonlinear edge enhancement algorithm has been introduced by Kirsch [79]. Referring to the notation of Fig.3.7 the enhancement is given as

$$G(x,y) = \max \{ 1, \max_{i=0}^7 [5S_i - 3T_i] \} \quad (3.30.a)$$

where $S_i = A_i + A_{i+1} + A_{i+2} \quad (3.30.b)$

$$T_i = A_{i+3} + A_{i+4} + A_{i+5} + A_{i+6} + A_{i+7} \quad (3.30.c)$$

The subscripts of A are evaluated modulo 8. Basically, the Kirsch operator provides the maximal compass gradient magnitude about an image point ignoring the pixel value A(x,y). Examples of edge detection with the Kirsch operator are presented in Fig 3.9.

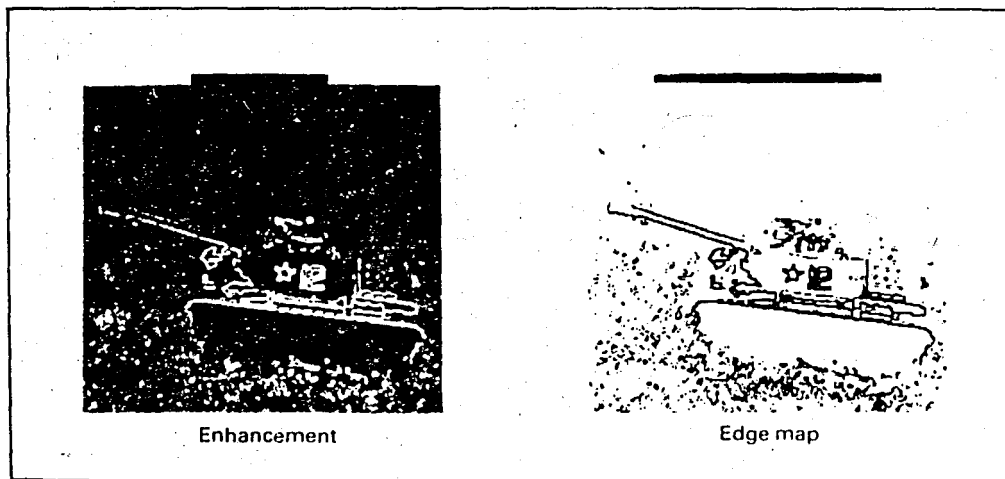


Fig 3.9 Examples of Kirsch Edge Detection

The Kirsch operator is equivalent to writing the eight compass gradient masks in Fig (3.10)

Direction of gradient

Kirsch masks

North

$$\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

Northwest

$$\begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

West

$$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}$$

Southwest

$$\begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}$$

South

$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}$$

Southeast

$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

East	$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}$
Northeast	$\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$

Fig 3.10 Kirsch compass gradient masks

The mask which produces the maximum output determines the direction and the magnitude of the edge. The gradient image is obtained by taking the magnitude of the output of that mask. A binary edge map is obtained by thresholding the gradient image by assigning a 1 to the points which have greater magnitude value and a 0 to the points which have smaller magnitude value than the threshold.

D) PREWITT

A simple set of compass gradient masks can be formed by rotating the differentiation masks (i.e. changing the angle). The compass names indicate the slope direction of maximum response, e.g., the north gradient mask produces a maximum output for vertical luminance changes, i.e., horizontal edges.

$$W_x = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \quad (3.31.a)$$

$$W_y = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix} \quad (3.31.b)$$

given by Prewitt [80] as shown in Fig (3.11)

North

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

Northwest

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

West

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

Southwest

$$\begin{bmatrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

South

$$\begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Southeast

$$\begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

East

$$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

Northeast

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$$

Fig 3.11 Prewitt compass gradient masks

E) WALLIS

Wallis has proposed a nonlinear edge detection scheme based on homomorphic image processing. According to this scheme an edge exists if the magnitude of the logarithm of the image luminance at a pixel, $A(x,y)$, exceeds the magnitude of the average logarithmic luminance of its four nearest neighbours A_0, A_1, \dots, A_7 , by a fixed threshold value. With reference to Fig(3.7) the edge enhancement plane is defined as:

$$G(x,y) = \log[A(x,y)] - 1/4 \log(A_1) - 1/4 \log(A_3) - 1/4 \log(A_5) - 1/4 \log(A_7) \quad (3.32.a)$$

or equivalently

$$G(x,y) = \frac{1}{4} \log \left| \frac{(A(x,y))^4}{A_1 A_3 A_5 A_7} \right| \quad (3.32.b)$$

Comparison of $G(x,y)$ against upper and lower threshold values is exactly equivalent to comparison of the function in the brackets of Eq.(3.32.b) against a modified threshold. Therefore, logarithms need not be explicitly computed. The principle advantage of the logarithmic edge detector besides its computational simplicity is that the technique is insensitive to multiplicative changes in the luminance level. Fig(3.12) contains examples of logarithmic edge detection.

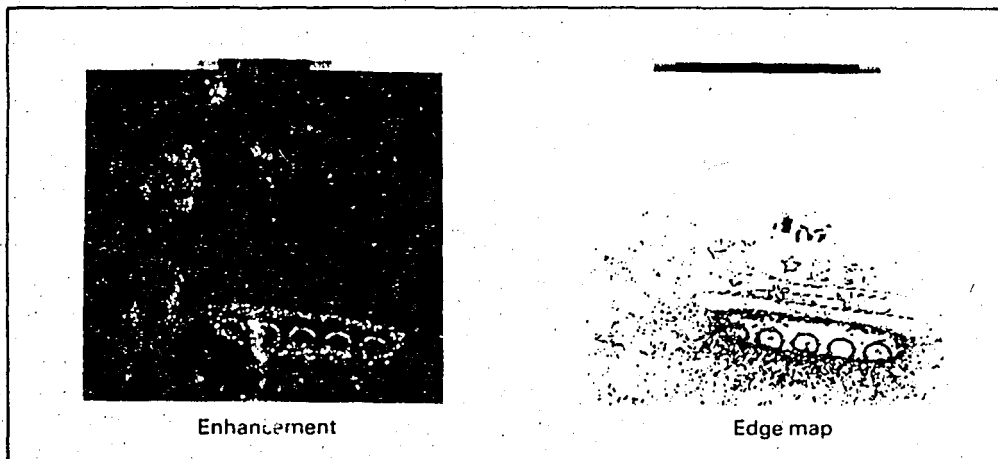


Fig 3.12 Examples of logarithmic edge detection

F) ROSENFELD

Rosenfeld [81] has developed a nonlinear product averaging mask for edge sharpening and edge isolation. Rosenfeld has first developed his technique on one-dimensional functions and then adapted them to two-dimensional pictures. His method can be explained as follows:

The standard edge detection and enhancement techniques generally involve some type of differentiation or differencing [82], which makes them highly sensitive to noise. Absolute differencing of a function, i.e., computing $|f(i+1) - f(i)|$ for each i , gives no spatial prominence to the major "edge".

A straightforward method of emphasizing this edge relative to the noise edges is to take absolute differences of running averages of the function values, i.e., to compute

$$d_k(i) = \left| \frac{f(i+k) + \dots + f(i+1)}{k} - \frac{f(i) + \dots + f(i-k+1)}{k} \right| \quad (3.33)$$

for each i . The results for $k=2,4,8,16$ and 32 are shown in Fig 3.13 (c) through (q) which are taken from Rosenfeld's study.

For large k , the central edge becomes more and more apparent relative to the noise edges. However, the larger the k , the less precisely localized is the detected edge.

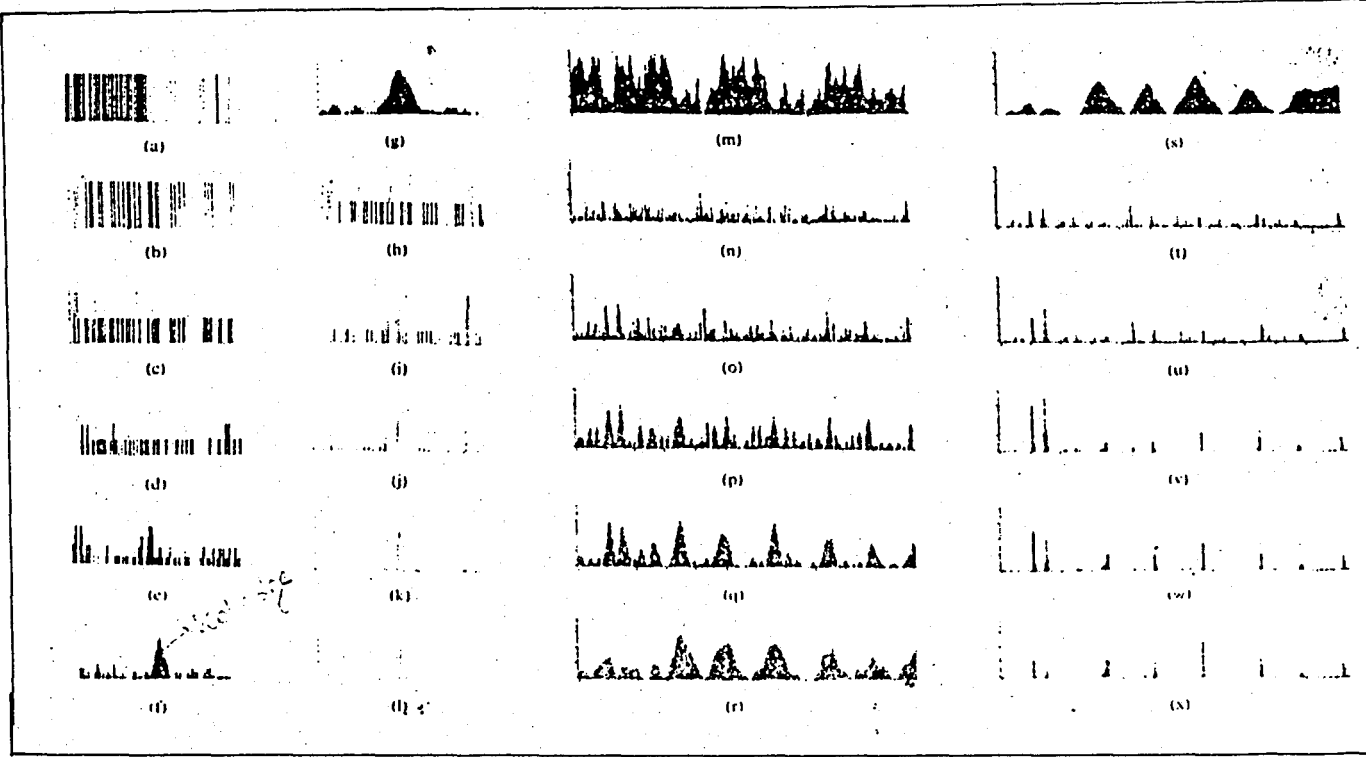


Fig 3.13 Differences of running averages and products of this differences, for a noisy one-dimensional step edge and for a television line

It has been found that, if differences $d_k(i)$ are multiplied together for a range of values of k , the result tends to yield sharply localized detection of major edges while suppressing noise. Intuitively, this is because the product is large only when all its factors are large, and as soon as one moves away from a position "just at" an edge point, the factors with low k 's become small; while if one is not at or near a major edge, the factors with high k 's are small. The results of computing $d_1 \cdot d_2, d_1 \cdot d_2 \cdot d_4, d_1 \cdot d_2 \cdot d_4 \cdot d_8, d_1 \cdot d_2 \cdot d_4 \cdot d_8 \cdot d_{16}$ and $d_1 \cdot d_2 \cdot d_4 \cdot d_8 \cdot d_{16} \cdot d_{32}$ are shown in Fig. 3.13 (h) through (l). Fig. 3.13 (m) through (x) shows analogous results for a single digital television line.

This approach is generalized to two-dimensions by taking differences of

averages over nonoverlapping two-dimensional neighborhoods at each point. The orientation of the pair of neighborhoods determines the direction of the edges which will be detected. For example H_k denotes the difference between averages taken over horizontally adjacent, nonoverlapping $k \times k$ squares in standard orientation, and V_k is defined analogously using vertically adjacent squares. A product of H_k 's is used to detect vertical edges and a product of V_k 's to detect horizontal edges.

Differences in two dimensions can be summarized as follows:

$$H_k(x,y) = \left| \frac{A(x,y+k)+\dots+A(x,y+1)}{k} - \frac{A(x,y)+\dots+A(x,y-k+1)}{k} \right| \quad (3.34.a)$$

$$V_k(x,y) = \left| \frac{A(x+k,y)+\dots+A(x+1,y)}{k} - \frac{A(x,y)+\dots+A(x-k+1,y)}{k} \right| \quad (3.34.b)$$

Fig.3.14 (e) through (h) which is taken from the study of Rosenfeld shows the result of computing

$$\max (H_1 \cdot H_2 \cdot H_4 \cdot H_8 \cdot H_{16}, V_1 \cdot V_2 \cdot V_4 \cdot V_8 \cdot V_{16}) \quad (3.35)$$

for pictures in Fig.3.14 (a) through (d). Those are 72x72 element binary valued digital pictures in which the probabilities of a "1" in the left and right halves are. (0.9, 0.1), (0.8, 0.2), (0.7, 0.3) and (0.6, 0.4), respectively.

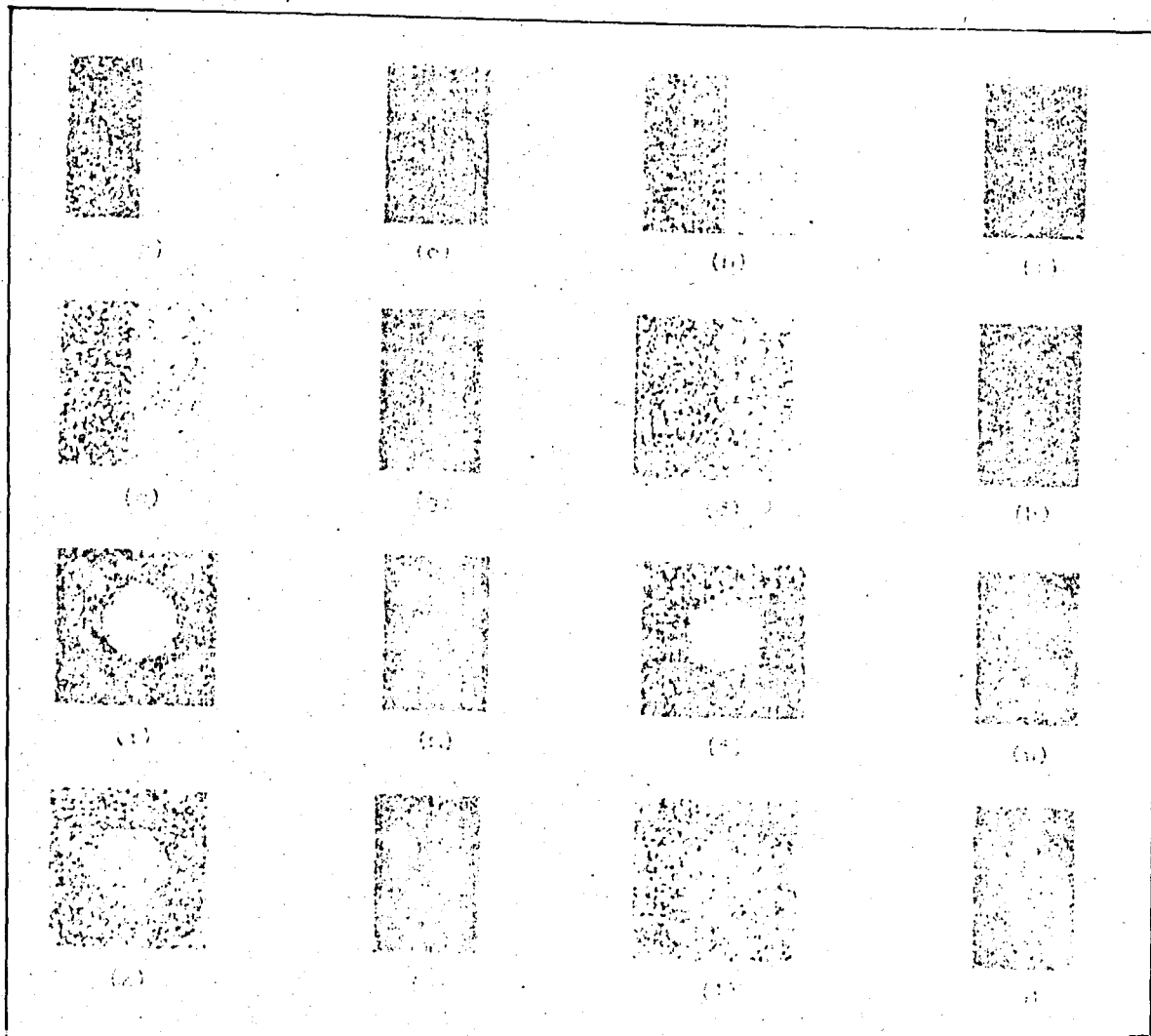


Fig.3.14 Product of differences for a set of noisy vertical edges and a set of noisy circles.

In principle, the scheme described above for detecting edges between regions of different average grey level can be applied to detect a wide variety of "textured areas", in which two regions differ with respect to the average value of some local property.

Rosenfeld and Thurston have also proposed [83] a nonlinear thresholding procedure for isolating large edges in the neighborhood of smaller edges. This procedure, which can be called dominant neighbour suppression, is

performed by scanning the edge enhanced plane $G(x,y)$; i.e., gradient image with a small pixel window. The value of $G(x,y)$ in the center of the window is suppressed (set to zero) unless its magnitude is the greatest of all samples within the window. Conventional amplitude thresholding then follows. A variation of the process is to permit suppression of $G(x,y)$ only if a neighbour in the window dominates by a significant amount. The dominant neighbour suppression thresholding algorithm has proved quite effective for edge detection when coupled with an edge enhancement method that provides some noise smoothing.

G) ROBINSON

Robinson [84], [85] has described a new image coding system which combines the detection and coding of visually significant edges in natural images. The edges are defined as amplitude discontinuities between different regions of an image. The edge detection system makes use of 3x3 masks. Use of an edge direction map improves the simple thresholding of gradient modulus images. He has suggested that the concept of local connectivity of the edge direction map is useful in improving the performance of this method as well as other edge operators such as Kirsch and Sobel. He has also introduced the concepts of an "Edge Activity Index" (EAI) and a "Locally Adaptive Threshold" (LAT).

The directional masks, used by Robinson, can be formed by rotating the differentiation masks,

$$W_x = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad (3.36.a)$$

$$W_y = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad (3.36.b)$$

just as Prewitt has done [80]. These directional masks are called three-level simple masks. The two orthogonal masks W_x and W_y in Eq.(3.36), which measure the gradients in the north and west directions, approximate the partial derivatives in the x-direction and y-direction respectively. Application of W_x and W_y to an image results in spatial differentiation in two orthogonal directions. The gradient magnitude and direction can be obtained by taking the magnitude and direction cosines at each point. An enhanced picture results when the gradient magnitudes are displayed as grey values. A set of five-level simple directional masks are considered for obtaining the analog gradient image and the edge direction in a simple manner. These masks, called five-level simple masks, contain five integer weights between -2 and +2. Three-level and five-level simple masks are shown in Fig 3.15. The two orthogonal masks

$$M_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (3.37.a)$$

$$M_y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (3.37.b)$$

approximate the partial derivatives in the x-direction and y-direction respectively. Fig 3.16 shows the eight principle direction on a 3x3 grid.

<u>Direction of edges</u>	<u>Direction of gradient</u>	<u>Three-level simple masks</u>	<u>Five-level simple masks</u>
0	North	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
1	Northwest	$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$
2	West	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
3	Southwest	$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$
4	South	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
5	Southeast	$\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$

6.	East	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & -2 \\ -1 & 0 & 1 \end{bmatrix}$
7.	Northeast	$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$

Fig 3.15 Three-level and five-level simple masks

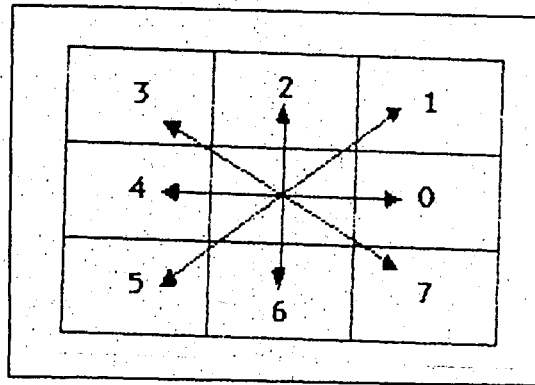


Fig 3.16 The eight principle directions on a 3x3 grid

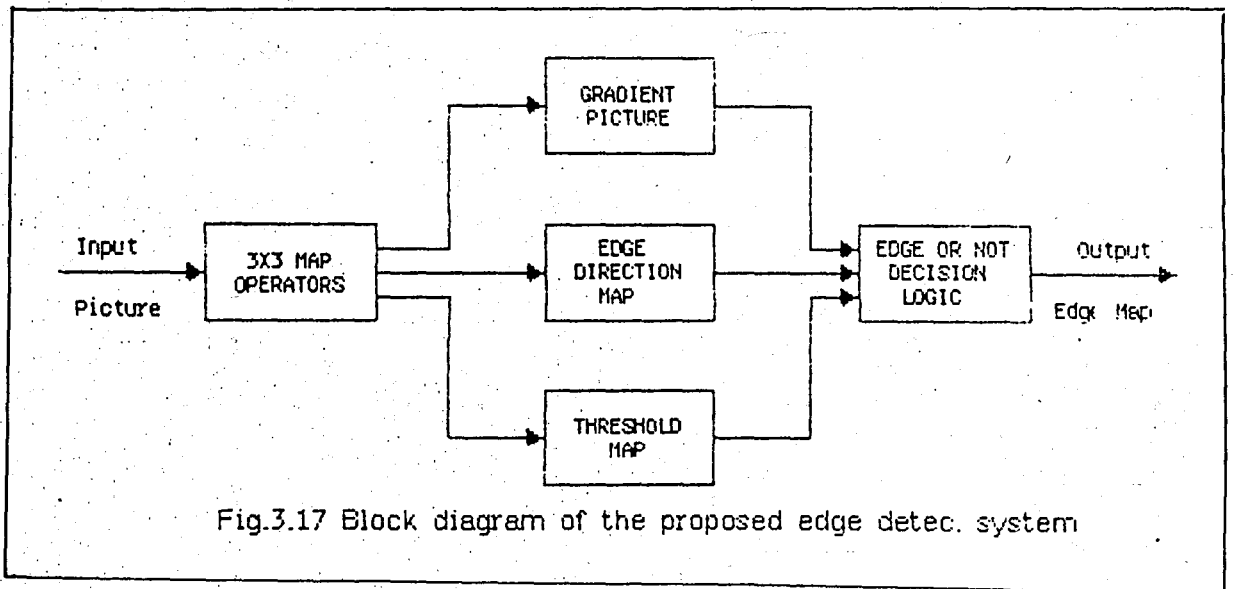
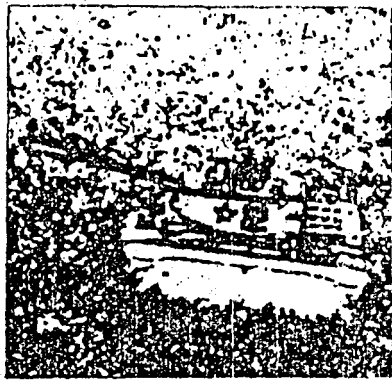


Fig.3.17 Block diagram of the proposed edge detec. system

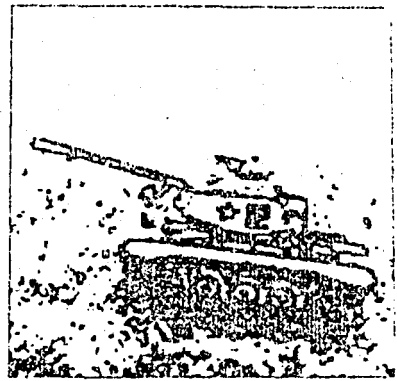
Fig 3.17 Block diagram of the proposed edge detection system

Fig 3.17 shows a block diagram of the proposed edge detection system. The application of the simple masks to a 3x3 grid surrounding a picture element gives the gradient magnitude and direction (See Fig 3.16). The gradient picture is obtained by taking the maximum gradient magnitude at each point. The mask which yields the maximum gradient value determines the direction of the edge. The edge map thus generated is a two-dimensional array of numbers which range between 0 and 7. The edge map is used to determine the local connectivity. If the direction at the center of the 3x3 grid is k ($k = 0, \dots, 7$), and if the directions of the preceding and succeeding edge vectors are $k-1$ or $k+1$, for any of the eight compass directions then the edges are connected. The threshold map is used in determining whether the gradient value is large enough to accept or reject the presence of an edge point. The presence of an edge is determined by examining the gradient values, edge direction map and the threshold map simultaneously. If the edge vector in a 3x3 grid surrounding a point satisfy the local connectivity conditions and if they are above the threshold, set by the threshold map, then it is determined that there is an edge point. Thus, a binary edge map is generated at the output.

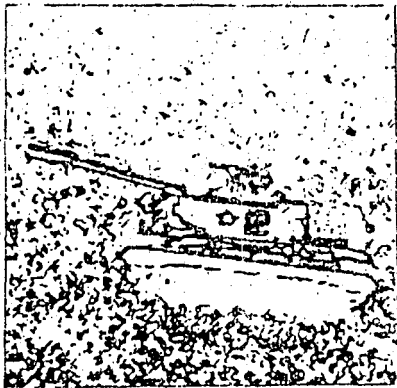
Fig 3.18, which is taken from the study of Robinson, shows the importance of the three basic blocks of the edge direction system for extracting the edges in a picture of a toy tank. Comparison of figures 3.18 b,c and d shows that simultaneous use of the local connectivity and locally adaptive threshold will bring out details, such as wheels, and eliminate the spurious edges such as those in the gross area.



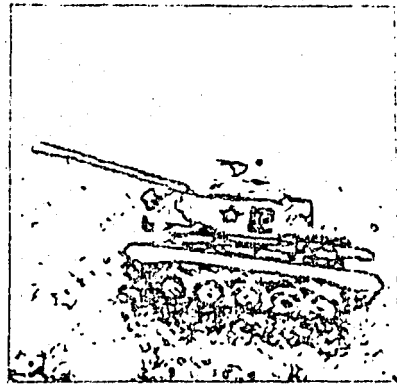
a) Fixed threshold
No connectivity test
23350 edge points



b) Adaptive threshold
No connectivity test
11144 edge points



c) Fixed threshold (same
as a) and connectivity
test; 6789 edge points



d) Adaptive threshold (same
as b) and connectivity
test; 4413 edge points

Fig 3.18 Performance of the edge detection system using
five-level simple directional mask

In general, a suitable fixed threshold value would produce the edges directly. However, if the threshold is too low, too many edge points are obtained and if the threshold is too high, then some significant edges are lost. That's why improvement of the analog gradient image and the use of local threshold is necessary in order to bring out most edges and boundaries in natural images.

Robinson has suggested that improvement of the analog gradient image can be obtained by use of the edge activity index (EAI) defined as the ratio of the

maximum gradient magnitude at an image point to the average magnitude of gradients in the eight compass directions. If the eight compass gradient values at a pixel (x,y) are y_0, y_1, \dots, y_7 then EAI is defined as

$$EAI = \frac{\text{Max} \{ |y_k|, k=0,1,\dots,7 \}}{\sqrt{1/8 \sum_{k=0}^7 y_k^2}} - 1 \quad (3.38)$$

This expression becomes simpler in the case of simple masks since only the first four masks are enough to obtain gradient in all eight compass directions. The analog gradient image can be improved by imposing a threshold on EAI. If EAI is greater than some threshold, i.e., the edge activity is considerably superior in the direction of the maximum gradient, then the maximum gradient value is taken, otherwise the gradient value is set to 0. This operation results in a sharper histogram for the analog gradient image. A locally adaptive threshold LAT, has been obtain by comparing the gradient image with a blurred version of the original image, which is obtained by low-pass operation on the image. The particular low-pass operation can be performed by the mask:

$$M_0 = 1/16 \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (3.39)$$

Thus, the locally adaptive threshold has been defined as:

$$LAT = \frac{\text{Max} \{ |y_k|, k=0,1,\dots,7 \}}{\text{Output of the low-pass filter } M_0 \text{ at pixel } (x,y)} \quad (3.40)$$

The result of the use of a locally adaptive threshold is shown Fig 3.18

3.4 - QUANTITATIVE EVALUATION OF EDGE DETECTION SCHEMES

Relatively few studies of edge detector performances have been reported in the literature. A performance evaluation is difficult because of the large number of proposed methods, difficulties in determining the best parameters associated with each technique and the lack of definitive performance criteria. In developing performance criteria for an edge detector it is wise to distinguish between mandatory and auxiliary information to be obtained from the detector. Obviously, it is absolutely essential to determine the pixel location of an edge. Other information of interest includes the height and slope angle of the edges as well as its special orientation. Another useful item is a confidence factor associated with the edge decision, for example, the closeness of fit between actual image data and the idealized edge model. Unfortunately, few edge detectors provide the whole information, mentioned above.

3.4.1 - EDGE DETECTION PERFORMANCE BY PRATT

According to W.Pratt [10] there are three major types of error associated with the determination of an edge location

- (1) - Missing valid edge points; deletion
- (2) - Failure to localize edge points; insertion
- (3) - Classification of noise pulses as edge points; substitution

Fig 3.19 illustrates a typical edge segment in a discrete image, an ideal

edge representation, and edge representations subject to various types of errors.

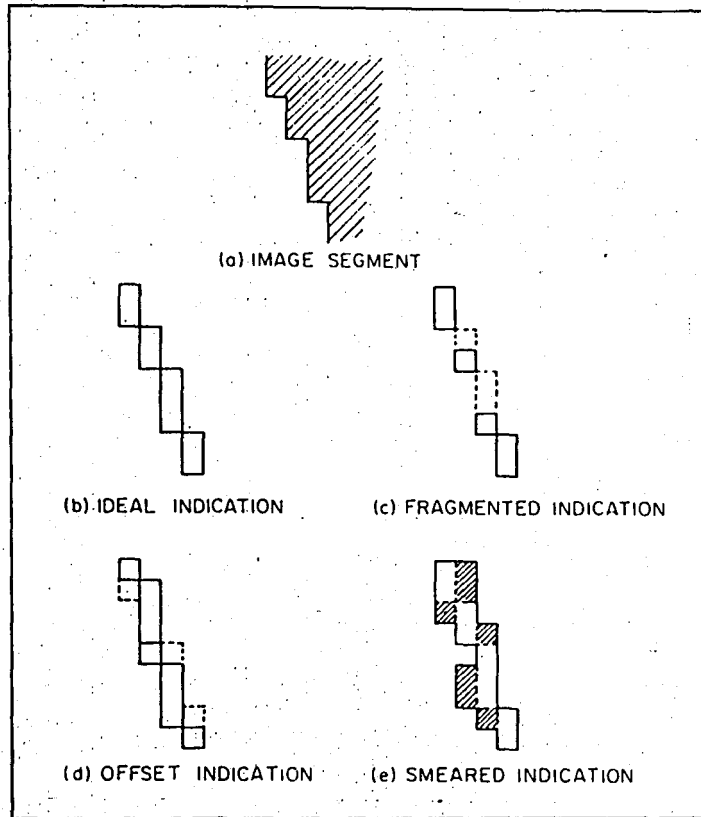


Fig 3.19 Indication of edge location by Pratt

Pratt has suggested that, a common strategy in signal detection problems, is to establish some bound on the probability of false detection resulting from noise and then attempt to maximize the probability of true signal detection. Extending this concept to edge detection simply involves the setting of the edge threshold at a level such that the probability of false detection resulting from noise alone does not exceed some desired value. The probability of true edge detection can be readily evaluated by a coincidence comparison of the edge maps of an ideal and an actual edge detector. The penalty for nonlocalized edges is somewhat more difficult to assess. Edge detectors, that provide a smeared edge location should clearly be penalized; however, credit should be given to edge detectors whose edge locations are

localized but biased by a small amount. Edge location accuracy may be assessed by the figure of merit rating factor defined by

$$R = \frac{1}{I_N} \sum_{i=1}^{I_A} \frac{1}{1+\alpha d} \quad (3.41)$$

where $I_N = \max(I_I, I_A)$ and I_I and I_A represent the number of ideal and actual edge map points, α is a scaling constant, and d is the separation distance of an actual edge point normal to a line of ideal edge points. The rating factor, R , is normalized so that $R = 1$ for a perfectly detected edge. The scaling factor, α , may be adjusted to penalize edges that are localized but offset from the true position. Normalization by the maximum of the actual and ideal number of edge points ensures a penalty for smeared or fragmented edges. As an example of performance, if $\alpha = 1/9$, the rating of a vertical detected edge offset by one pixel becomes $R = 0.9$, and a two-pixel offset gives a rating of $R = 0.69$. With $\alpha = 1/9$ a smeared edge of three-pixel width centered about the true vertical edge yields a rating of $R = 0.93$ and a five-pixel-wide smeared edge gives $R = 0.84$.

A higher rating for a smeared edge than for an offset edge is reasonable because it is possible to thin the smeared edge by post-processing.

Pratt has applied this performance evaluation methodology described above to the assessment of some of the most promising edge detection techniques, like Kirsch, Sobel, Roberts. He has used a test image consisting of a 64x64 pixel array over a 0-255 amplitude range with a vertically oriented edge of variable contrast and slope, placed at its center. Independent Gaussian noise of standard deviation has been added to the image and the resultant picture has been clipped to the maximum display limits (0-225). The signal-to-noise ratio is defined as

$$\text{SNR} = \frac{h^2}{\sigma_n^2} \quad (3.42)$$

where h is the edge height. Since the purpose of the testing is to compare the performance of various edge detection methods, for fairness it is important that each edge detector be tuned to its best capabilities. Consequently, each edge detector has been permitted to train both on random noise fields without edges and the actual test images before evaluation.

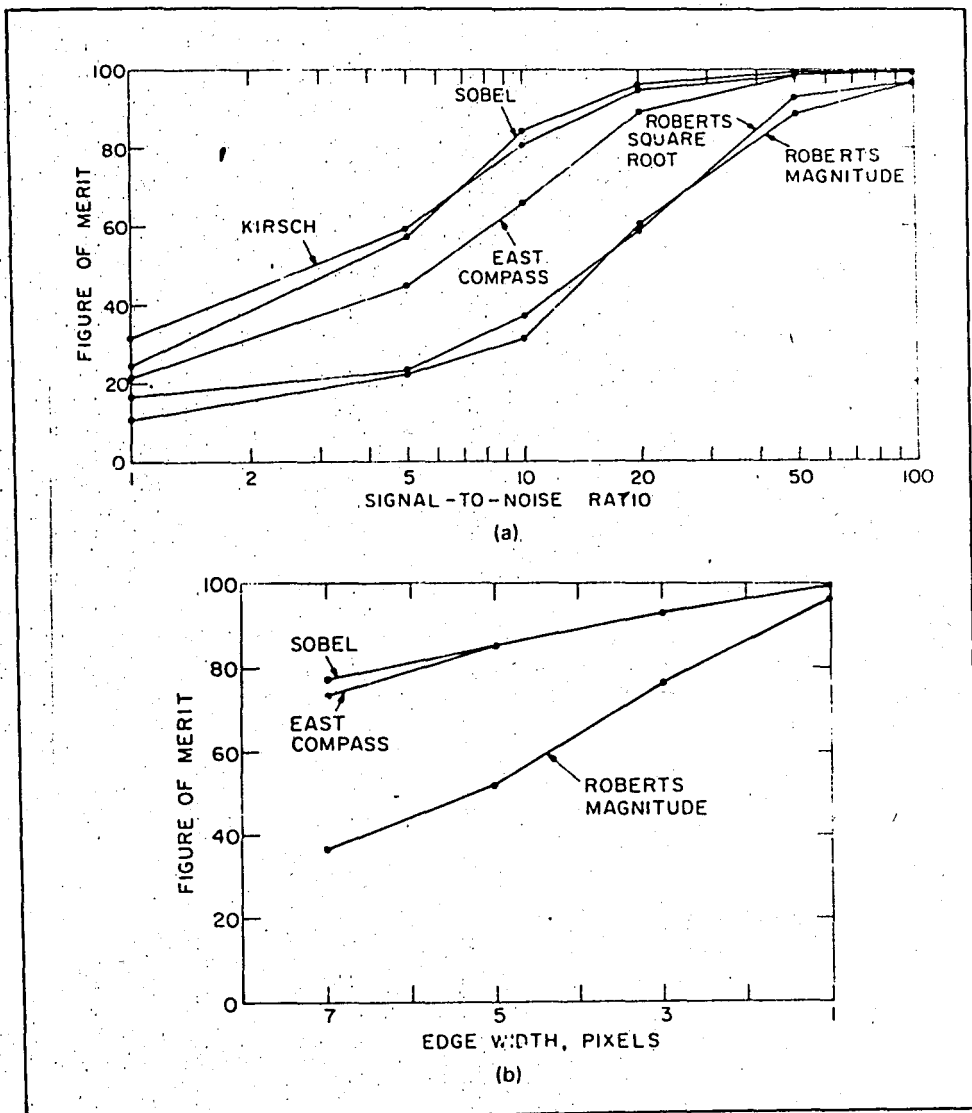


Fig 3.20 Edge location figure of merit as a function of SNR and edge width. (a) - Figure of merit versus SNR, $H = 25$, $W = 1$
 (b) - Figure of merit versus edge width, $H = 25$, $\text{SNR} = 100$

For each edge detector the threshold parameter has been set to achieve the maximum figure of merit subject to the maximum allowable false detection rate.

Fig 3.20.a contains a plot of the figure of merit as a function of signal-to-noise ratio for several edge detectors with $\alpha = 1/9$. The figure of merit is also plotted in Fig 3.20.b as a function of edge width. As one can see from Fig 3.20, the figure of merit is low for contrast, wide noisy edges, and high in the opposite case.

3.4.2 QUANTITATIVE EVALUATION OF EDGE DETECTION SCHEMES BY J.R.FRAN AND E.S.DEUTSCH

According to J.R.Fram and E.S.Deutsch, [86], characteristics of edge detection schemes which should be investigated for comparative purposes include the following: edge orientation biases, edge detection in the presence of noise, range in scale of edge detectability, ability to detect blurred edges, ability to detect curved edges, ability to extract an edge in the presence of other edges and computer speed and storage requirements.

They have investigated the performance of edge detectors in the presence of noise, as this is important in a wide range of applications and can be isolated relatively easily from other characteristics.

The approach followed by Fram and Deutsch has one principle feature that the comparative results must be quantitative. They have attempted to find reproducible numbers corresponding to the parameters which accurately

reflect the edge detection performance in the presence of noise. And this is accomplished with artificial pictures.

The edge detection schemes due to Hueckel, Macleod and Rosenfeld have been taken for comparative purposes.

A set of images was considered for which both the noise and the edge signal could be characterized numerically. The edge signal of a picture is parametrized by establishing two regions with different mean grey levels so that the regions could be said to be separated by an edge. The edge signal's strength could thus be parametrized as the difference in mean grey levels of the two regions. The noise in these regions could be given by the variance in their grey level and it was decided to keep this variance largely independent of position. Approximately ten pictures were generated for each such contrast.

These test images consisted of 36×36 matrices each of which was divided into three zones as shown in Fig 3.21. Picture points in zones 1 and 2 were assigned grey levels which were selected randomly from Gaussian distribution of means g_1 and g_2 respectively and $\sigma = 24$. These distributions were truncated at 0 and 63, respectively, the minimum and maximum grey level used. Elements in zone three were assigned values in essentially the same manner, however, the mean of the grey level distribution for each column was obtained by interpolating between zones 1 and 2.

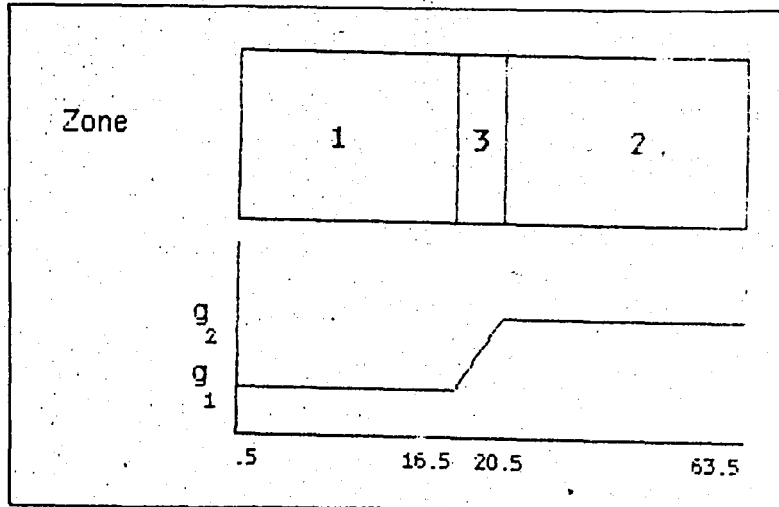


Fig 3.21 The lay-out of a test image. The square indicates the partitioning of the three zones of the test image, and the graph indicates their mean grey levels.

Specifically, the truncation points of the Gaussian distributions were set at -0.5 and 63.5 ; this resulted in reducing the variance of the distributions pushing their means towards 32 .

The quantification of edge detector performance was divided into two steps:

- (1) - The edge detector output was put into a standard form by a procedure which took into account its various characteristics.
- (2) - From the standard form, two parameters reflective of the performance of the edge detector were calculated. This calculation was done in identical manner for all edge detectors.

The standard form chosen was a binary plane in which a 1 denoted that the corresponding point was considered an edge point and a 0 meant otherwise. The size of this plane was the output domain of the edge detector. Hueckel, Macleod and Rosenfeld edge detectors evaluated here all produced quantities

which could be considered as edge weights for every point of the output domain. These quantities were rounded off to the nearest integer for convenience of display and storage. They were then thresholded to produce approximately the number of points n_{fiii} , expected for a well found edge. This number was chosen for each edge detector on the basis of an inspection of a small sample of outputs from the test images so as to optimize the results.

Given n_{fiii} ; the method of deciding the above threshold for each picture was as follows. For each edge detector output, one can define a monotonically decreasing function $n(t)$ equal to the number of points whose edge weights are greater than or equal to t . The threshold used then is defined by

$$T = \{ t \mid n(t) > n_{fiii} \} \quad (3.43)$$

The number of points of edge weight near threshold (T) was always much less than n_{fiii} . For this reason, it was thought that rounding off the edge weights did not greatly influence the results. For every picture of the sample set, two parameters are calculated from the standardized form of output discussed earlier. These may be conceptualized in terms of the following model. It is assumed that each 1 within the binary plane results from either one or possibly both of the two sources, noise or signal. It is further assumed that the 1's derived from the noise are randomly distributed in the whole picture with constant probability, whereas the 1's resulting from signal are restricted to a small subset of points within the plane, which contains the edge. If the position of the 1's generated by signal and the 1's generated by noise had been known, one could form the standard output binary plane by simply OR'ing these two sets of 1's together.

The first parameter P_1 may be defined on the basis of the above model as the maximum likelihood estimate (MLE) of the ratio of the total number of signal 1's divided by the sum of the number of noise 1's plus the number of signal 1's. The number of noise 1's is normalized to correspond to a standard number of columns in the binary plane making this ratio independent of the size of the output domain of the edge detector. The second parameter, P_2 , was visualized as follows. A row of the edge region is defined as "covered" if it contains at least one 1. (In this study the edges of the sample are all vertical). For this purpose, all rows of the edge which are "covered" by noise 1's regardless of whether they are also covered by signal 1's are disregarded; P_2 is then set equal to the fraction of remaining edge rows "covered" by signal 1's. It thus provides a measure of the distribution of the signal over the length of the edge.

The above two parameters have the following model independent properties:

- (1) - Should the 1's on the binary plane be distributed randomly with constant probability, the most probable values of both P_1 and P_2 are 0;
- (2) - Should all the 1's of the binary plane fall within the edge region, then $P_1=1$;
- (3) - Should every row of the edge be "covered", then $P_2=1$.

If it is assumed that the above model is correct, then defining the edge region larger than necessary will not affect the expectation values of the two parameters P_1 and P_2 . It will however, decrease the accuracy to which they may be determined. A further consideration was that a likely failing of the model is for signal 1's to be shifted away from the edge.

In the study of Fram and Deutch [86] it was noted that the two parameters calculated may not be expected to reflect the degree of success in finding the edge for every picture. Rather, when averaged over many of a given class of pictures, they should give a good indication of the performance of the edge detector with that set. Fig 3.22 which is taken from the study of Fram and Deutch, gives a pictorial display of the computation of the edge detection performance parameters for some typical images.


























HUECKEL					
	.11 .09	.10 .21	.62 .49	.57 .60	.65 .50
MACLEOD (LARGE)					
	.26 .28	1.00 .96	.87 .68	1.00 1.00	1.00 1.00
MACLEOD (SMALL)					
	.22 .49	.47 .84	.60 .61	.82 1.00	.90 1.00
ROSENFELD DIFF					
	.20 .14	.71 .94	.66 .80	.98 1.00	1.00 1.00
ROSENFELD DIFF & EDGE					
	.19 .44	.68 1.00	.72 1.00	1.00 1.00	.84 1.00

Fig 3.22 A pictorial display of the steps taken in computing the edge detection performance parameters.

During the comparison of the speed of the three schemes tested, it was noted that the method suggested by Hueckel seems appropriate. In the case of the schemes of Macleod and Rosenfeld, the speed is a function of the picture size and it approaches a minimum as the picture size increases. Another feature of these two schemes is that they do not cover a complete range in edge orientations as does the operator of Hueckel. Their performance is best for

edges of an optimum orientation and they cannot be expected to function, at all the edges perpendicular to this.

Economies were introduced into each of these methods by various means:

- (1) - The use of convolution with Fast Fourier Transforms in Macleod's scheme is advantageous.
- (2) - Parallel processing could reduce the computation time of all three schemes but most effectively that of Rosenfeld;
- (3) - The entire picture does not necessarily have to be examined in order to find its edges. In particular, the additional information, which Hueckel's operator supplies, could facilitate economies.
- (4) - Optical preprocessing could be beneficially employed by all schemes but most of all by that of Macleod where the speed would then be reduced to one operation per picture point for each orientation

The results of evaluating the performance of the edge detectors, described above, are given by the graphs of Fig 3.23 and table 3.1

		FIGURES OF PERFORMANCE FOR THE DETECTOR SCHEMES									
EDGE DETECTOR	PARAMETER NUMBER	NOMINAL CONTRAST									
		3	6	9	12	15	18	21	24	27	30
HUECKEL	1	.00±.03	.13±.07	.03±.06	.21±.08	.23±.11	.26±.09	.35±.09	.66±.07	.52±.07	.67±.07
	2	.01±.04	.10±.08	.05±.05	.23±.07	.20±.10	.27±.09	.34±.07	.62±.06	.67±.08	.66±.04
MACLEOD (LARGE)	1	.06±.09	.37±.40	.54±.02	.80±.06	.68±.03	.90±.02	.95±.02	.99±.03	.95±.02	.97±.02
	2	.07±.10	.27±.19	.50±.10	.73±.07	.81±.07	.82±.04	.90±.03	.96±.03	.91±.04	.96±.04
MACLEOD (SMALL)	1	.01±.04	.16±.04	.19±.03	.37±.03	.52±.04	.59±.04	.73±.04	.86±.03	.87±.03	.91±.03
	2	.20±.11	.26±.12	.31±.07	.54±.06	.74±.06	.82±.06	.90±.03	.98±.02	.98±.01	.99±.01
ROSENFELD DIFF ONLY	1	.06±.04	.23±.05	.39±.05	.63±.05	.71±.03	.78±.03	.80±.04	.97±.01	.93±.02	.97±.02
	2	.07±.13	.27±.17	.32±.45	.74±.06	.81±.04	.97±.03	.86±.04	.96±.02	.95±.03	.99±.01
ROSENFELD DIFF & EDGE	1	.03±.05	.16±.04	.24±.05	.47±.04	.55±.05	.66±.04	.69±.04	.74±.06	.82±.05	.81±.03
	2	.07±.13	.38±.07	.47±.11	.93±.05	.90±.06	.85±.03	.93±.03	.98±.02	.99±.01	1.00±.00

Table 3.1 Evaluation results of the performances for the detector schemes

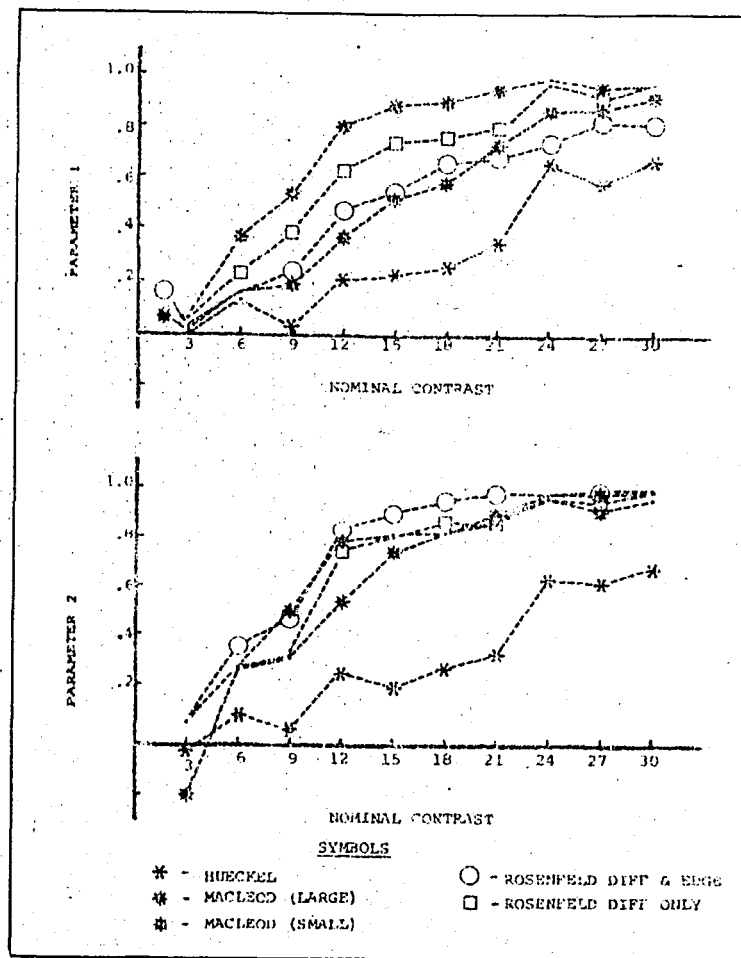


Fig 3.23 The edge detection performance parameters as a function of nominal contrast

It is readily seen from Fig 3.23 that of the two parameters evaluated, parameter 1, which indicates the ratio of the edge detector's signal to its noise consistently, shows more pronounced differences in the performance of the various edge detecting programs tested, than parameter 2, which gives a measure of the fraction of the edge covered by the signal.

A clear result indicated in Fig 3.23 is that Hueckel's operator as implemented in the study does not perform as well on the sample images as the other two schemes.

In the second work [88] of Fram and Deutsch, the performance of these three

edge detection schemes was studied at skew orientations. The question of orientation biases was investigated either by rotating the test images or by changing the optimum orientation of the edge detectors. Four sets of test images were generated to the same specifications. These were rotated by 15° , 30° , 45° , 60° .

Thresholding the edge detector output was performed after it was rotated back so that the edge region was vertical. As was done for vertical edges, the threshold was determined for each test image to permit enough points to pass and to fill the edge region. A difference between the rotated-edge output and the vertical-edge output was that the first one contained a greater proportion of points in the edge region than did the second. As in the first study, the number of points in the threshold determination which were considered to fill the edge region, n_{fill} , varied from detector to detector. To generalize the definition of parameter 1, it was necessary only to re-express it in such a manner that it no longer was implicitly assumed that the domain was rectangular. Parameter 2 was dependent on the edge region being rectangular, but the "rotated out" corners in some cases extended into the edge region. When this occurred, the rows of the edge region which were missing points were excluded from the computation of parameter 2.

The results of the tests described above are plotted in Fig 3.24. Within statistical fluctuations, the changes in the rated performance of the edge detection schemes with orientation and contrast-to-noise ratios is consistent with what one would expect on general principles. The performance of Hueckel's operator is roughly independent of the orientation of the test edges, and the performance of the other schemes fall off as the orientation

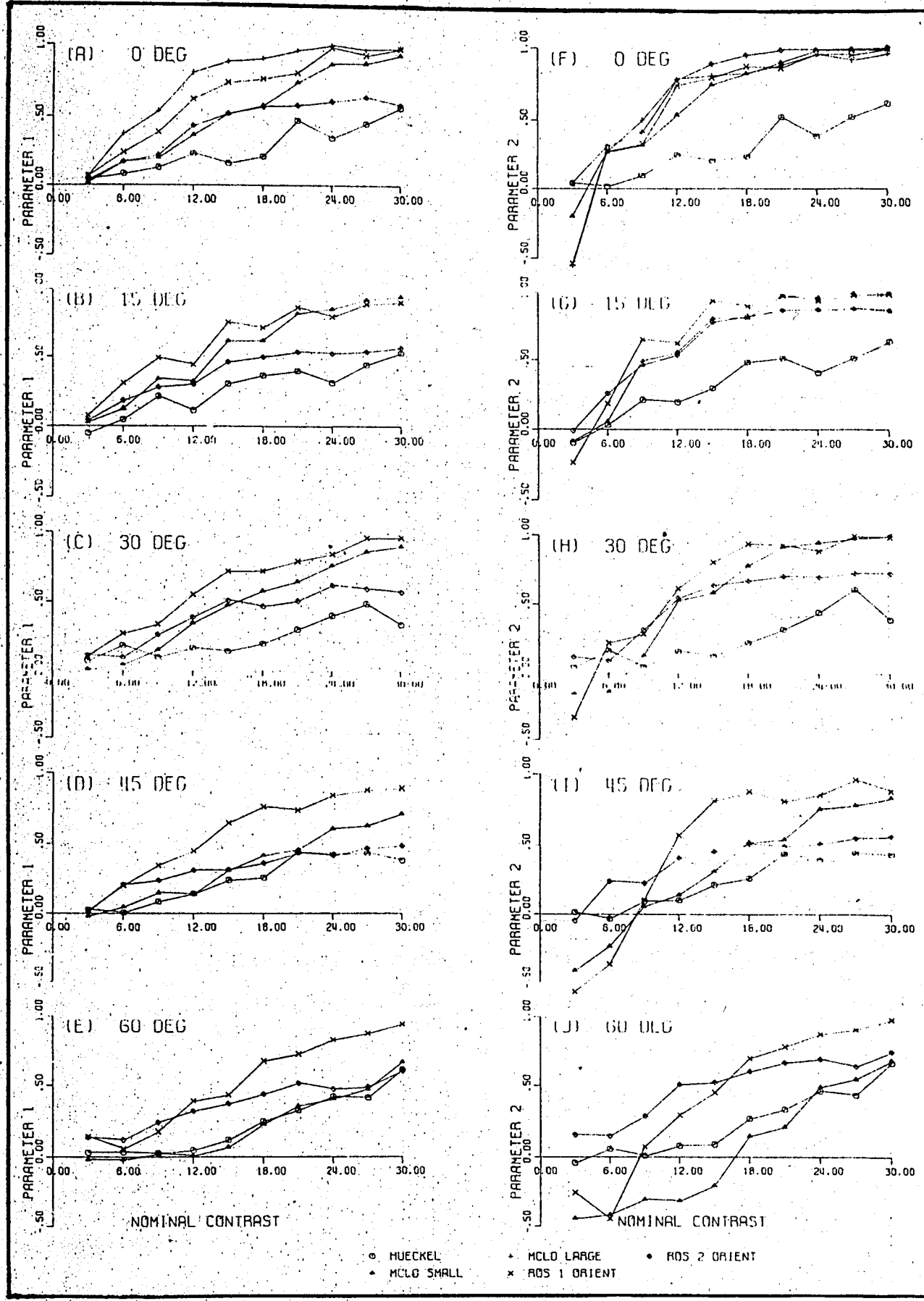


Fig 3.24 Behaviour of the edge detection schemes in terms of the variation of parameters 1 and 2 with changes in nominal contrast taken at various orientations.

of the test images becomes more apart from the ideal orientations. This fall-off is most pronounced at the intermediate contrast-to-noise-ratio. If an edge is very distinct, then it can be detected over a wide range of orientations. If the edge is not very distinct, then the range of orientations in which it can be detected is smaller.

It can be seen in Fig 3.24 that the method of Macleod as programmed by Fram and Deutsch is much more biased with respect to the orientation of the test edges than the method due to Rosenfeld. A likely explanation for this is the shape of the mask used in implementing Macleod's scheme. The selection of the square-shaped mask for Macleod's scheme, was not an inherent feature of the method, but an arbitrary choice of the investigators.

3.4.3 QUALITATIVE COMPARISON OF DIRECTIONAL LAPLACIAN MASKS AND SOBEL AND PREWITT OPERATORS BY E.ALPASLAN

Erhan Alpaslan [87] has described the directional Laplacian masks and suggested some new algorithms in his paper. He has taken an actual image and applied three different edge detection methods, Prewitt, Sobel and the directional Laplacian masks by making a computer simulation. He has prepared his simulation with PDP 11/45. The actual picture is sampled to 128x128 pixels each of which have a weight according to its grey level.

After construction of the gradient image, he applied both fixed and locally adaptive threshold to the gradient image. He has reported in his study that the directional Laplacian masks served for edge detection as well as Prewitt and Sobel operators did.

IV COMPUTER SIMULATION RESULTS OF EDGE DETECTION SCHEMES AND FURTHER RESEARCH ON THEIR COMPARISON

4.1 INTRODUCTION

In this chapter, some local edge detection techniques are studied. The first one is the method which uses Kirsch masks. After preprocessing of images with the directional Kirsch masks, the enhanced image is put into a labeling stage which is done by both adaptive and fixed thresholding. The second method uses the Sobel operator. After preprocessing with the Sobel operator, again both the adaptive and fixed thresholding are applied to the enhanced image. Then a second level thresholding is used in order to get sharper edges. The third one uses Rosenfeld's product of difference equations in order to enhance images. After the enhancement operation, fixed thresholding is applied to make a decision whether a given pixel is an edge or not.

The above three methods is chosen for the subject of this study, among various different edge detection schemes, because each scheme has some typical properties and has different algorithm which was discussed in Sec.3.3.

The methods mentioned above are not tested on actual images, because of some limitations of the computer used. The memory capacity and the operation time of the computer is the basic handicap. Instead of actual images, some test images with vertical, diagonal and circular edges are generated and these methods were then applied on them. Evaluation and comparison of the methods

mentioned above is done depending on some qualitative and quantitative performance criteria.

A package program is developed consisting of three edge detection techniques, various thresholding techniques and formulations which are used for evaluation and comparison processes. In section 4.2 test images which are generated by the computer will be explained in 4.3. Three edge detection schemes will be discussed in more detail. In section 4.4 performance criteria will be explained. In 4.5 depending on each performance criteria evaluation of each method and their comparison will be done.

4.2 TEST IMAGES

The study on edge detection schemes and evaluation processes is done with a set of test images generated by the computer. These are the pictures in which both the noise and the edge signal can be characterized numerically and in which these are to be the only variables allowed to change. The edge signal of a picture is parametrized by establishing two regions with different mean grey levels so that the regions can be said to be separated by an edge. Therefore, the edge signal's strength can be parametrized as the difference in mean grey levels of the two regions. The noise in these regions is given by the random variances in their grey levels.

The maximum dimension of the test images is 32x32 which is set by the memory limitation of the computer. But the pictures which are used for analysis are of size 20x20, because of the long operation time (run time of the computer) of the processing. The test images consist of 20x20 matrices and may have three different shapes of edges, namely vertical, diagonal and circular. They

all are divided into three zones as shown in Fig(4.1). Picture points in zones 1 and 3 are assigned grey levels which are selected randomly from Gaussian distribution of means D_1 and D_3 , respectively. These distributions are truncated at 0 and 280, respectively, the minimum and maximum grey levels used. Elements in zone 2 are assigned values, especially as the averages of means of the grey level distributions of zones 1 and 3.

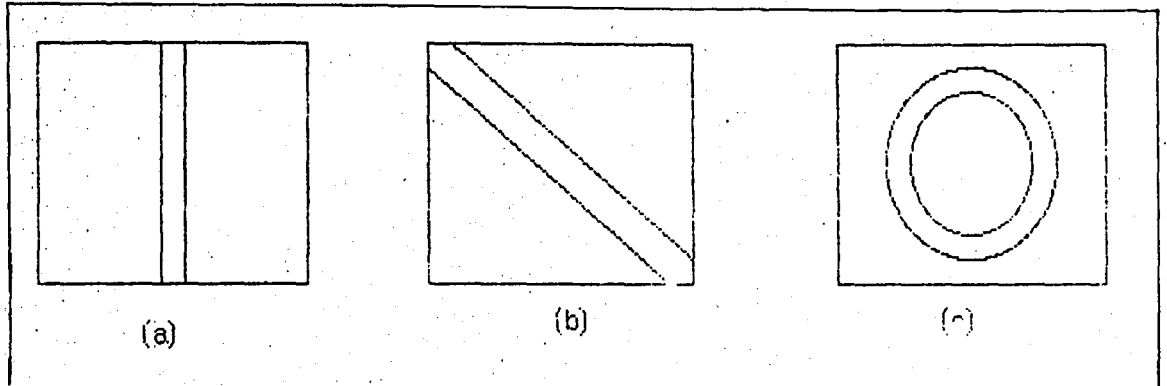


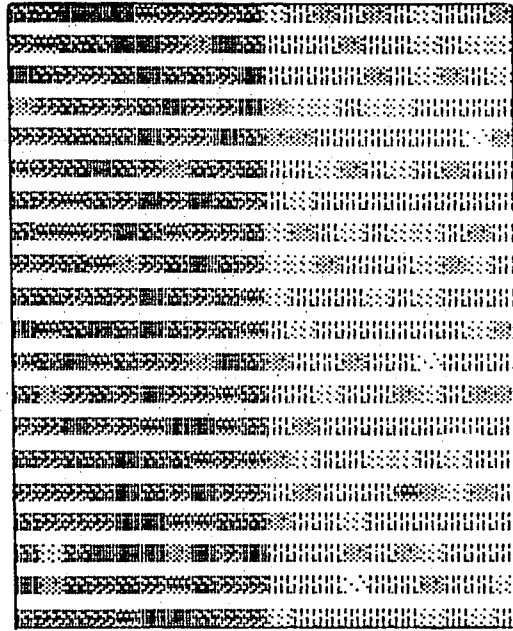
Fig.4.1 The regions of three different test images

Zone 2 is neglected while taking most of the analysis. It is taken into consideration when it is desired to see the behaviour due to blurred edges.

The program is prepared such that the grey levels of the two different regions of the edges can be changed each time the user wants to do so. Fig 4.2 and 4.3 shows the test edges with different contrasts. The contrast value can be defined by the equation

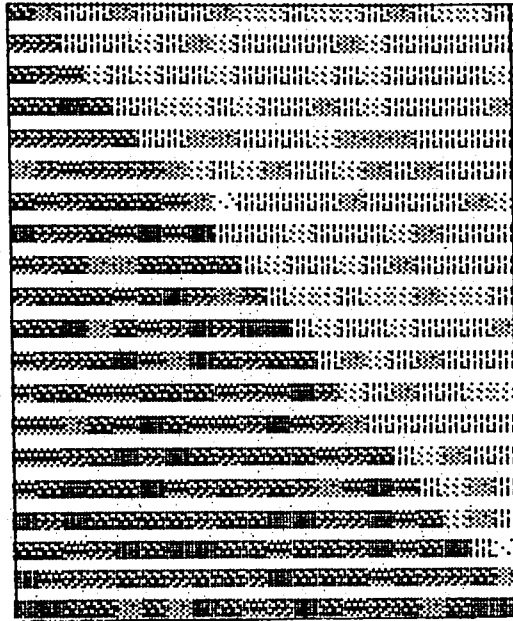
$$C = D_1 - D_3 \quad (4.1)$$

where D_1 and D_3 are the mean values of the Gaussian distribution in zones 1 and 3 respectively. For the vertical image, zone 2 can be divided into two regions with different means of grey levels if it is desired.



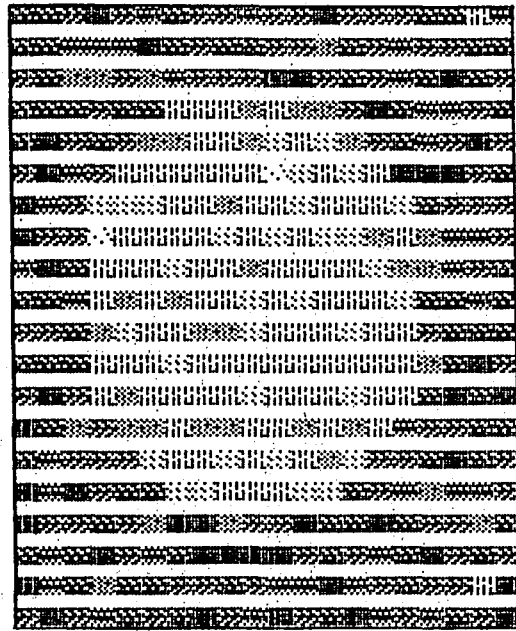
7	7	8	8	8	5	6	6	6	7	2	3	4	3	4	3	2	3	3	4
6	5	7	7	8	8	6	4	8	7	3	3	3	4	3	3	2	3	2	2
8	7	6	6	7	8	7	7	6	8	3	3	3	3	4	3	2	4	3	2
4	6	7	6	6	7	8	6	6	8	4	2	2	3	2	2	3	3	3	3
6	6	7	7	7	8	6	6	8	7	4	4	3	3	3	3	3	3	1	4
5	6	7	8	7	6	4	7	6	7	3	3	2	4	3	2	3	4	3	3
7	6	5	7	6	8	6	8	7	6	3	2	3	3	3	3	3	3	3	3
7	5	5	6	8	7	5	6	6	7	2	4	3	2	3	2	2	3	4	3
6	6	7	5	4	6	7	8	7	6	2	2	4	3	3	3	2	4	2	3
7	7	6	7	6	8	7	6	7	5	2	3	3	3	2	3	2	3	3	3
8	5	7	8	7	8	7	6	7	5	3	2	3	3	3	3	3	3	2	4
5	7	8	5	7	6	6	4	8	7	4	3	3	4	3	3	1	3	3	3
7	4	6	7	6	8	6	6	5	7	3	3	2	3	3	4	2	3	4	4
6	6	8	6	6	5	8	8	8	5	7	3	4	3	3	3	3	3	3	3
7	6	7	7	8	7	7	5	6	5	4	2	3	3	2	2	3	2	3	3
6	6	6	7	8	7	6	9	6	6	3	4	3	3	3	5	4	2	4	3
7	6	6	6	9	8	5	5	7	7	4	3	3	2	3	3	3	3	3	3
7	2	7	9	8	8	4	8	6	10	3	3	3	4	3	4	2	3	3	3
8	4	7	6	7	6	5	7	6	6	3	3	3	1	3	3	4	3	3	2
7	6	7	6	5	8	8	7	6	6	2	3	3	3	3	3	2	3	2	3

Fig 4.2 Vertical, diagonal and circular test images with assigned characters and their intensity values with a contrast of $C=5$ a) Vertical test image



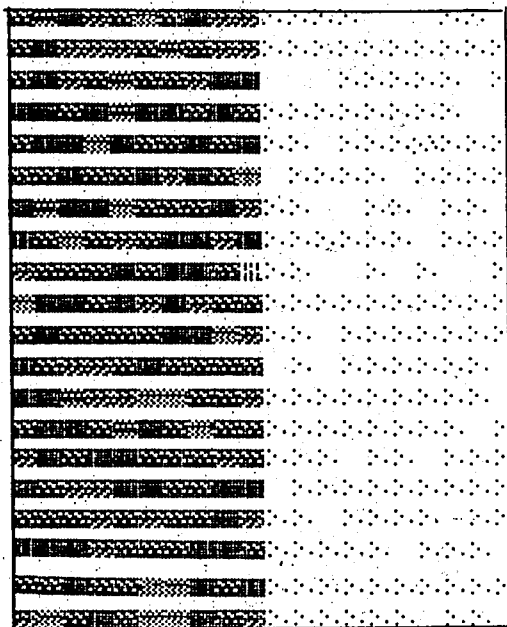
7	4	3	3	4	3	3	3	4	2	2	3	2	2	3	4	3	2	2	3
6	6	3	3	3	2	3	4	2	3	3	3	3	4	2	3	3	3	3	3
7	6	5	2	3	2	3	3	2	3	3	3	2	3	2	3	3	3	3	2
7	7	8	7	3	3	2	2	3	2	3	3	4	3	2	3	3	3	3	4
6	6	6	6	7	3	3	4	4	3	3	3	2	4	4	4	3	3	3	3
4	6	5	6	6	6	4	2	3	2	4	3	3	2	4	3	4	3	3	3
7	5	6	7	7	7	5	4	1	3	3	3	3	4	3	3	3	3	4	2
9	6	6	7	5	8	5	8	3	3	3	2	3	3	3	2	4	3	3	3
5	6	7	4	4	7	7	7	7	3	2	3	3	2	3	4	3	3	2	3
6	7	7	7	5	7	8	6	4	6	3	2	2	3	2	2	4	2	2	3
7	7	8	4	7	5	6	8	6	8	9	3	2	3	3	2	3	3	3	4
5	6	6	7	8	5	4	8	7	6	7	7	3	4	2	3	4	3	3	3
5	7	7	5	5	7	7	7	5	6	5	8	6	2	3	4	3	3	2	2
5	5	4	7	5	8	7	5	5	6	8	5	6	4	3	3	3	3	3	3
5	5	6	7	10	6	8	7	6	7	7	7	5	6	7	3	2	4	3	3
5	7	8	7	7	8	5	6	7	6	7	6	4	5	8	5	3	2	4	3
8	6	8	7	7	7	7	6	7	7	8	8	6	6	8	5	7	2	4	3
7	7	5	6	8	7	8	9	7	7	5	7	7	6	8	5	7	8	3	1
8	5	6	6	7	7	6	7	7	6	8	7	7	7	5	7	6	6	7	4
8	8	7	7	4	7	4	9	7	5	6	8	7	5	6	7	4	7	8	8

Fig 4.2 b) Diagonal test image with C=5



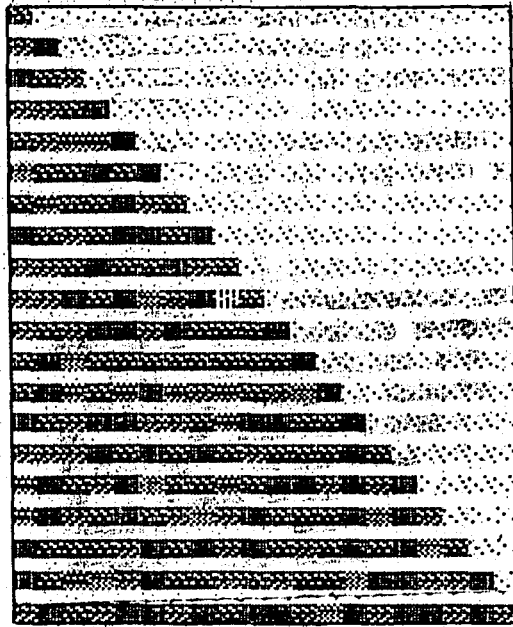
7	7	6	8	6	5	7	6	6	5	6	6	8	6	6	6	7	7	3	5
7	7	5	5	5	8	7	6	7	7	6	6	4	7	6	5	6	6	7	7
6	7	4	4	6	4	5	6	6	6	8	8	6	7	6	5	7	8	7	6
7	7	7	6	7	7	3	3	3	4	3	4	4	6	8	7	5	5	6	7
7	8	6	7	6	4	4	3	3	4	2	3	2	4	6	7	5	6	8	6
6	8	5	6	3	3	3	3	3	3	1	2	3	2	3	8	8	8	6	7
8	5	6	2	2	2	3	3	4	3	3	2	3	3	3	2	7	6	6	6
9	6	6	1	3	3	3	3	2	3	2	3	2	2	4	3	4	5	5	6
5	8	7	3	3	3	2	3	3	4	3	3	3	3	2	4	4	5	6	7
7	7	5	3	4	3	4	3	3	2	3	2	3	3	3	2	7	7	5	7
6	6	7	4	2	3	3	4	4	2	3	3	3	2	3	3	6	7	7	7
7	7	7	3	3	3	2	3	3	3	3	3	3	3	3	3	4	7	8	6
6	8	6	3	4	3	3	3	3	2	3	3	3	2	3	3	7	9	7	8
8	7	4	6	4	4	3	4	4	3	3	4	3	4	3	5	6	6	7	7
7	5	6	6	6	2	3	3	2	3	2	3	4	2	6	6	7	8	7	6
8	5	8	6	7	7	2	2	3	3	3	2	2	7	6	5	4	5	5	6
9	6	6	7	6	4	8	8	4	6	6	8	7	7	8	7	6	6	4	7
7	5	7	8	6	5	7	8	9	10	8	6	7	6	5	7	8	6	7	6
8	5	7	4	7	7	6	6	7	6	5	5	8	5	6	7	6	6	3	8
6	8	6	5	7	6	7	8	6	5	8	6	7	8	5	6	5	7	6	8

Fig 4.2 c) Circular test image with $C=5$



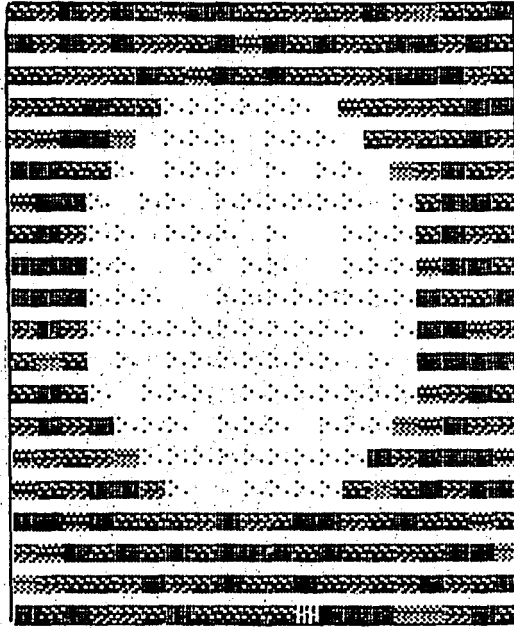
53	43	62	47	51	36	56	73	48	48	3	3	3	4	2	2	2	3	3	2
51	61	50	48	48	55	43	57	48	46	3	4	4	3	3	4	4	3	3	4
59	83	50	51	44	54	58	47	61	66	2	2	2	3	3	3	3	3	2	3
71	84	54	66	43	65	60	51	65	56	4	3	3	4	4	3	3	3	2	2
57	73	62	37	71	57	59	75	54	63	3	3	2	4	3	3	5	3	3	3
59	53	60	56	51	61	50	68	57	38	2	3	3	3	4	2	3	4	3	4
61	44	62	68	40	57	55	55	61	47	4	3	2	2	3	3	2	3	3	2
61	52	38	55	46	57	66	61	47	64	3	3	3	3	4	3	2	4	3	3
50	55	51	57	61	53	68	70	56	24	3	3	2	2	3	2	3	2	2	3
38	62	74	52	65	49	67	48	58	53	3	3	3	3	3	3	4	4	3	4
54	73	54	56	58	53	60	69	38	46	4	3	2	4	3	4	3	3	3	4
74	52	50	46	56	68	54	58	59	52	2	4	2	3	3	4	3	3	3	2
65	70	44	49	50	36	36	53	57	50	3	4	3	3	3	3	3	3	3	2
58	62	62	56	43	72	56	34	59	52	3	3	3	4	3	3	4	3	2	3
50	64	59	72	64	58	55	51	49	57	3	3	3	1	3	3	2	3	3	3
68	53	50	49	64	63	59	53	63	66	2	4	3	3	3	4	4	4	4	3
54	53	51	49	56	46	54	59	63	49	3	3	2	3	3	4	3	2	3	3
62	69	72	46	59	52	58	61	64	55	3	3	3	4	4	2	4	3	3	2
58	52	69	56	58	32	38	62	59	67	3	3	3	4	4	3	3	3	4	4
48	40	51	60	52	39	34	75	56	45	4	3	4	3	4	4	2	3	3	2

Fig 4.3 Vertical, diagonal and circular test images with assigned characters and their intensity values with contrast of 30
a) Vertical test image with $C=30$



57	3	3	4	4	5	4	6	6	4	6	4	5	6	3	8	4	4	4	5
46	53	7	7	5	4	4	6	4	7	5	5	4	3	6	4	3	7	3	4
66	54	50	6	5	3	4	6	6	6	6	4	5	3	3	5	5	4	6	3
51	50	54	69	4	6	7	6	4	3	5	2	5	6	6	4	3	3	7	7
54	50	45	43	73	6	6	4	5	7	5	5	5	3	6	5	5	5	3	4
37	55	54	69	57	65	4	4	5	6	4	5	3	6	6	5	6	7	2	3
56	43	60	61	63	50	56	6	8	4	6	6	3	5	4	7	7	3	5	3
77	60	48	54	76	70	59	64	6	5	7	6	5	5	3	4	3	6	4	5
51	49	61	65	57	57	62	48	57	6	2	5	6	6	3	3	5	5	6	4
49	50	69	57	65	41	51	66	27	53	4	5	4	6	6	6	4	4	7	4
49	56	47	78	64	51	77	57	56	53	67	4	3	3	6	2	5	7	5	7
59	78	37	55	57	61	57	57	55	61	58	70	5	3	4	8	4	7	6	4
56	68	42	56	43	66	42	51	45	60	49	41	64	7	7	3	3	7	7	7
71	58	51	76	68	52	48	60	44	64	65	56	59	74	3	5	5	7	7	4
49	48	49	69	59	64	54	73	53	49	60	59	54	65	55	6	6	6	5	5
43	71	54	50	64	35	61	53	44	53	65	71	49	63	50	73	4	6	5	7
45	67	50	60	87	53	47	38	49	65	56	55	53	67	41	72	52	6	3	3
65	53	60	55	48	64	60	69	47	69	61	52	59	62	55	71	41	53	6	4
66	58	42	38	49	64	58	57	49	57	51	60	58	33	66	74	50	52	66	7
50	71	61	51	66	68	48	67	54	67	68	47	39	63	50	62	62	48	72	47

Fig 4.3 b) Diagonal test image with C=30



55	51	60	49	75	58	42	67	69	55	57	51	55	47	60	48	31	59	57	64
66	47	63	48	78	49	51	55	77	41	63	53	60	46	54	60	70	50	67	57
59	55	48	48	59	83	54	44	66	58	61	55	54	50	51	79	85	75	49	56
50	58	52	76	59	58	3	4	3	3	4	3	2	44	55	50	49	56	63	64
48	41	71	71	38	2	4	3	3	2	4	3	4	2	52	48	54	54	60	47
60	63	55	59	3	2	3	4	3	2	3	2	4	3	2	40	51	60	52	47
42	63	60	4	2	3	3	2	4	3	3	3	3	3	2	3	53	63	65	58
54	64	47	3	3	2	3	4	3	2	3	2	2	3	3	3	52	64	51	54
65	71	62	4	3	2	2	4	2	3	3	2	2	3	4	3	43	67	70	56
62	62	61	4	3	3	2	2	4	3	3	3	4	3	3	3	65	56	55	60
49	62	48	3	3	3	3	3	3	3	4	3	3	3	2	4	62	63	43	45
55	38	52	2	3	2	3	3	3	3	3	3	3	2	3	2	63	60	62	61
58	61	56	3	2	3	3	3	2	3	3	3	4	3	4	3	42	51	64	55
45	62	46	68	3	4	3	2	4	3	3	2	3	3	4	39	44	61	46	51
41	48	59	50	40	3	3	3	3	3	3	3	4	4	63	47	69	65	61	44
42	59	45	72	63	46	4	2	2	4	3	3	4	55	33	54	69	49	62	61
62	62	42	61	57	56	59	46	62	47	59	71	78	51	57	61	53	54	41	54
51	43	61	55	60	54	60	58	64	62	61	55	71	55	58	46	55	74	69	38
40	48	57	56	46	65	51	53	63	58	55	64	53	48	59	49	62	47	57	62
60	54	65	51	49	57	70	54	59	54	56	28	66	61	63	39	39	47	62	59

Fig 4.3 c) Circular test images with $C=30$

Random numbers are generated independently from a uniform distribution on the interval (0,1) and the noise is introduced by adding one of these numbers to the density value at each pixel if it is desired to see the results of applying the three edge detection schemes on the test images.

The microprocessor which is used in this study is APPLE IIe with 64 Kbyte memory and the printer is EPSON MX-80 which can provide discrete level display of the images. APPLE has no grey level differences in its graphic mode. That's why 14 new characters, each of which has 7x8 dots, have been generated. Each has different number of dots in the 7x8 matrix according to the respective weightings of pixels of the image. Grey level distribution which is truncated at 0 - 280 is divided by 14 and for each level one character is assigned. The actual image and the gradient picture of enhanced edges are printed character by character according to the above criteria. (See Fig(4.2) and Fig(4.3)). The circular edge is used more for qualitative analysis, like ability to detect curved edges, instead of quantitative ones. For quantitative purposes the vertical and diagonal images are used.

4.3. EDGE DETECTION SCHEMES

The edge detection schemes which are considered in this section are explained in detail in section 3.3.5 (Sobel operator), 3.3.6 (Kirsch masks), 3.3.9 (Rosenfeld's difference equations), 3.3.10 (Connectivity test and Locally Adaptive Threshold from Robinson). Here, methods which are used in the computer simulation and new additions to these methods will be discussed.

A) - EDGE DETECTION BY USING DIRECTIONAL KIRSCH MASKS, THE CONNECTIVITY TEST AND THRESHOLDING

In this method eight directional Kirsch masks is used for preprocessing level. (See Fig 3.10). In order to get the gradient image, convolution of each gradient mask and the 3x3 grid surrounding a picture element is taken by using the equation:

$$GD(x,y) = \sum_{l=y-1}^{y+1} \sum_{k=x-1}^{x+1} MSK(k,l)*A(k,l) \quad (4.2)$$

where $GD(x,y)$ is the gradient value of the pixel which is in position x,y .

$MSK(k,l)$ is one of the Kirsch's masks $A(k,l)$ is the element of the image.

Eq.4.2 must be repeated for each mask. The application of the eight masks to a 3x3 grid surrounding a picture element gives the gradient magnitude and direction as $G1(x,y).....G8(x,y)$. The gradient picture is obtained by taking the maximum gradient magnitude at each point, i.e.,

$$GD(x,y) = \text{Max} \{G1(x,y).....G8(x,y)\} \quad (4.3)$$

Two different types of thresholding techniques are applied to the gradient picture in order to determine whether the gradient value is large enough to accept or reject the presence of an edge point. At the end of this procedure a threshold map is generated.

(1) - Fixed threshold, which is equal to the average intensity of the corresponding gradient image, is used as one alternative. After computation of the threshold value, the gradient value of each pixel is compared with that value. If it is above that level, a "1" appears in the position of that

pixel, and if it is not, then a "0" appears in the same position in the threshold map.

(2) - Three different adaptive thresholding techniques are tested:

a) The one which is suggested by Robinson. A locally adaptive threshold, LAT, is obtained by comparing the gradient image with a blurred version of the original image, which is obtained by convolving the image by a low-pass filter. The particular low-pass operation can be performed by the mask M_0 :

$$M_0 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (4.4)$$

Note that, in Eqn(3.39) M_0 is multiplied by a constant of 1/16. This does not hold for the case of thresholding. Locally adaptive threshold is obtained by deviding the gradient value of the pixel by the output of the low-pass filter M_0 at that pixel, i.e.

$$LAT = \frac{GD(x,y)}{\text{Output of the low-pass filter } M_0 \text{ at pixel } (x,y)} \quad (4.5)$$

b) The second choice of adaptive thresholding is obtained by applying a fixed threshold to the output of the above procedure (low-pass filtering). The fixed threshold is again the mean intensity of the gradient image.

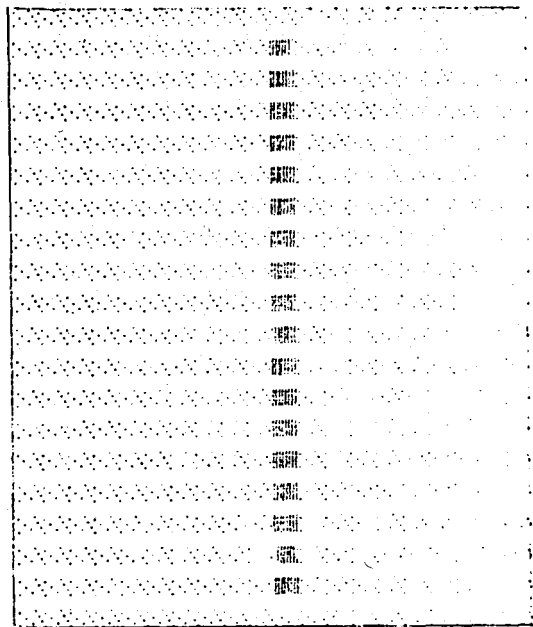
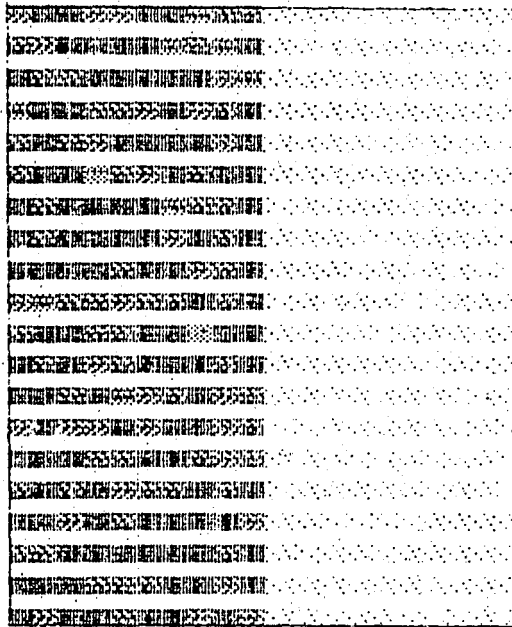
c) In the third case, the average magnitudes of pixels in a 3x3 grid, surrounding each of the pixel in the gradient image, are taken and their average intensity value is found. This value is determined as a threshold value for that pixel. If the magnitude of the pixel is higher than the threshold value, then a "1", if not a "0" appears in the position of that

pixel in the threshold map.

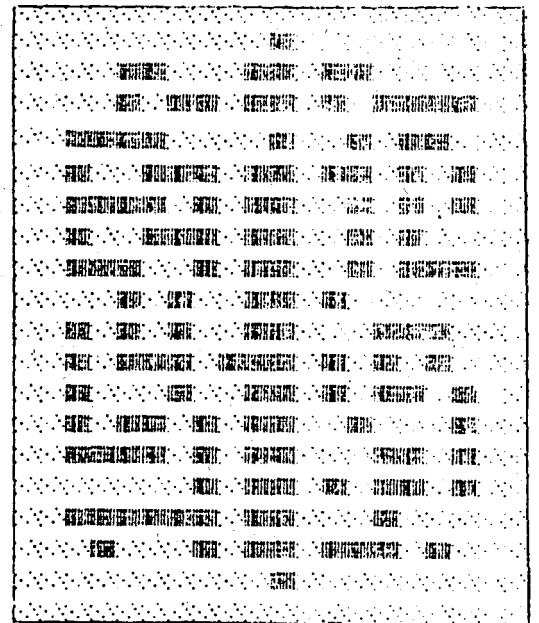
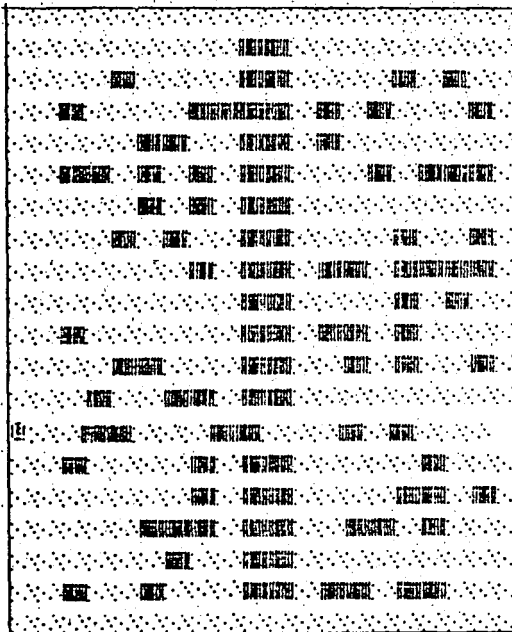
The comparison of these three thresholding methods was done at the beginning, and it was discovered that the second alternative for adaptive thresholding gives the sharpest edge, but as a disadvantage some edge points disappeared. The third choice gives many edge points in addition to the points of the main edge. In a real image, if details of the picture are required the last method can be used, but in our case, it is preferred to deal with the first method for adaptive thresholding. In the succeeding sections of this study, only the first method will be used. Examples of these three adaptive thresholding techniques are given in Fig (4.4). They are taken with a contrast value of 20 and there is no addition of noise.

Alternative combinations of thresholding techniques can be considered. For instance, after applying the third method, in order to sharpen the main edge and neglect the subedges, fixed threshold can be applied to the output of the adaptive threshold. But this investigation would exceed the purpose of this study.

The mask which yields the maximum gradient value determines the direction of the edge. The direction edge map is generated as a two-dimensional array of numbers which range between 0 and 7. The direction edge map is used to determine the local connectivity. If the direction at the center of the 3x3 grid is k ($k = 0, 1, \dots, 7$), and if the directions of the preceding and succeeding edge vectors are $k-1$, k , $k+1$, for any of the six compass directions and the directions of the other two edge vectors are $k-2$ or $k+2$, then the edges are assumed to be connected. (See Fig 3.16 for eight compass directions). A tolerance is made for any of the two preceding and succeeding



a



b-c

Fig 4.4 The outputs (edge maps) of three different methods of adaptive thresholding. a) Low-pass filter, Mo b) Local Averages + Fixed threshold c) Local averages

edge vectors in order to fill the edge region totally. Because, when the connectivity test of Robinson is applied to the Kirsch masks without such a tolerance, edge becomes sharper but many of the edge points are lost. The connectivity test is applied to the threshold map and only the 1's which are above the threshold level in the map are tested whether they are connective or not. The connectivity test can be considered as a post processing operation. The block diagram of the proposed edge detection scheme is shown in Fig.4.5

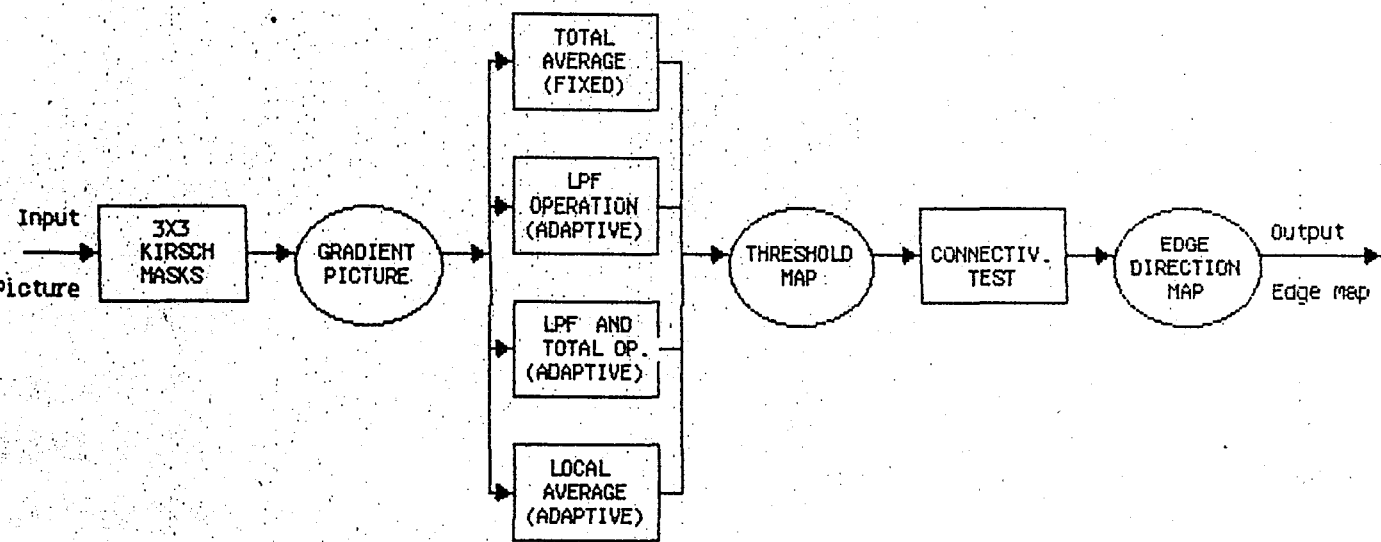
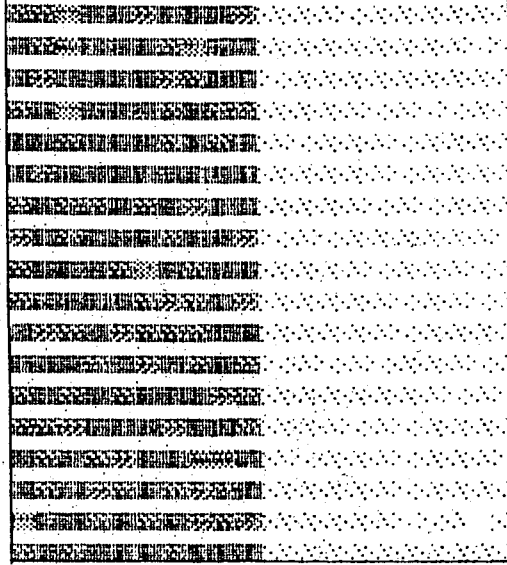


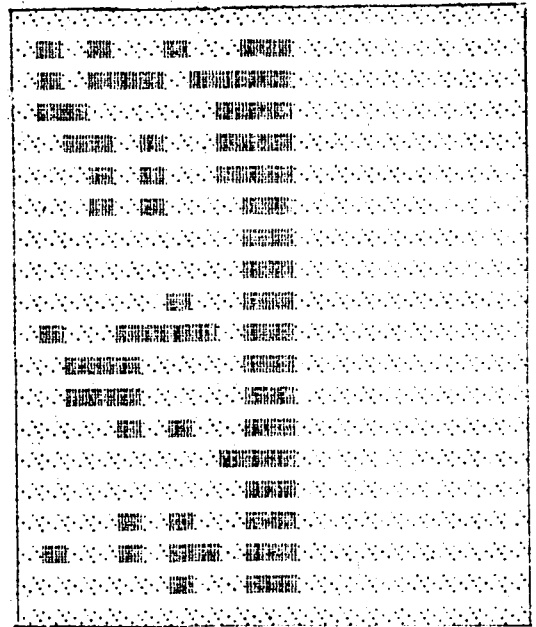
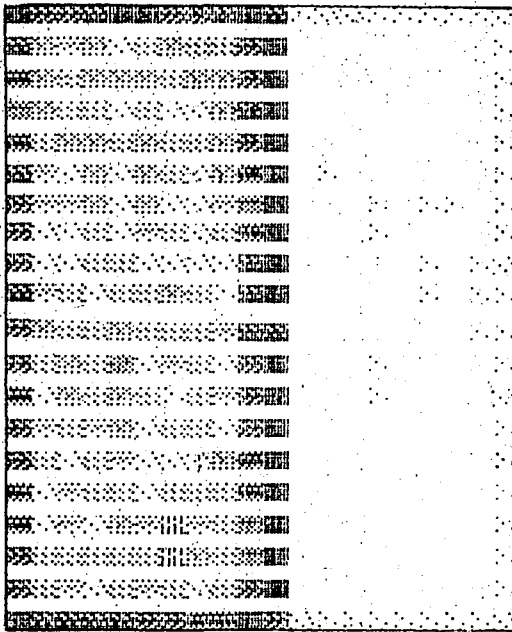
Fig.4.5 Block diagram of the Edge detection scheme using Kirsch Masks

In Fig.4.5 square blocks show the operations while circles show outputs of the preceding operations.

In Fig.4.6 the outputs of the edge detection scheme with Kirsch masks and fixed threshold which is applied to the vertical edge is shown step by step. In Fig 4.7 the application of the same scheme on a diagonal and in Fig 4.8 on a circular edge is shown. In Fig 4.9, 4.10 and 4.11 the same detection scheme with locally adaptive threshold (application of LPF-M) on vertical,

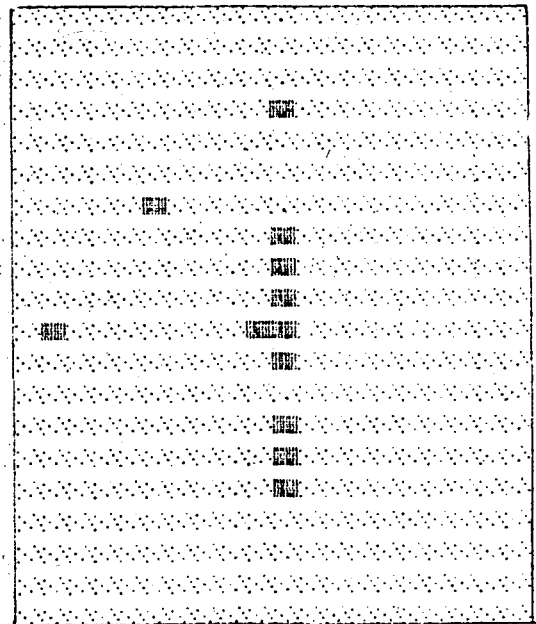


a



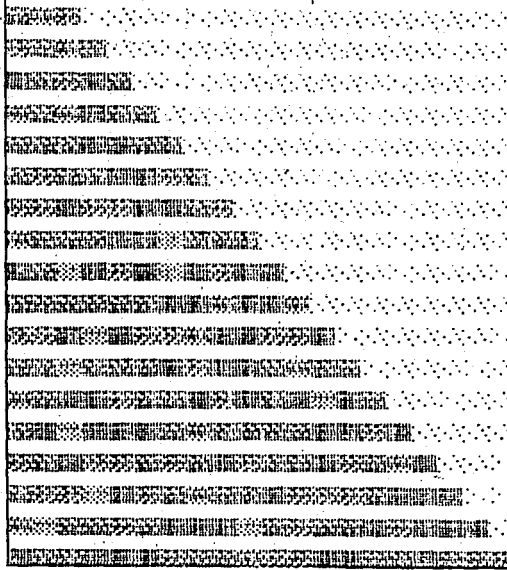
b-c

6	4	6	5	4	6	6	4	4	4	5	6	4	4	6	4	4	6	4		
8	2	1	8	2	3	8	1	3	2	3	6	1	2	1	1	1	1	8	2	
8	3	1	1	1	2	1	1	8	1	2	3	1	1	2	6	4	4	4	3	2
6	3	7	6	3	4	1	1	6	4	3	2	5	4	4	1	1	1	1	2	2
8	3	7	6	5	4	3	6	8	2	3	1	6	4	3	1	1	1	2	2	4
6	4	7	6	4	4	3	8	7	2	3	7	8	1	2	1	1	5	1	2	2
7	2	7	7	1	3	3	5	6	4	3	3	7	2	3	6	6	4	5	4	2
7	2	3	8	2	2	4	3	7	2	3	3	6	4	3	7	7	2	3	2	2
7	1	7	6	5	3	1	2	6	3	3	2	7	1	2	1	7	1	3	2	2
7	6	6	6	3	2	2	7	5	3	3	1	1	1	3	6	7	4	3	2	2
7	6	6	4	1	1	1	7	2	3	3	4	5	4	3	7	7	1	2	2	2
6	8	7	2	3	4	8	7	5	3	3	1	6	5	4	7	1	2	1	4	4
7	4	8	1	2	3	4	7	6	3	3	1	7	8	3	7	8	2	7	4	4
8	3	7	6	3	4	5	4	8	2	3	1	6	1	2	5	4	1	8	2	2
8	1	7	7	5	4	7	6	4	3	3	4	1	2	1	7	1	2	1	2	2
6	5	8	8	7	5	4	8	2	4	3	1	1	1	1	1	6	3	4	3	3
6	5	4	4	7	6	3	1	1	2	3	5	5	5	4	3	6	1	2	2	2
7	6	5	3	7	8	2	3	7	4	3	6	4	2	2	3	7	1	2	2	2
7	8	2	6	7	4	3	6	5	4	3	7	2	1	2	3	5	5	5	4	4
8	8	2	1	8	2	2	8	8	2	2	8	1	2	2	2	8	2	8	2	2

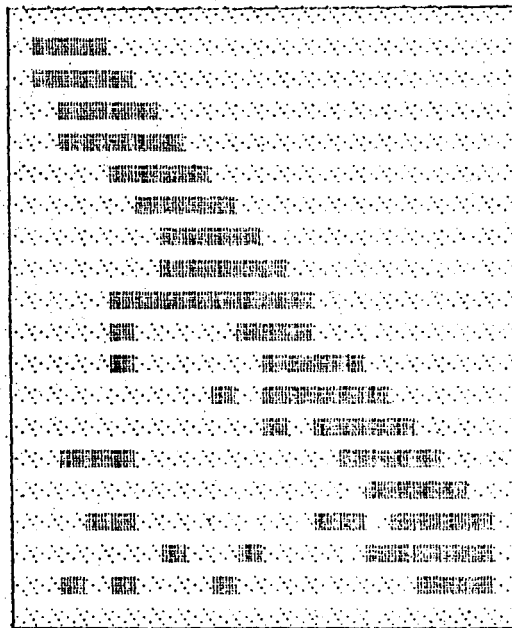
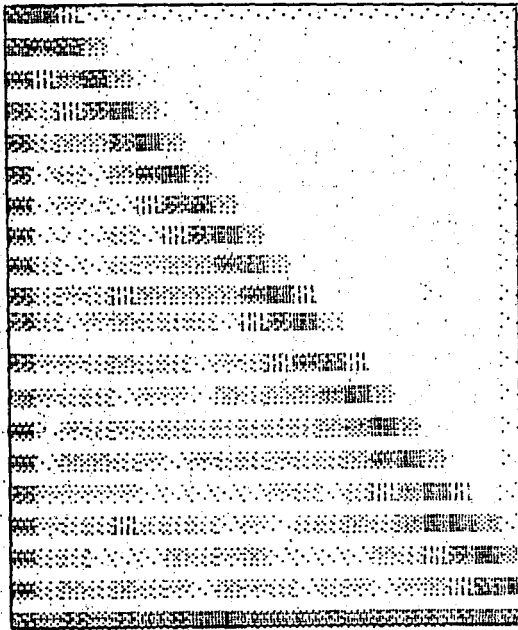


d-

Fig 4.6 The outputs of the edge detection scheme with Kirsch Masks and fixed thr. applied on a vertical edge
 a)Original im. b)Enhanced im. c)Threshold map
 d)Mask direction map e)Edge map after postprocess.algor.

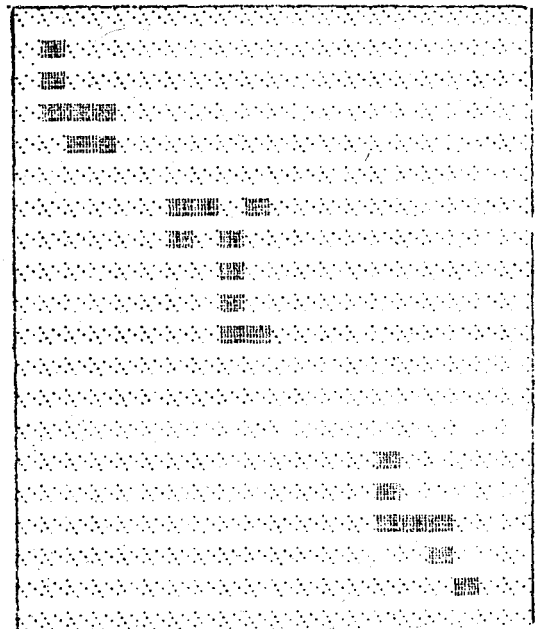


a



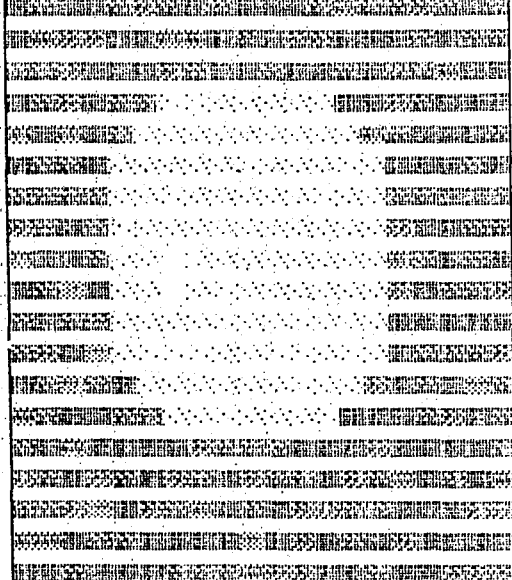
b-c

5	4	4	4	4	6	5	4	4	6	4	4	6	4	4	6	4	4
4	3	2	4	3	7	8	1	4	1	1	1	5	3	1	1	1	2
5	4	5	4	3	6	5	3	5	4	4	5	4	3	3	7	1	2
6	5	4	3	3	7	6	1	8	6	4	3	1	2	1	1	5	4
7	6	3	4	3	7	5	5	4	8	1	1	1	1	5	4	4	2
7	8	3	4	4	4	8	2	2	4	1	1	6	6	5	4	3	2
7	8	6	4	3	4	3	1	1	1	2	2	6	7	8	1	2	2
6	8	7	6	4	5	4	3	7	1	2	5	4	1	1	1	1	2
5	4	7	8	3	2	3	4	3	5	4	5	4	3	6	4	5	4
2	6	7	4	3	1	3	5	4	5	4	3	2	3	8	1	2	2
8	1	8	2	3	5	3	6	4	4	3	1	2	3	5	4	1	2
4	7	1	2	3	2	6	8	2	3	4	3	6	6	4	2	2	3
1	7	5	3	1	5	6	4	3	3	5	4	3	1	1	1	3	2
5	7	8	3	5	4	8	2	3	4	5	5	4	5	1	1	2	3
7	6	4	5	8	3	8	1	2	5	4	5	3	4	5	6	3	4
8	8	2	4	1	2	6	5	1	6	4	4	2	5	4	5	4	3
1	8	1	1	2	7	6	4	4	8	1	2	7	6	5	4	3	2
2	3	5	6	5	5	4	3	1	1	2	3	6	7	6	5	4	3
5	5	5	6	4	7	4	1	1	2	6	4	7	7	1	2	3	4
2	8	2	8	1	1	1	2	1	8	8	8	1	2	8	8	2	2

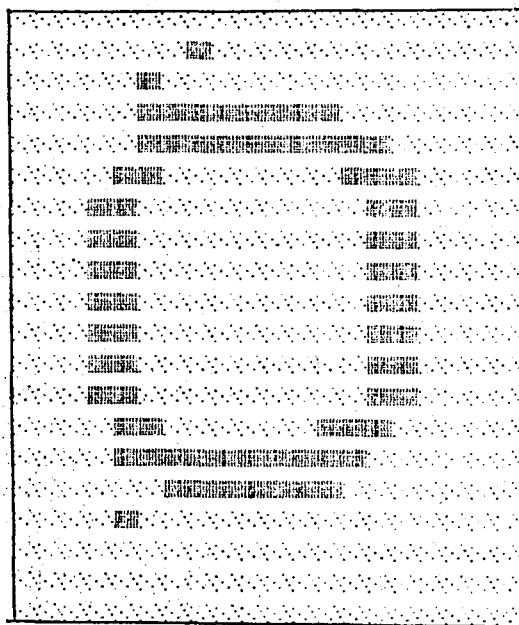
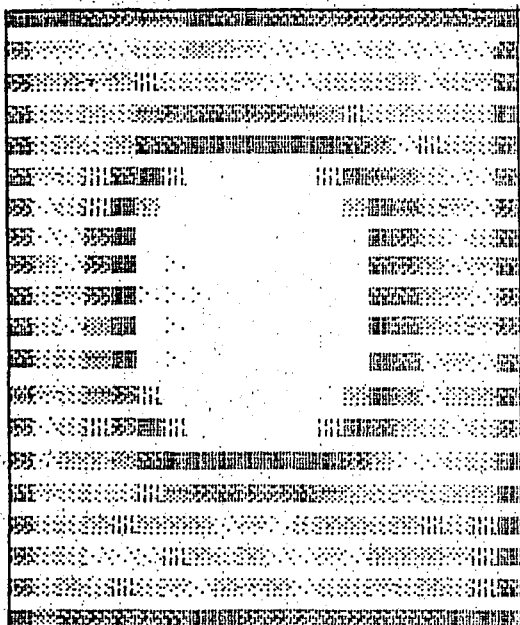


d-e

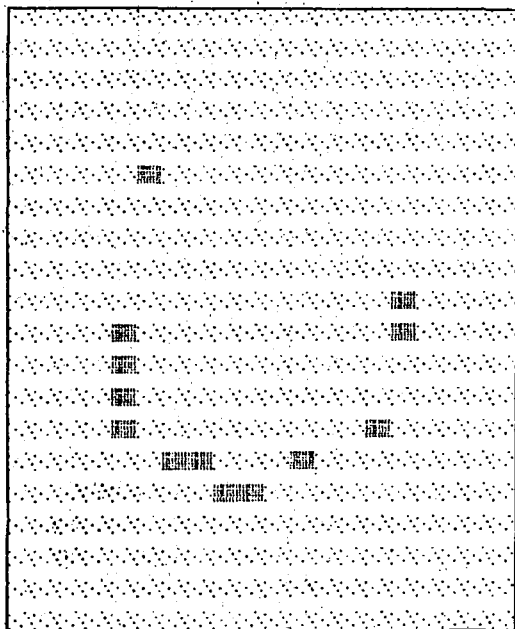
Fig 4.7 The outputs of the edge detec. scheme with Kirsch masks and fixed thr. applied on a diagonal edge a) Original image b) Enhanced image c) Threshold map d) Mask direction map e) Edge map after postprocess. algor.



a



b-c

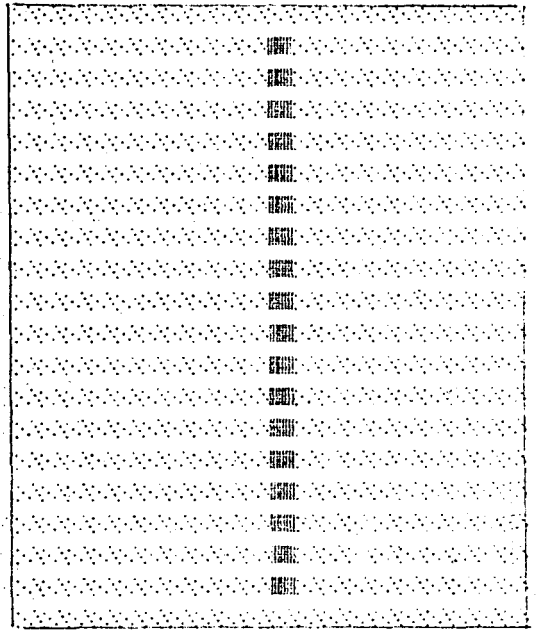
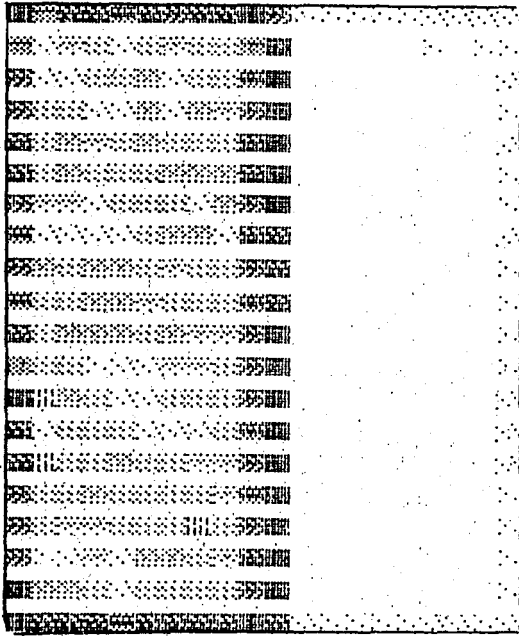


d

g. 4.8 The outputs of the edge detec. scheme with Kirsch masks and fixed thr. applied on a circular edge a)Original image b)Enhanced image Threshold map c)Edge map after postprocess. algor. d)Edge map after postprocess. algor.

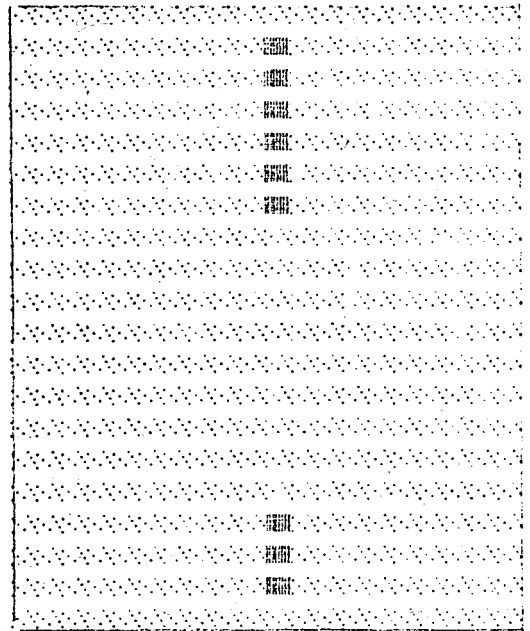


a



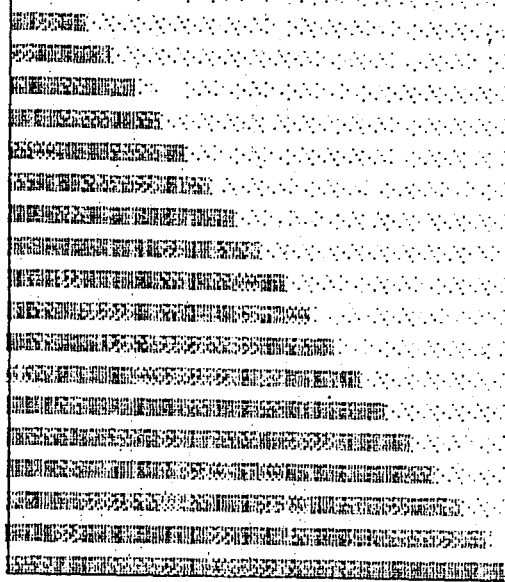
b-c

5	6	4	6	6	6	4	5	4	4	4	5	4	4	4	6	4	4	4
7	5	3	7	7	6	3	2	4	3	3	7	1	5	4	8	1	2	4
3	8	3	4	7	1	2	1	2	3	1	1	1	6	4	4	1	3	4
5	4	5	4	7	6	3	5	4	3	5	5	4	7	1	2	5	4	2
6	3	8	2	7	6	3	5	4	3	4	2	1	7	6	1	2	1	2
1	2	1	5	1	8	1	2	2	3	1	1	1	8	1	2	1	1	2
2	6	3	6	5	4	5	1	3	3	2	5	5	5	5	1	5	4	3
3	7	6	1	6	5	4	4	3	3	1	2	7	2	3	4	4	1	2
3	8	7	5	4	5	3	2	2	3	6	1	1	1	1	1	1	2	3
2	6	7	6	3	2	3	3	4	3	6	3	3	6	1	2	1	6	2
3	6	7	8	2	3	3	6	4	3	7	2	3	1	2	3	6	1	2
7	6	1	8	2	2	3	6	4	3	6	3	2	1	5	4	5	1	2
5	6	3	2	1	2	3	7	2	3	7	2	1	1	2	1	7	6	4
3	7	2	3	8	1	5	5	2	3	5	1	3	5	1	1	1	8	2
1	1	6	4	4	8	2	7	2	3	7	8	2	6	5	4	5	1	2
4	7	8	2	5	4	5	6	4	3	1	1	1	1	2	1	2	6	3
3	7	1	1	6	5	4	4	4	3	5	5	3	4	1	1	1	1	2
1	6	3	8	7	8	3	2	2	3	4	4	1	2	7	6	3	4	2
6	4	3	1	6	1	8	1	2	3	3	1	1	1	8	7	2	3	2
8	2	2	8	8	2	2	8	2	2	2	1	1	2	1	8	1	2	2

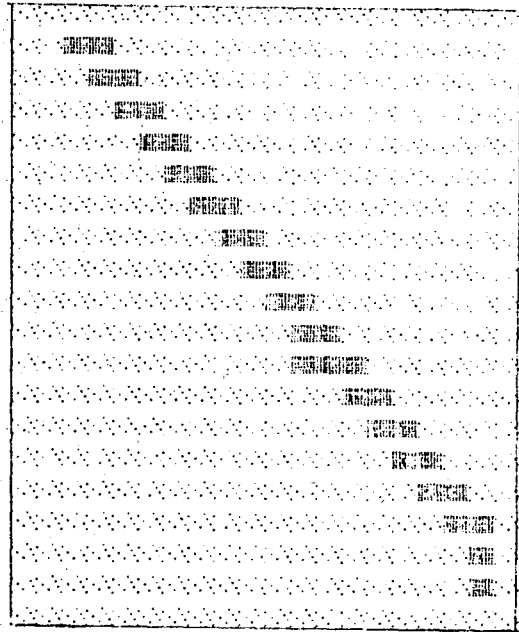
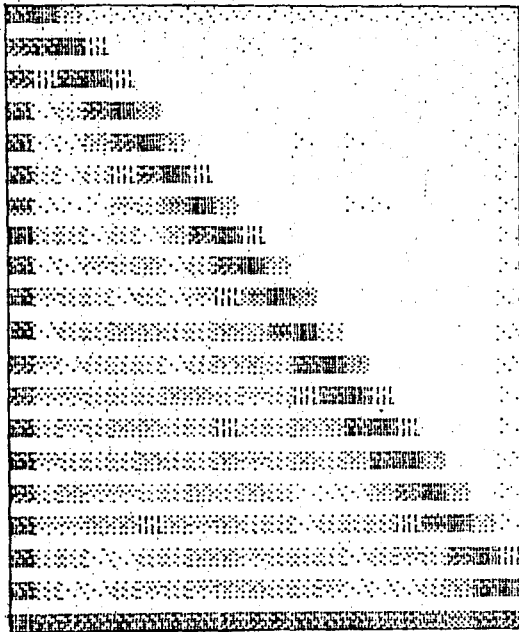


d-e

Fig 4.9 The outputs of the edge detect. scheme with Kirsch masks and adaptive thr. applied on a vertical image a)Original image b)Enhanced image c)Threshold map d)Mask direction map e)Edge map after postprocess algorithm

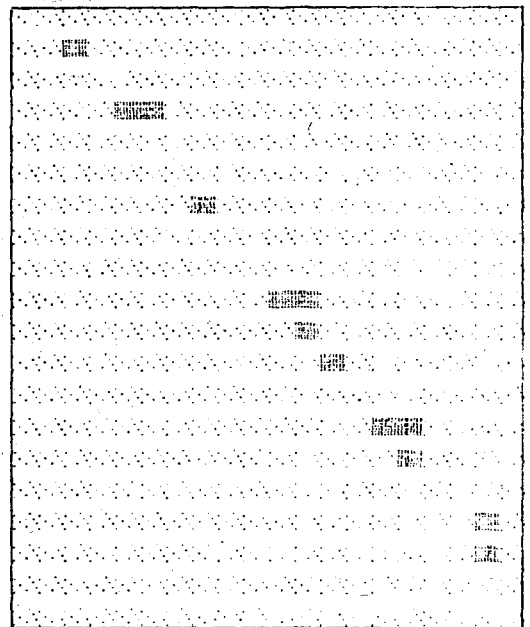


a



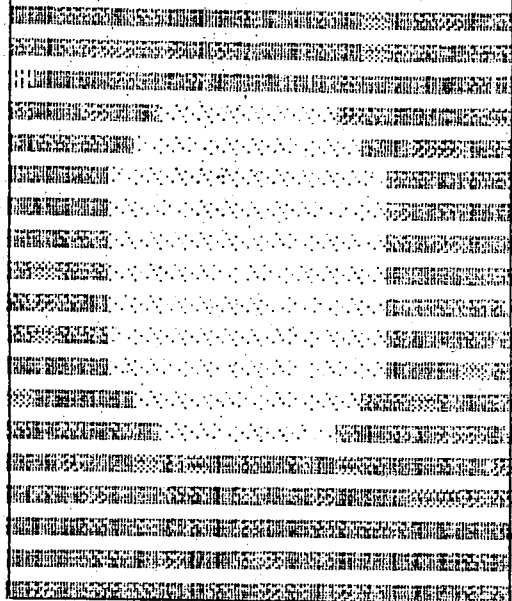
b-c

4	4	5	6	4	6	4	4	6	6	4	4	5	4	5	4	4	4		
3	4	3	1	2	1	8	1	4	3	7	2	1	8	1	6	5	4	3	
6	5	4	4	3	3	7	2	3	4	8	2	4	4	1	7	8	3	4	
7	4	5	4	3	1	1	1	2	3	1	8	1	2	4	6	1	2	2	
4	3	6	3	4	5	1	1	5	4	5	5	5	3	3	7	2	7	2	
2	1	8	2	3	4	4	3	6	7	6	3	4	4	2	6	3	1	2	
6	6	3	2	3	4	4	3	1	8	1	1	1	1	2	6	8	2	1	2
5	5	5	5	7	5	3	4	4	1	2	1	1	2	7	5	1	5	4	
5	4	5	6	4	3	2	4	4	4	3	5	5	3	4	6	4	4	2	
4	3	8	6	3	2	1	3	5	4	4	4	7	6	3	7	1	1	4	
1	2	1	7	2	3	6	4	6	4	4	3	6	1	2	1	7	6	3	
3	5	4	8	1	3	8	2	7	5	3	4	3	8	2	4	7	1	2	
2	6	3	4	8	1	8	2	7	5	6	5	4	5	5	3	3	8	2	
1	7	6	5	5	5	5	6	5	4	7	5	3	4	4	2	1	1	2	
4	6	6	3	4	4	6	7	2	3	8	8	2	5	4	3	8	1	2	
4	7	1	2	3	2	1	1	2	3	1	7	6	4	4	4	3	1	2	
3	8	8	1	2	3	7	6	5	3	7	7	8	1	2	3	4	3	4	
4	3	1	1	5	5	6	7	2	3	7	1	8	1	2	2	3	4	3	
2	2	6	4	4	7	4	8	1	2	7	8	7	2	3	5	5	5	4	
2	1	8	1	1	1	1	2	8	1	1	1	1	8	2	2	2	2	2	

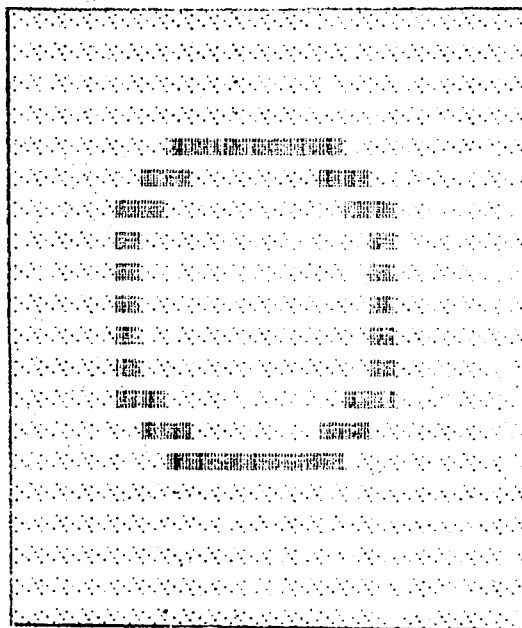
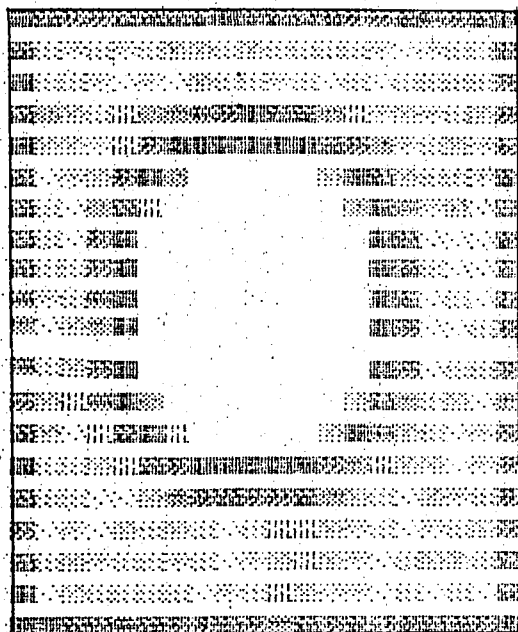


d-

Fig 4.10 The outputs of the edge detec. scheme with Kirsch masks and adaptive thr. applied on a diagonal image a)Original image b)Enhan.image c)Threshold map d)Mask direction map e)Edge map after postprocess.algor.

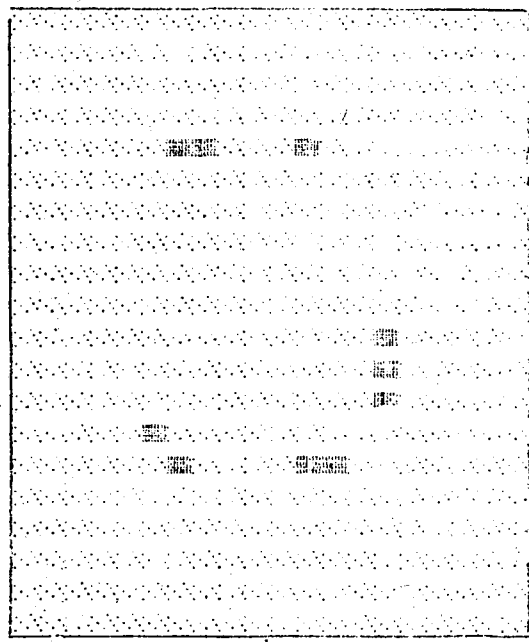


a



b-c

4	4	5	4	5	6	6	4	4	6	5	4	4	5	4	4	6	4	4
2	1	2	3	8	8	8	3	6	5	4	3	1	8	1	7	8	3	3
6	4	4	5	5	4	1	8	6	4	2	3	5	7	5	6	5	4	3
6	4	3	6	4	3	2	1	8	1	2	1	2	7	6	6	4	3	2
8	2	2	8	2	2	1	1	1	1	1	1	7	6	8	8	2	4	3
4	3	1	1	2	1	1	1	4	1	1	1	8	7	2	2	6	3	4
3	4	4	3	1	2	6	6	4	3	3	8	1	8	1	6	7	8	3
2	1	2	3	5	4	5	6	5	3	3	1	5	7	6	5	8	6	4
1	1	2	3	6	3	4	7	2	2	2	5	4	7	6	5	4	8	3
1	7	4	3	1	1	1	1	1	2	1	7	6	7	7	2	3	4	2
3	6	4	3	1	2	2	3	4	1	1	6	6	7	7	1	1	2	2
5	5	4	3	1	3	6	4	3	2	3	7	7	7	7	5	3	3	2
6	6	5	4	4	2	6	1	2	2	3	1	6	7	8	8	2	2	3
6	8	2	3	4	3	4	1	1	5	3	5	6	7	5	1	2	7	4
8	2	1	2	3	4	5	5	5	5	5	5	7	7	6	4	5	6	3
2	1	8	2	4	5	6	5	5	5	4	5	6	6	1	2	2	7	2
3	4	7	6	3	4	7	4	7	6	4	3	6	7	2	2	5	5	3
4	4	3	7	2	3	7	4	3	8	2	3	8	1	6	6	4	7	4
2	2	3	8	1	6	5	3	1	8	1	3	3	6	7	5	4	8	2
1	1	2	1	8	8	2	2	2	1	8	1	2	1	8	8	2	1	2



d-e

Fig 4.11 The outputs of the edge detec. scheme with Kirsch masks and adaptive thr. applied on a circular image a)Original image b)Enhan.image c)Threshold map d)Mask direction map e)Edge map after postprocess.algor.

diagonal and circular shapes, respectively are shown. These outputs are taken with the contrast value of $c=20$ and without the existence of noise.

The evaluation of these two alternatives will be done in the section 4.5 with the other edge detection schemes.

B) - EDGE DETECTION BY USING SOBEL OPERATOR AND SEVERAL THRESHOLDING TECHNIQUES

In this scheme, the Sobel operator (See Eq.4.2 and 4.3 and Fig 3.7) is used in the preprocessing level. In order to get the gradient image, the Sobel operator is applied to each pixel of the picture and the gradient values $GD(x,y)$ of each pixel at location x,y is obtained.

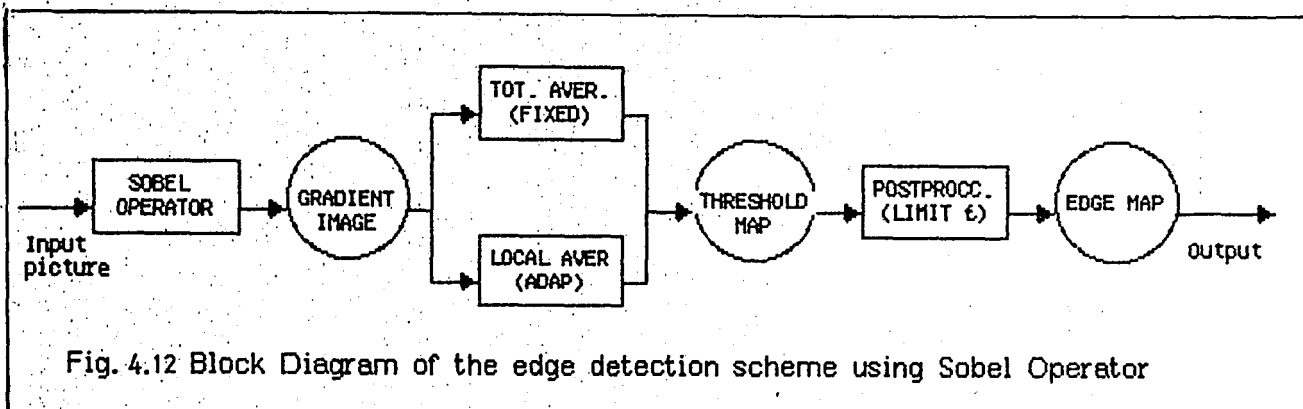
For this scheme, again two different types of thresholding techniques are used in order to obtain a decision on each pixel whether it is an edge point or not.

(1) - The fixed threshold is the same one which is used in the scheme with Kirsch masks. The threshold value is determined as the mean intensity of the gradient image. The threshold map is also obtained in the same manner.

(2) - The adaptive threshold is performed just as the third case explained in the first scheme. (The scheme with Kirsch masks) The threshold value is determined as the average intensity of the pixels in a 3x3 grid, surrounding each of the pixel in the gradient image. Then the threshold map is generated.

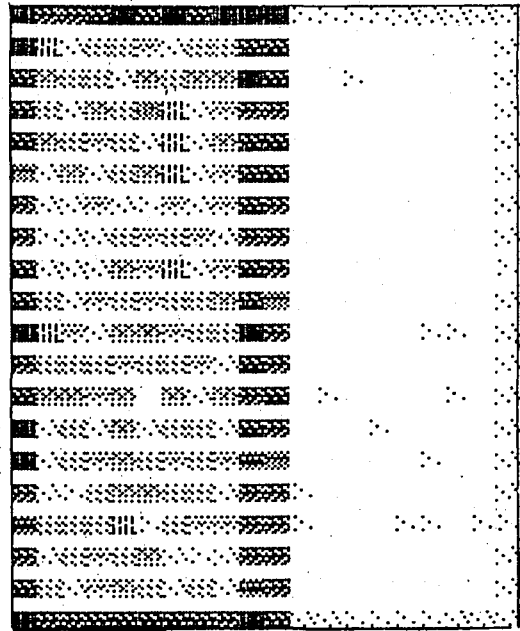
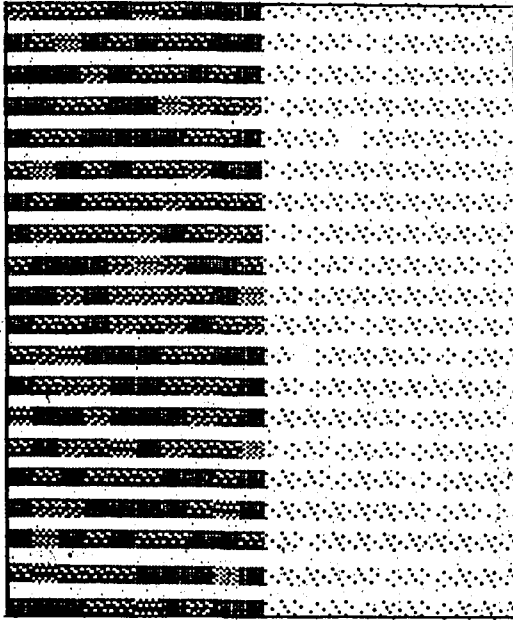
A post-processing algorithm is applied on the threshold map, as a second level thresholding. A limit for the number of 1's in the threshold map is

given. This limit can be changed according to the user requirements. If the number of 1's in the edge region can be expected, to be as in the case of the test images used in this project, this limit can easily be determined. If it is an actual image, some tests must be performed before getting the best result. This post-processing operation gives good results like sharpening the existing edge and eliminating the subedges. The block diagram of the proposed edge detection scheme is shown in Fig 4.12. The square blocks represent the operations and the circular blocks represent the outputs of the preceding operations.

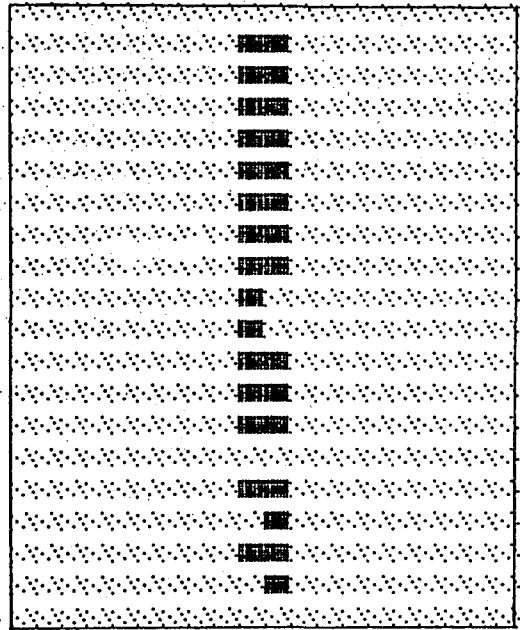
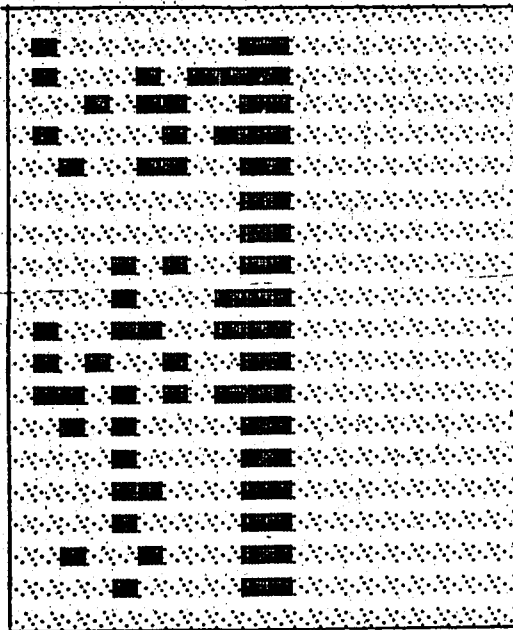


The results of the tests with the vertical, diagonal and circular shapes are shown in Fig 4.13 through 4.18. In Fig 4.13 edge detection scheme with Sobel operators and fixed threshold is shown on a vertical image. In Fig 4.14 the same scheme is shown on a diagonal image and in Fig 4.15 on a circular image. In Fig 4.16 through 4.18 Sobel operator with adaptive threshold is shown on a vertical, diagonal and circular shapes, respectively. All the outputs are taken with contrast value of 20 and without the existence of noise.

Comparisons of these techniques and the evaluation of each scheme will be discussed in section 4.5.

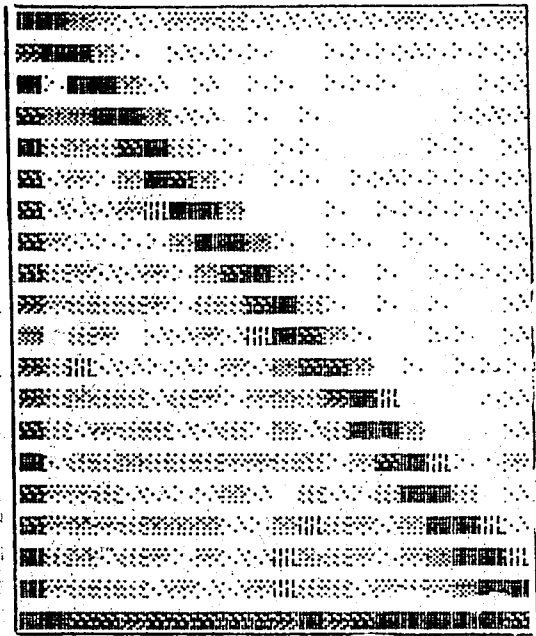
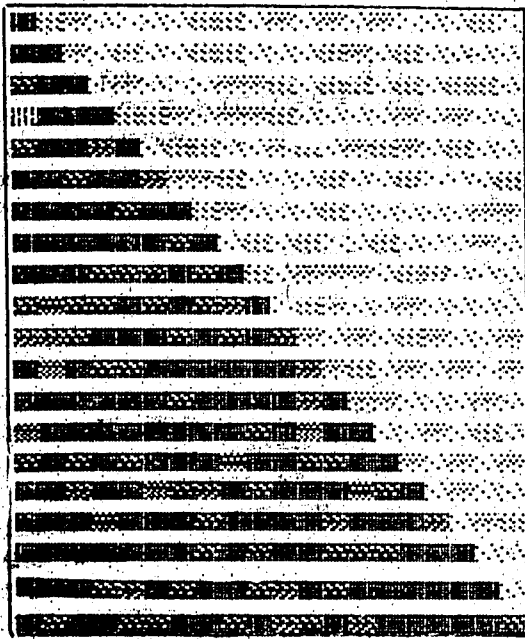


a-b

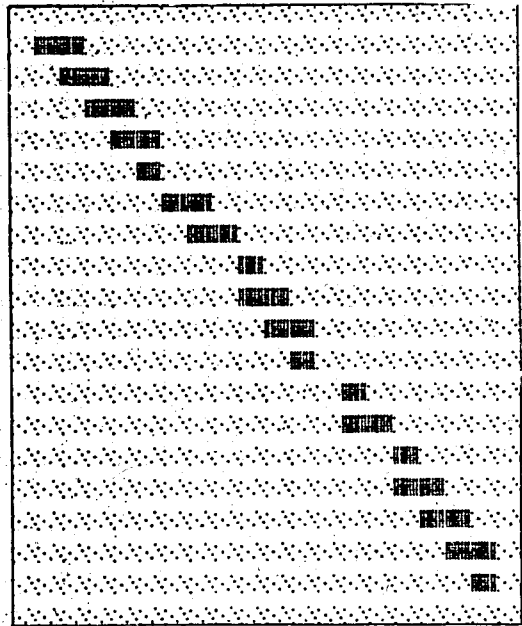
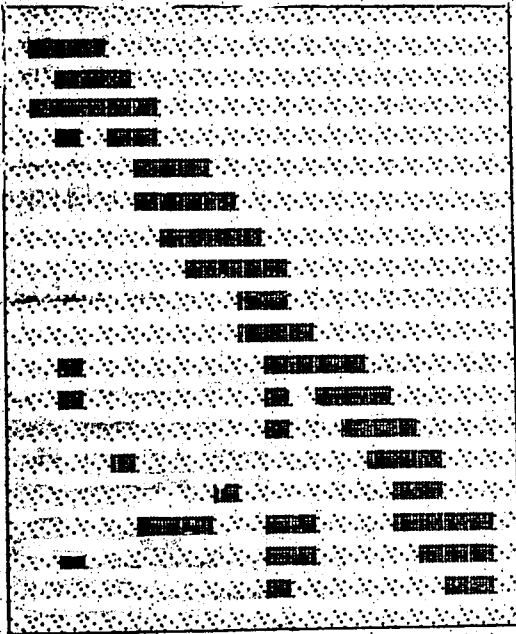


c-d

Fig 4.13 Simulation results of the Sobel operator and fixed threshold applied on a vertical edge a)Original image b)Enhan.image c)Threshold map. d)Edge map after postprocessing algorithm

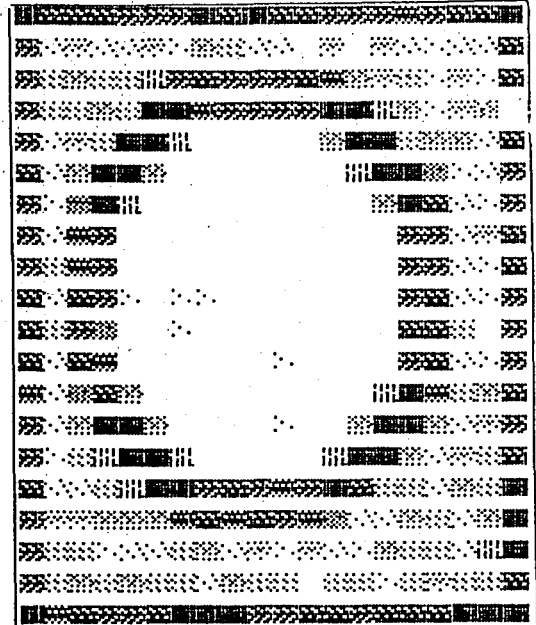
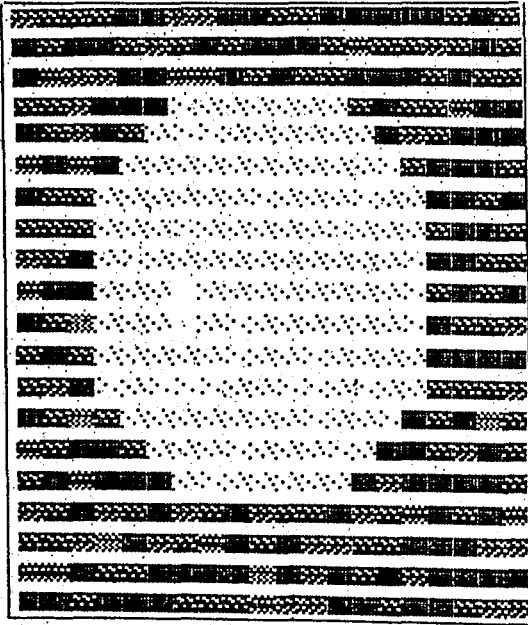


a - b

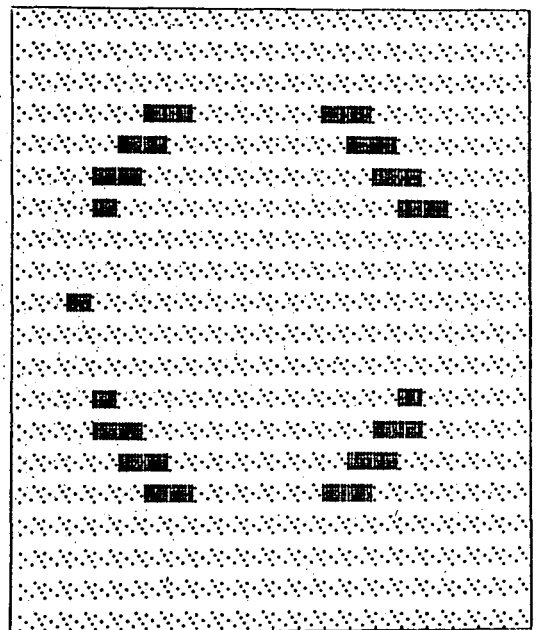
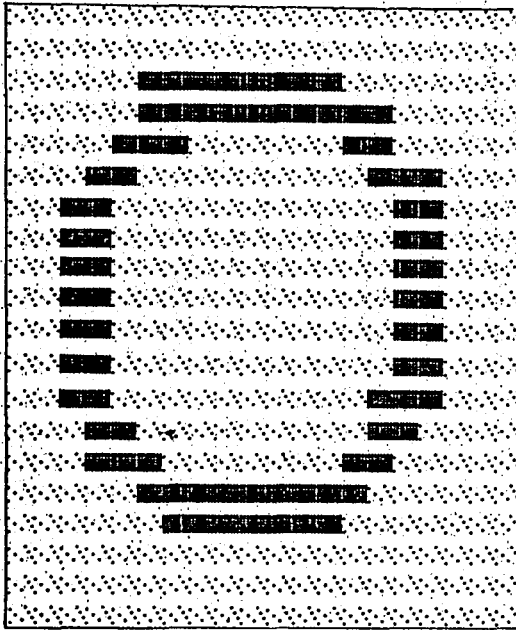


c - d

Fig 4.14 Simulation results of the Sobel operator and fixed threshold applied on a diagonal edge a)Original image b)Enhanced image c)Threshold map d)Edge map after postprocessing algorithm

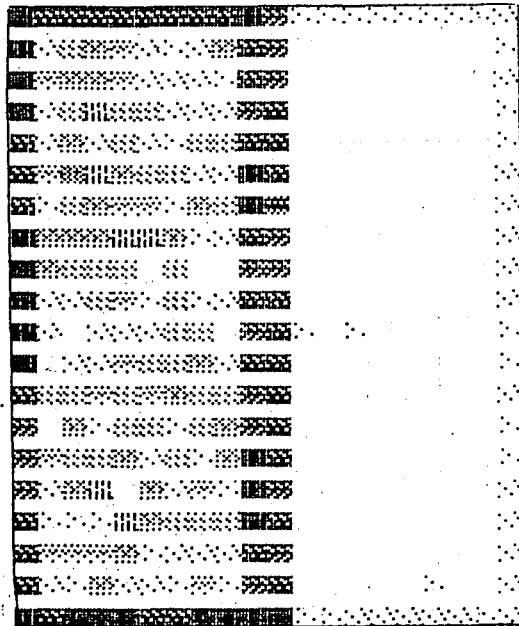
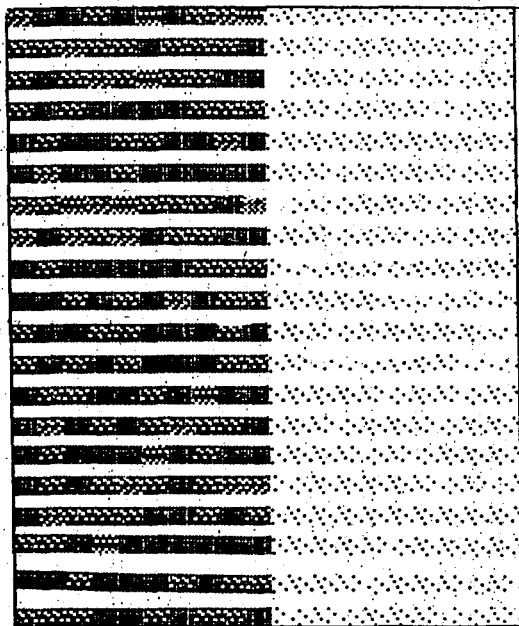


a-b

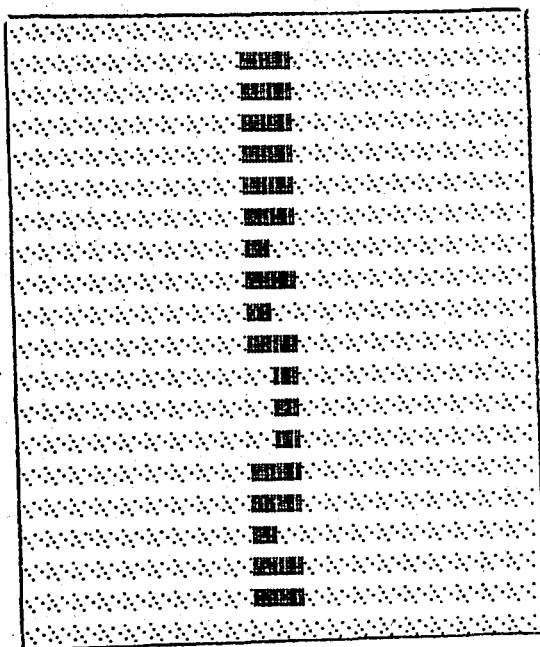
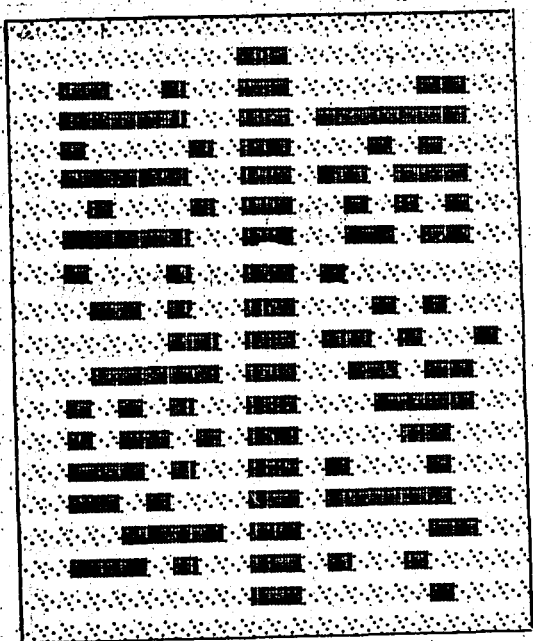


c-d

Fig 4.15 Simulation results of the Sobel operator and fixed threshold applied on a circular edge a)Original image b)Enhanced image c)Threshold map d)Edge map after postprocessing algorithm

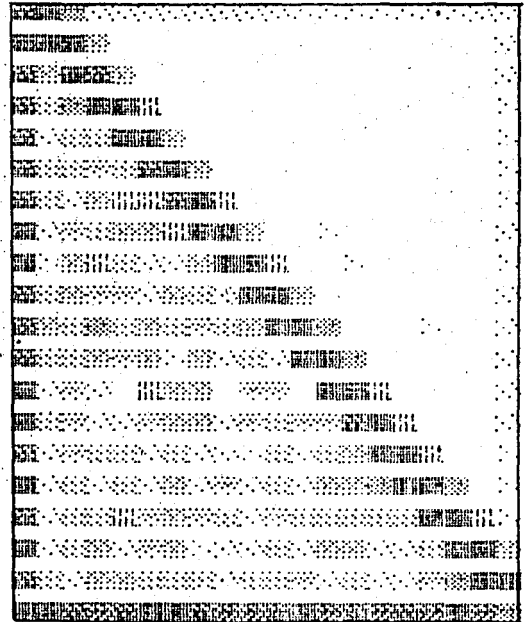
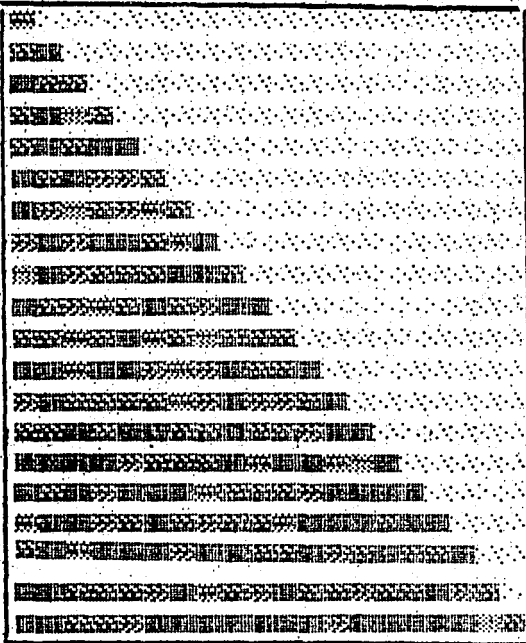


a-b

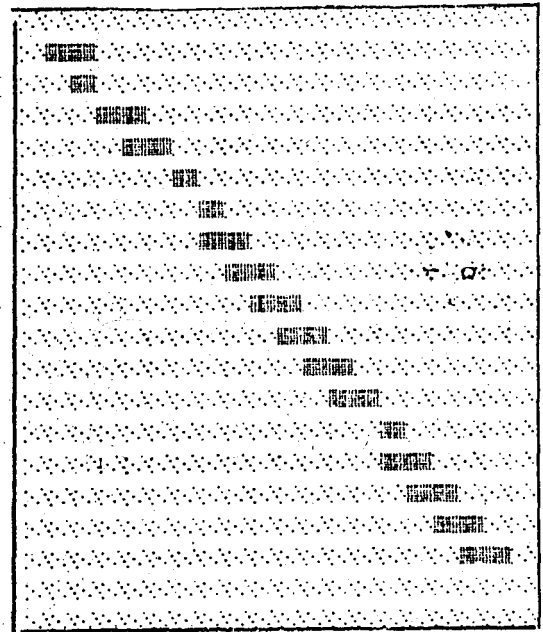
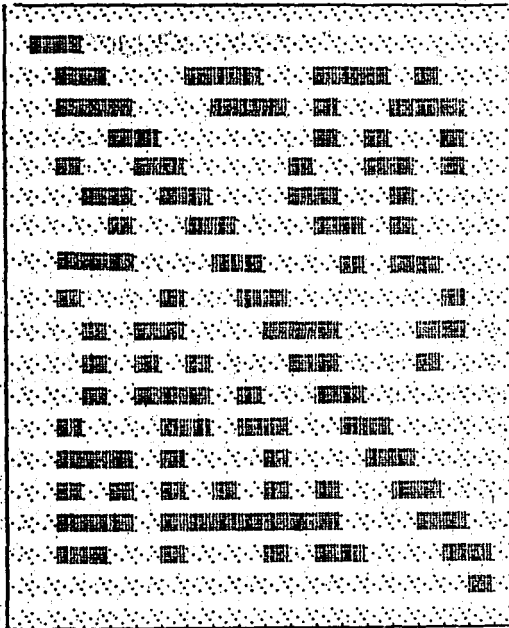


c-d

Fig 4.16 Simulation results of the Sobel operator and locally adaptive threshold applied on a vertical edge a)Original image b)Enhan.image c)Threshold map d)Edge map after postprocessing algorithm

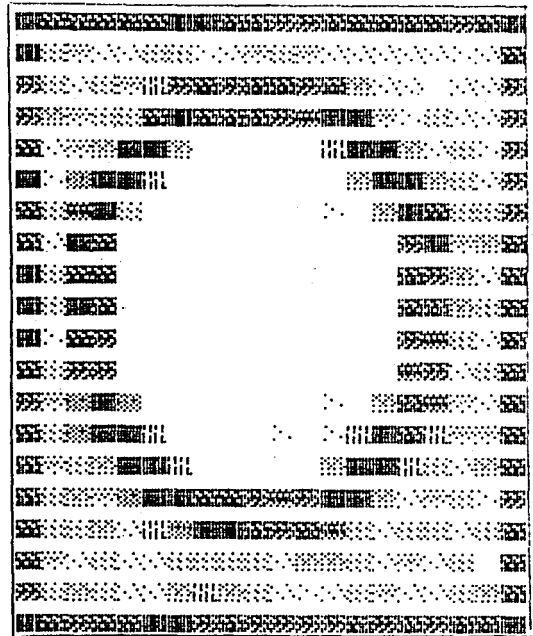
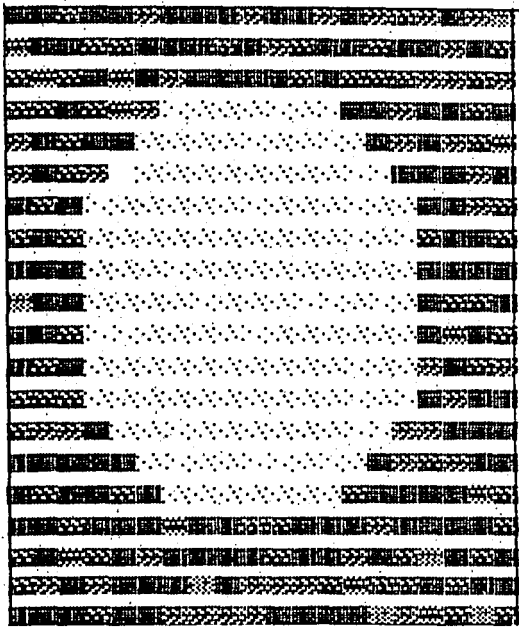


a - b

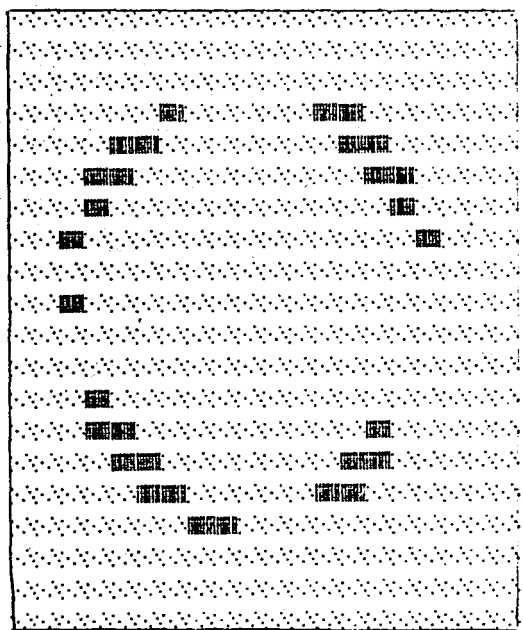
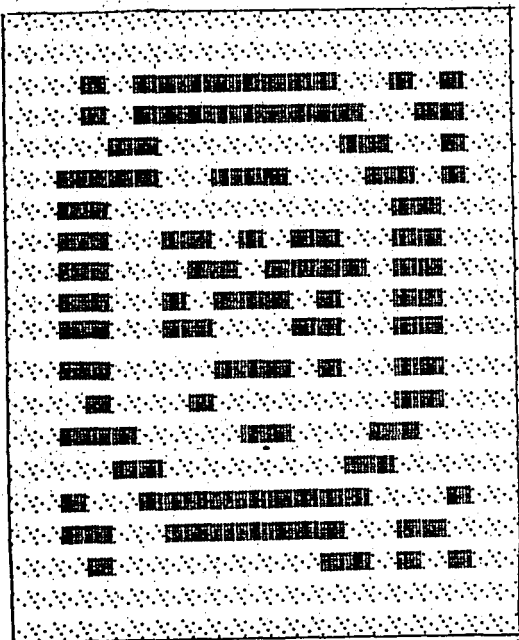


c - d

Fig 4.17 Simulation results of the Sobel operator and locally adaptive threshold applied on a diagonal edge a)Original image b)Enhanced image c)Threshold map d)Edge map after postprocessing algorithm



a-b



c-d

Fig 4.18 Simulation results of the Sobel operator and locally adaptive threshold applied on a circular edge a)Original image b)Enhanced image c)Threshold map d)Edge map after postprocessing algorithm

C) - EDGE DETECTION SCHEME USING PRODUCT OF DIFFERENCE EQUATIONS AND
FIXED THRESHOLDING TECHNIQUE

This scheme has been explained in more detail in section 3.3.9. Shortly, the formula for differencing operation is:

$$H_k(x,y) = \left| \frac{A(x,y+k)+\dots+A(x,y+1)}{k} - \frac{A(x,y)+\dots+A(x,y-k+1)}{k} \right| \quad (4.6.a)$$

$$V_k(x,y) = \left| \frac{A(x+k,y)+\dots+A(x+1,y)}{k} - \frac{A(x,y)+\dots+A(x-k+1,y)}{k} \right| \quad (4.6.b)$$

in horizontal and vertical directions, respectively, where $k=1,2,4,\dots,32$. For preprocessing level, the above equations are used. k is defined as a changeable value. If it is chosen to be 2 then,

$$H_k(x,y) = \left| \frac{A(x,y+2)+A(x,y+1)}{2} - \frac{A(x,y)+A(x,y-1)}{2} \right| \quad (4.7.a)$$

$$V_k(x,y) = \left| \frac{A(x+2,y)+A(x+1,y)}{2} - \frac{A(x,y)+A(x-1,y)}{2} \right| \quad (4.7.b)$$

must be calculated. In this case, the gradient value of a pixel at a location x,y can be defined by

$$\text{Max} \{H_2(x,y), V_2(x,y)\} \quad (4.8)$$

As an alternative

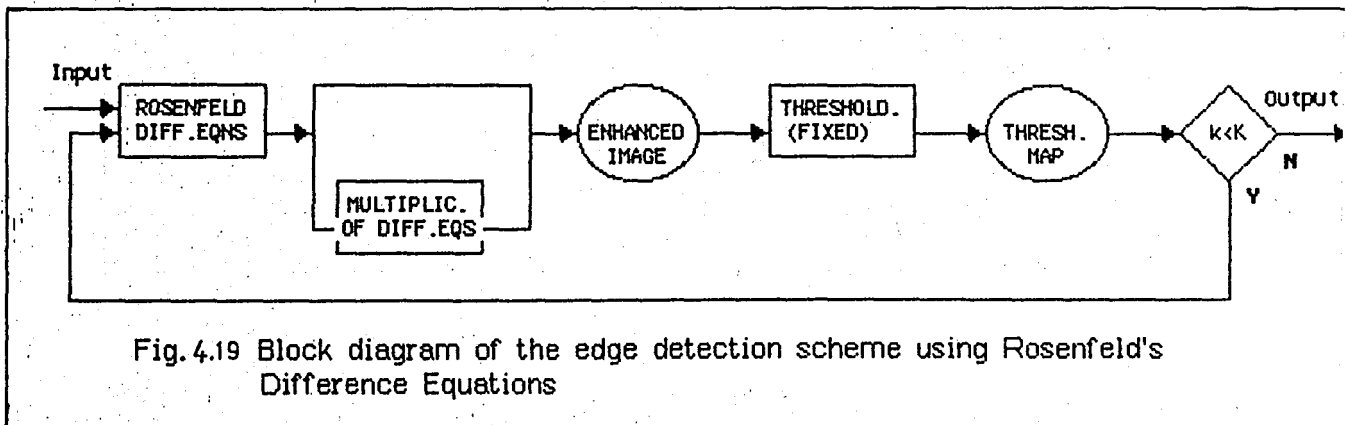
$$H_1(x,y) = | A(x,y+1) - A(x,y) | \quad (4.9.a)$$

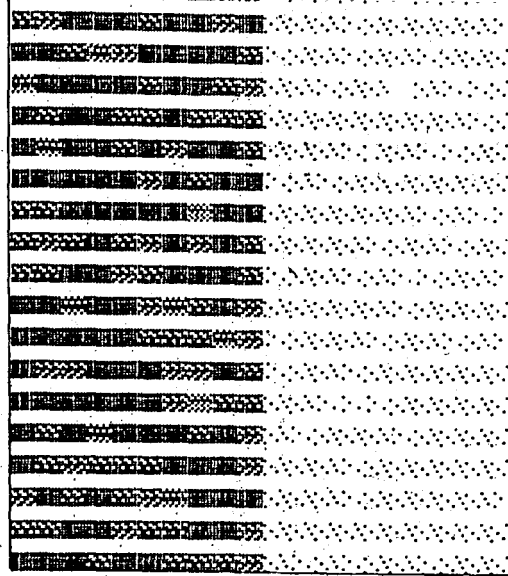
$$V_1(x,y) = | A(x+1,y) - A(x,y) | \quad (4.9.b)$$

are also calculated and the products of vertical and horizontal differences are found. Then the gradient value is determined as a maximum of these products.

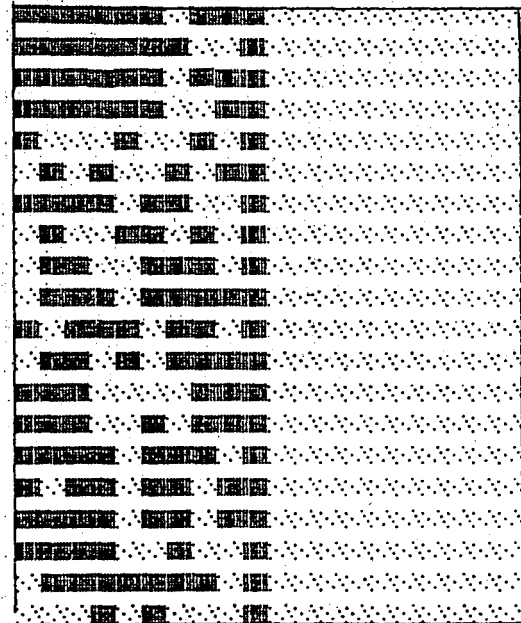
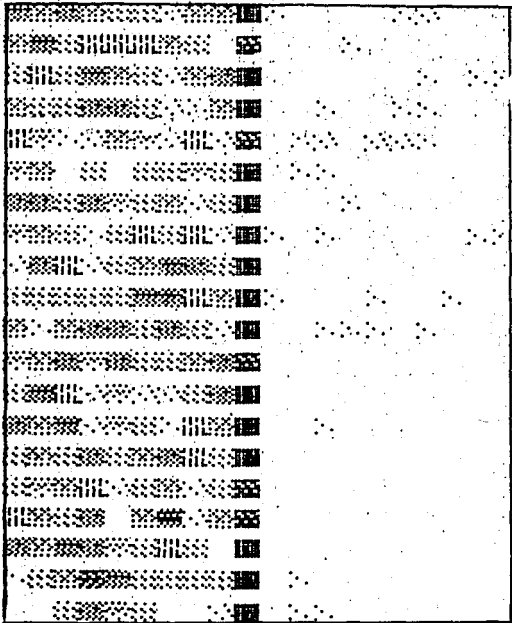
$$\text{Max} \{ H_1(x,y) \cdot H_2(x,y), V_1(x,y) \cdot V_2(x,y) \} \quad (4.10)$$

If large k is chosen, then in the same manner the gradient value can be determined from only the difference equation or the product of differences. The program of this scheme calculates both difference equations and their products. It can be seen from the outputs that for large k , the edge becomes more and more conspicuous, but the larger the k the less precisely localized is the detected edge. If the differences are multiplied together for a range of values of k , the result tends to yield sharply localized detection of the edges. The block diagram of the proposed edge detection scheme is shown in Fig 4.19. Some outputs are shown in Fig 4.20 through 4.23.

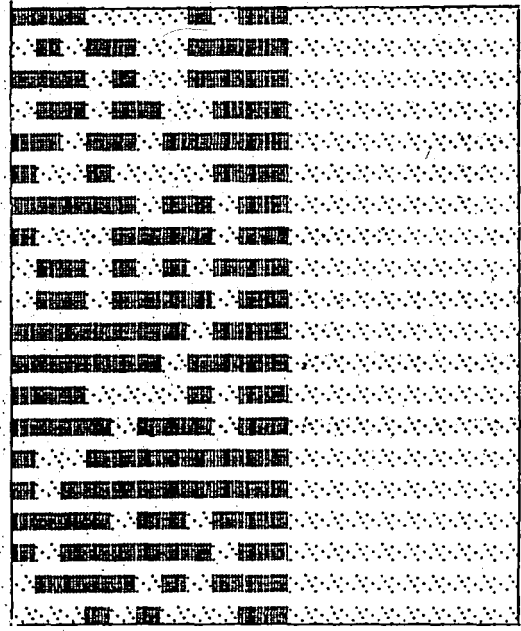
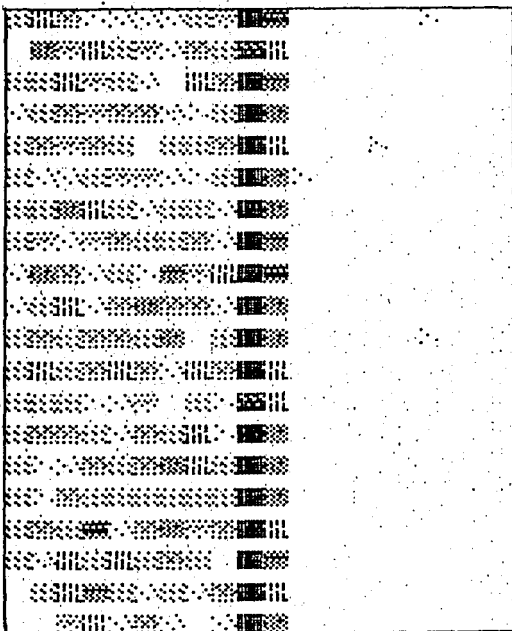




a

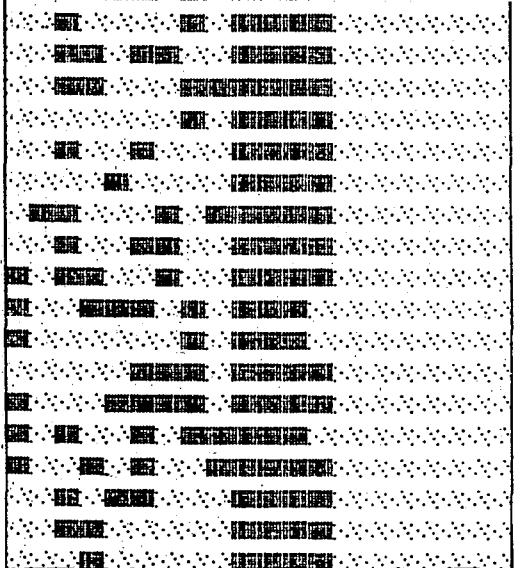
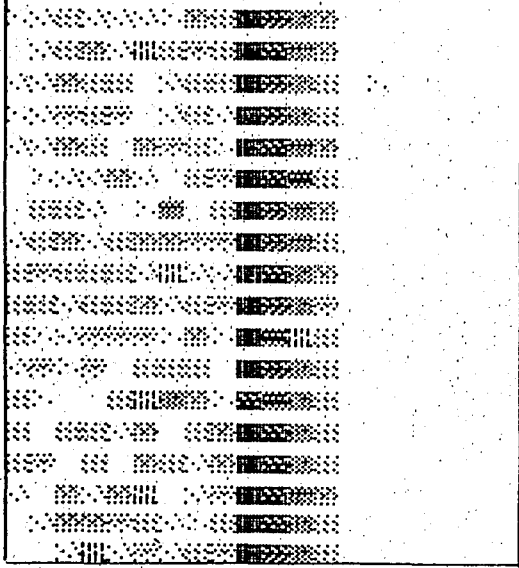


b-c

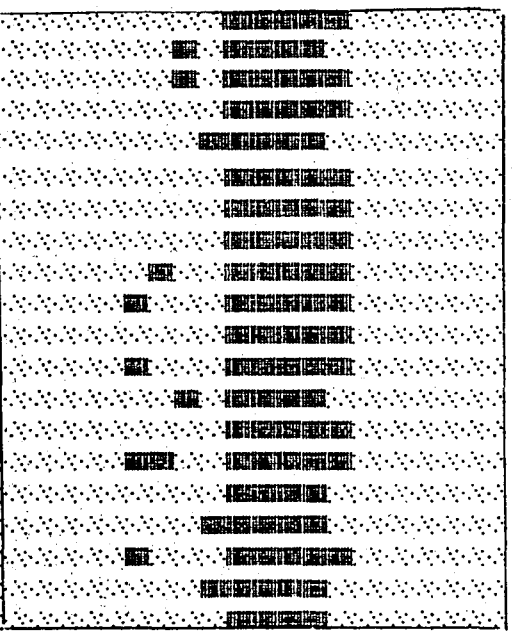
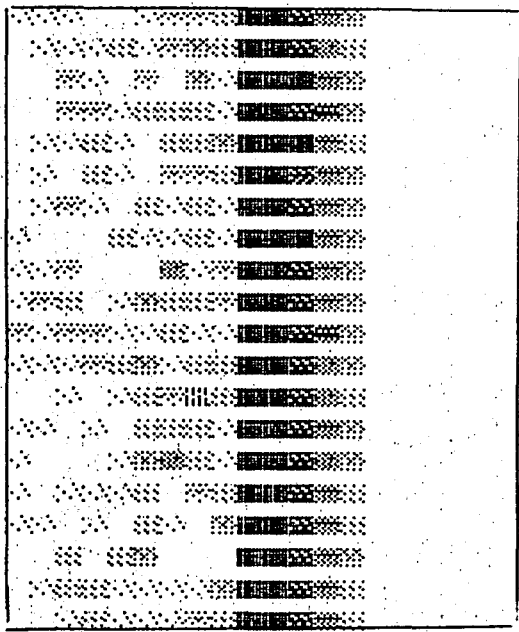


d-e

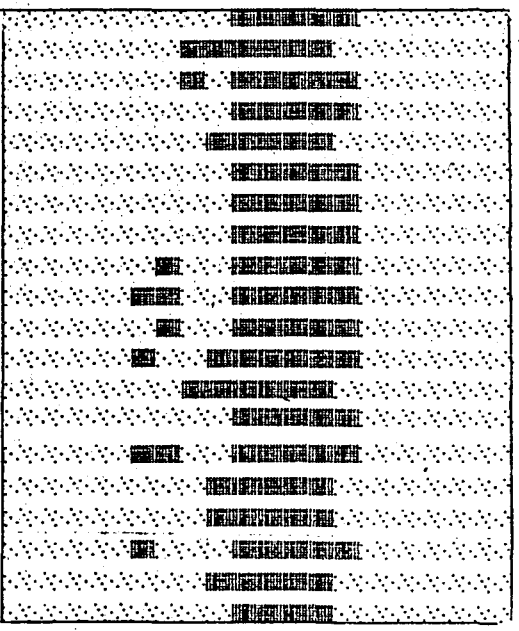
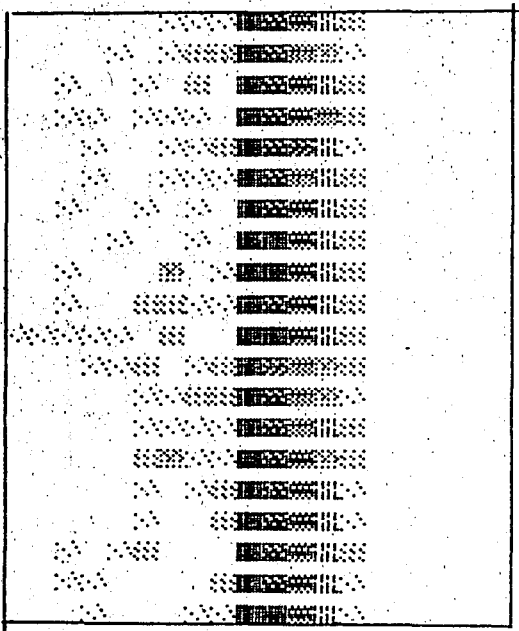
Fig 4.20 Simulation results of the Rosenfeld difference eqs. with fixed thr. using vertical edge a) Orig. image b) Enhan. image, $k=1$ c) Thr. map, $k=1$ d) Enhan. image, $k=2$ e) Thr. map, $k=2$



f-g

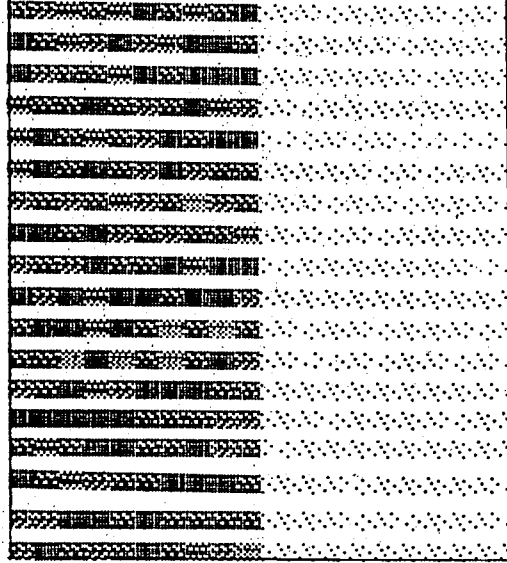


h-i

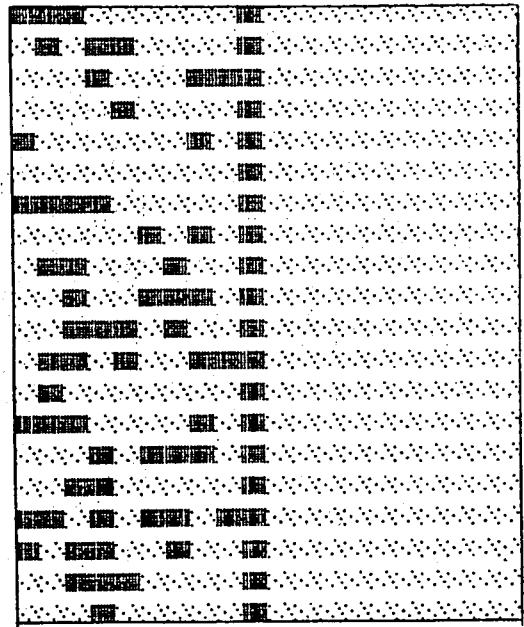
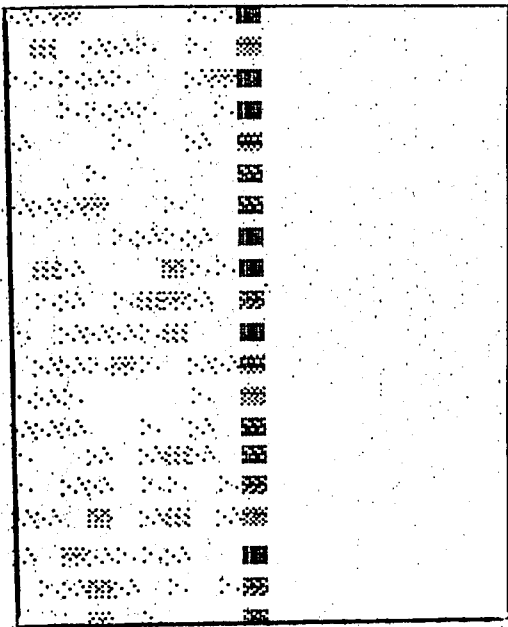


j-k

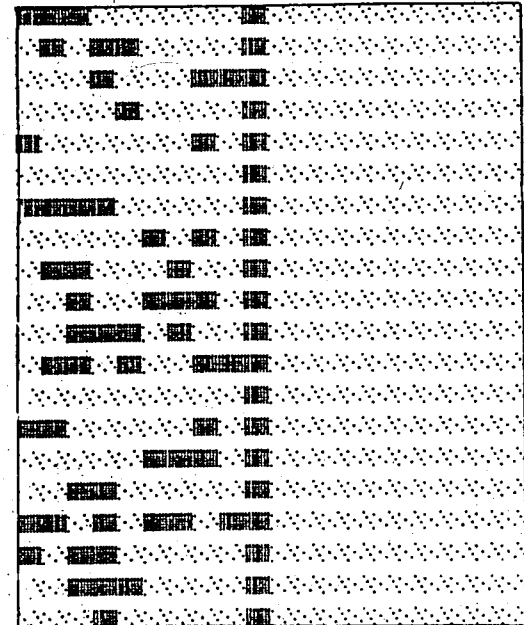
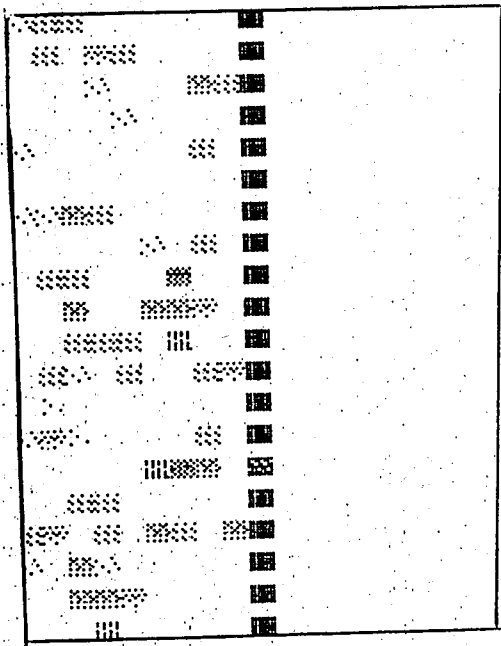
Fig 4.20 (continued) f) Enhan.image, k=4 g) Thr.map, k=4 h) Enhan.image, k=8 i) Thr.map, k=8 j) Enhan.image, k=32 k) Thr.map, k=32



a

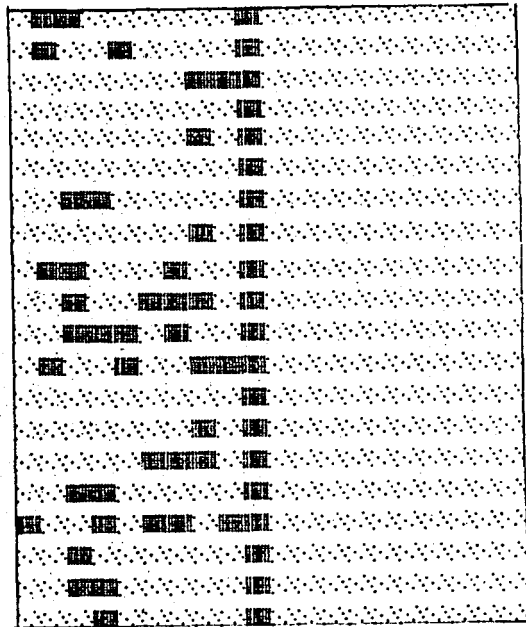
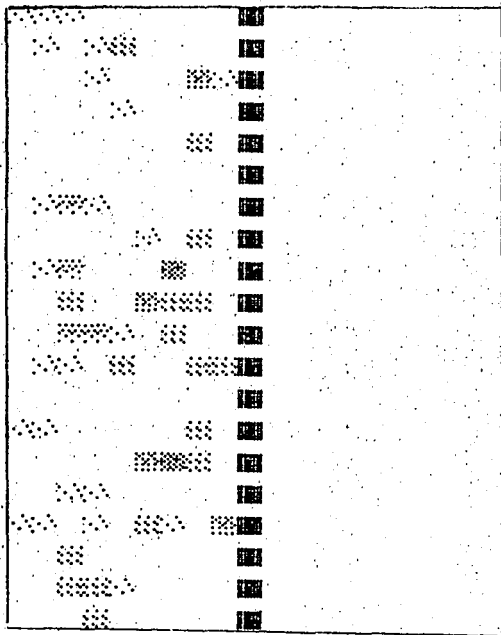


b-c

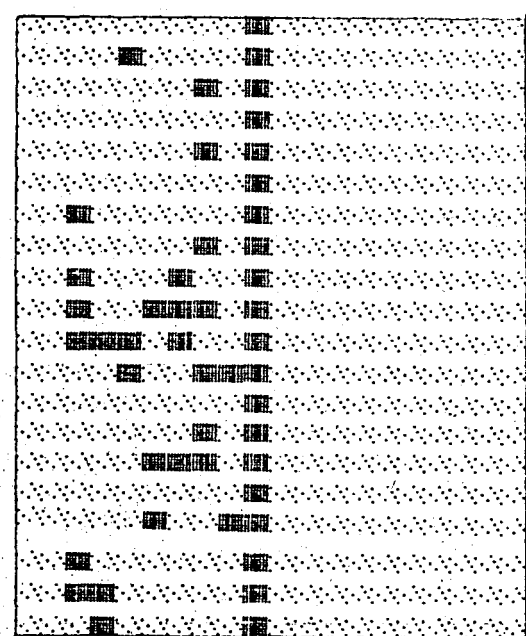
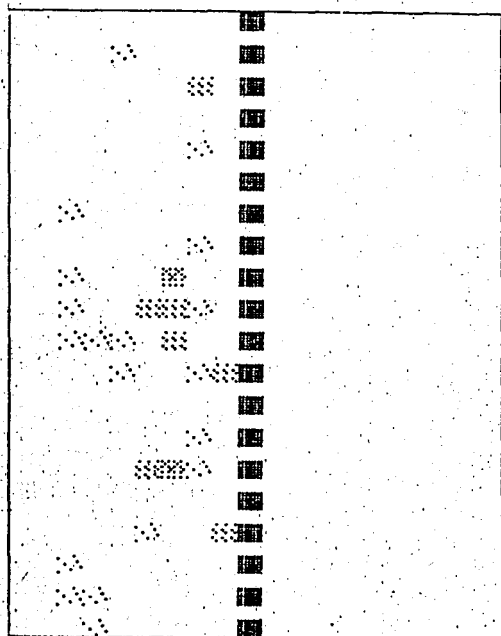


d-e

Fig 4.21 Simulation results of Rosenfelds product of differences with fixed thres.applied on a vertical edge a)Original image b)Enhan.image for $k=2*1$ c)Thr.map, $k=2*1$ d)Enhan.image, $k=4*2*1$ e)Thr.map, $k=4*2*1$

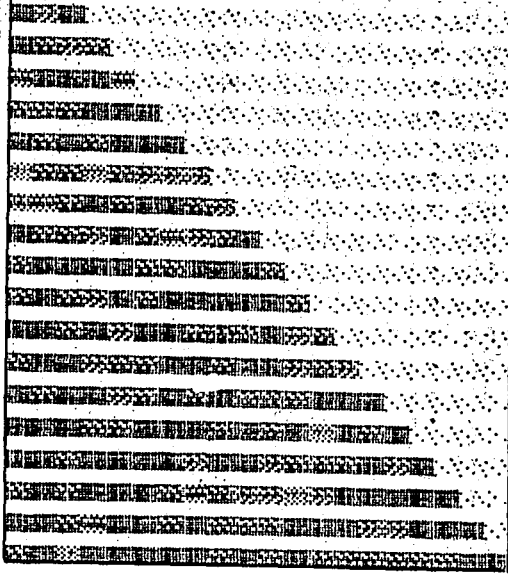


f-g

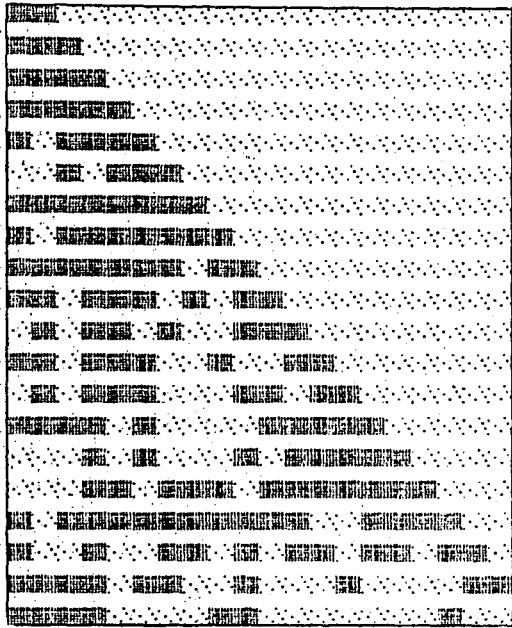
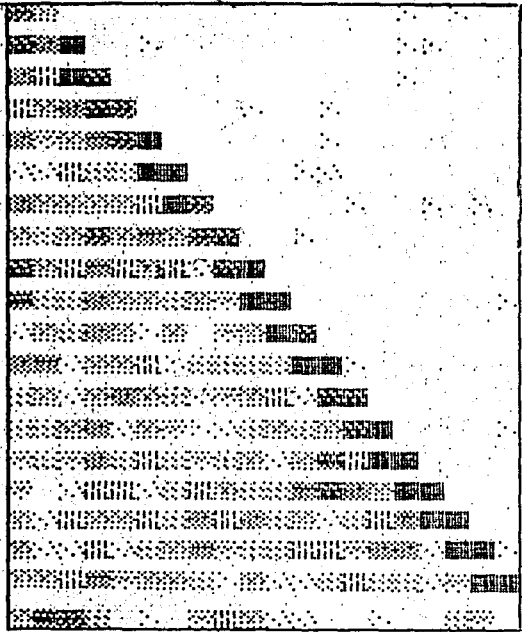


h-i

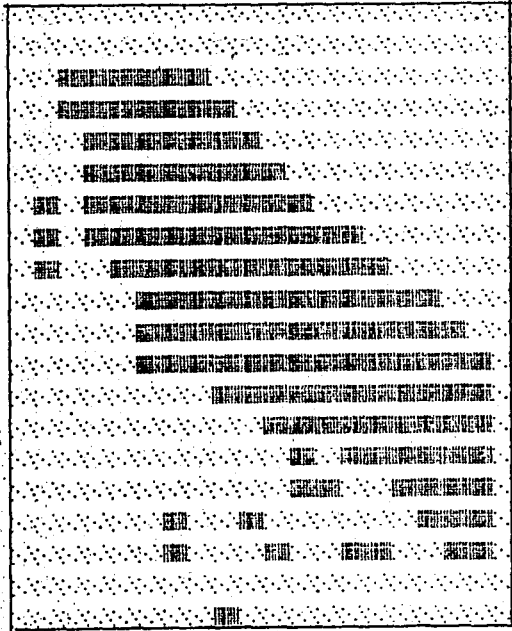
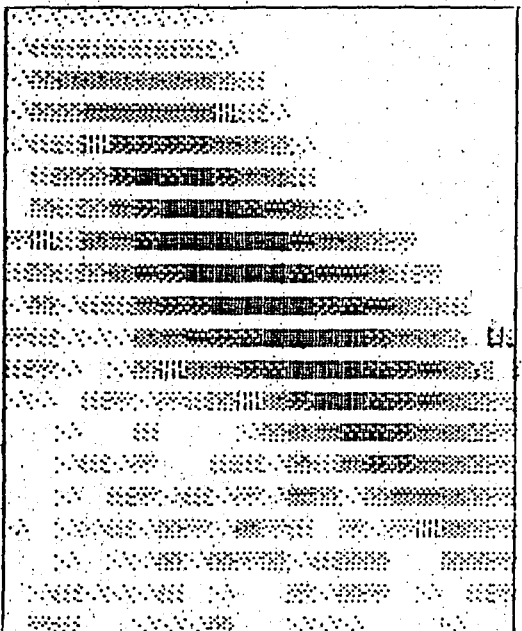
Fig. 4.21 (continued) f) Enhan. image, $k=8*4*2*1$ g) Thr. map, $k=8*4*2*1$
 h) Enhan. image, $k=32*16*8*4*2*1$ i) Thr. map, $k=32*16*8*4*2*1$



a

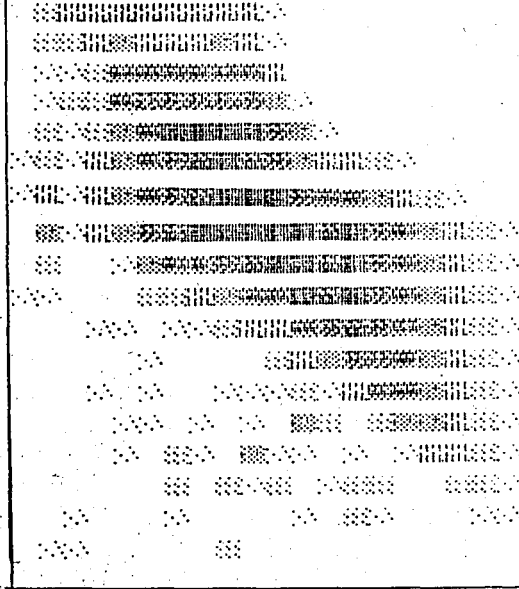
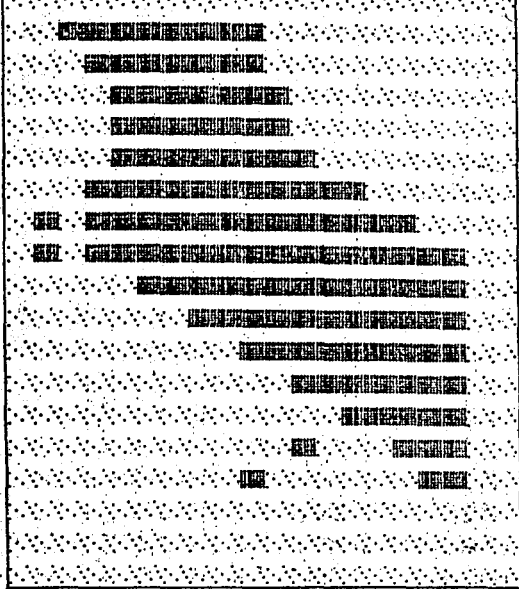


b - c

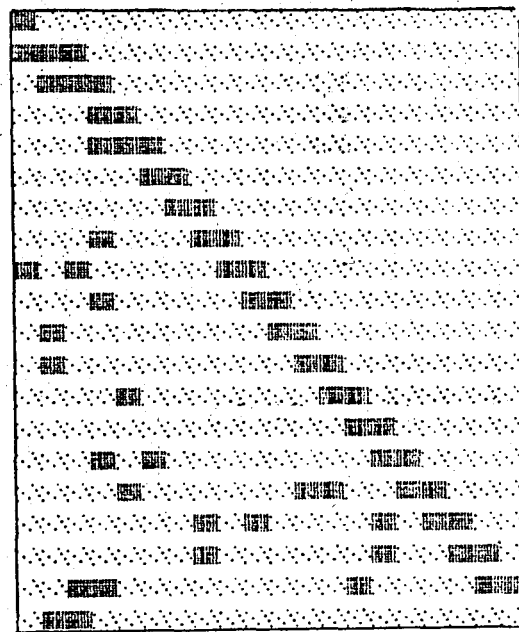
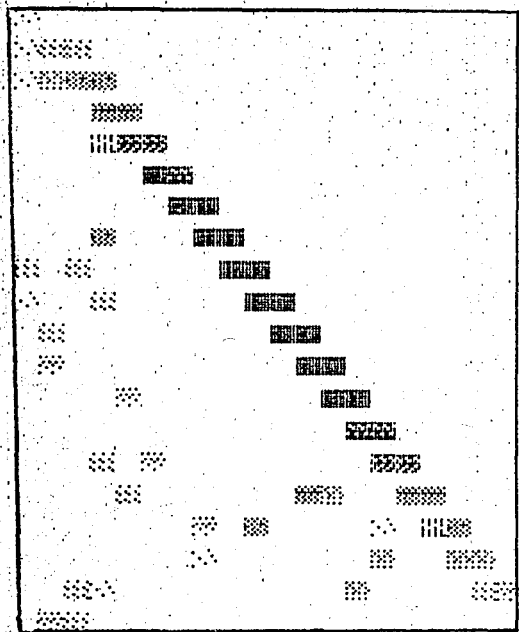


d - e

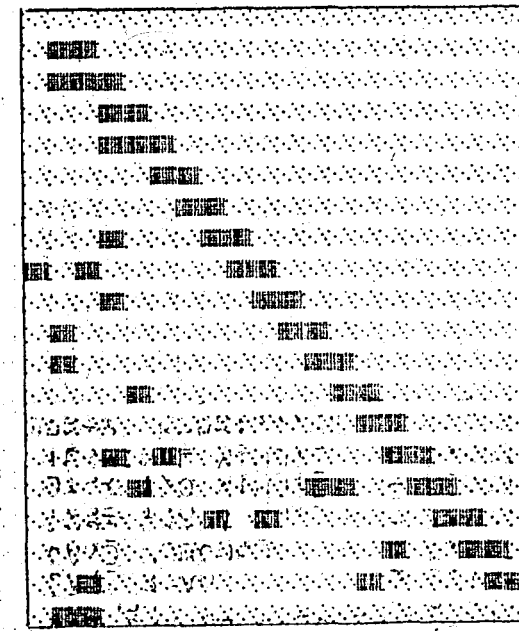
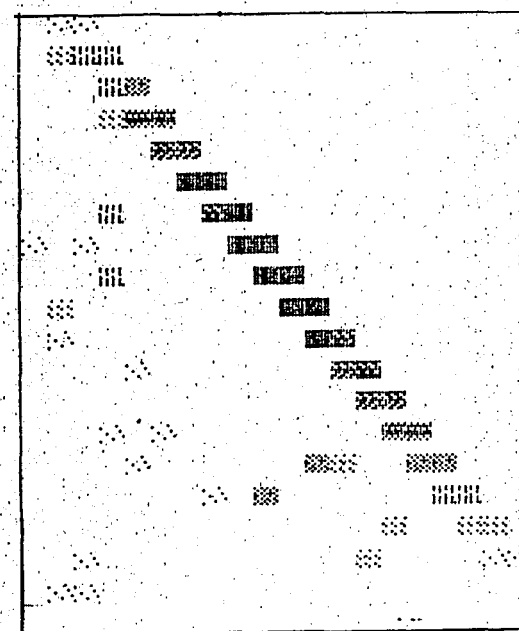
Fig 4.22 Simulation results of Rosenfelds algorithm with threshold. using a diagonal edge a)Original image b)Enhan. image, k=1 c)Thr. map, k=1 d)Enhan. image, k=8 e)Thr. map, k=8



f-g

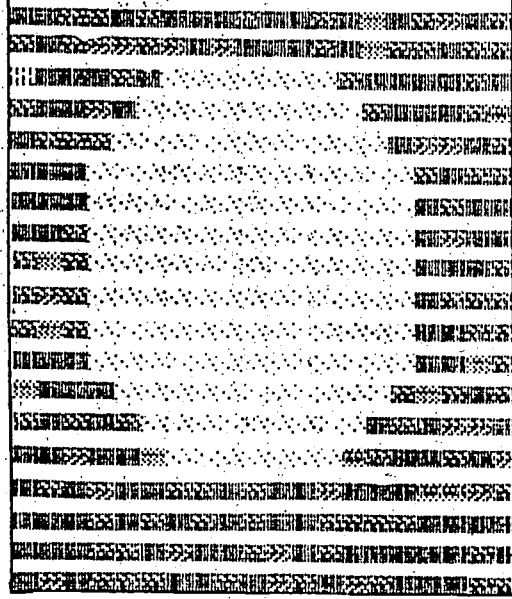


h-i

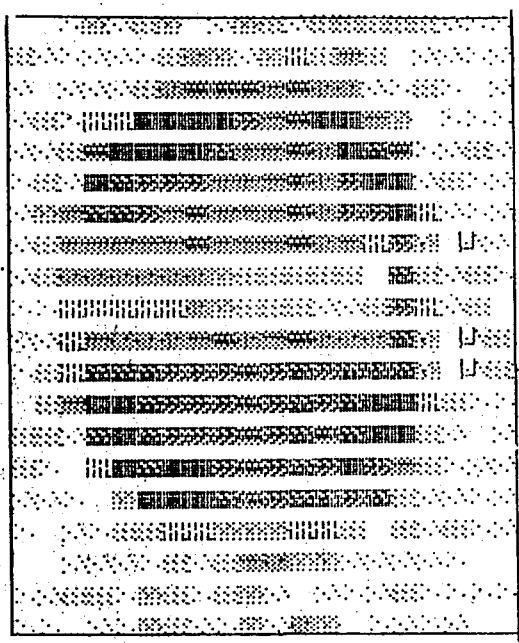
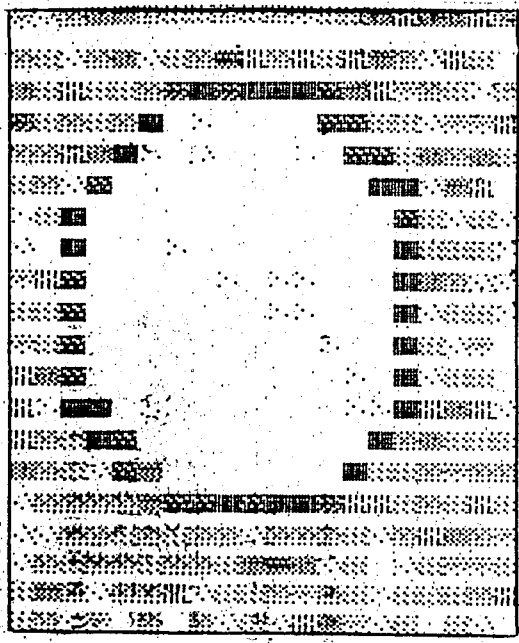


j-k

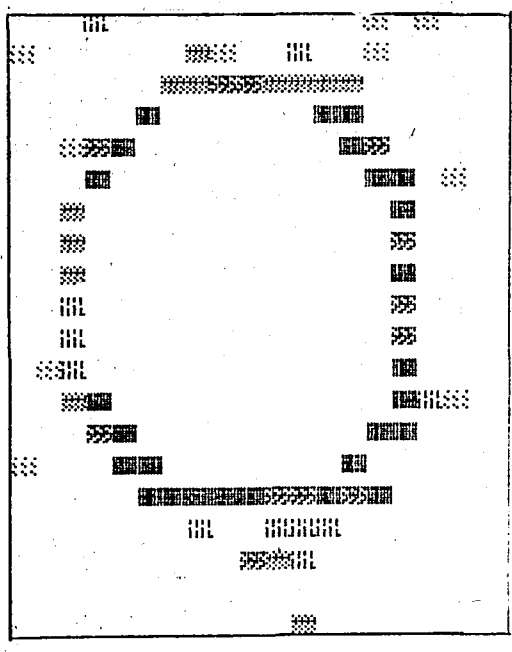
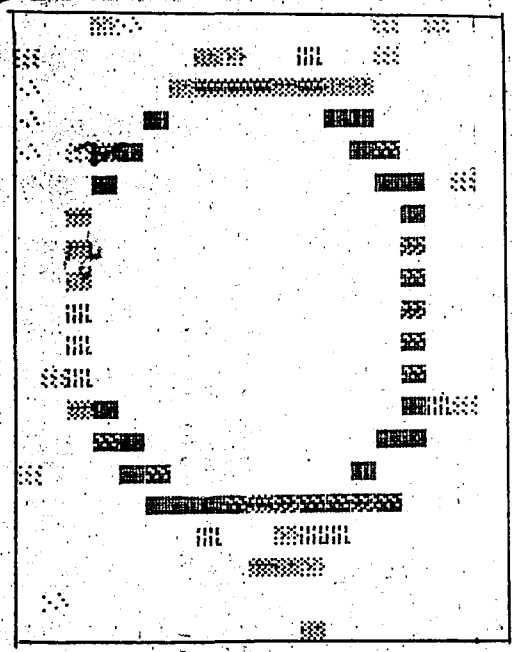
Fig 4.22 (Continued) f) Enhan.image, $k=32$ g) Thr.map, $k=32$ h) Enhan.image for $k=8*4*2*1$ i) Thr.map, $k=8*4*2*1$ j) Enhan.image, $k=32*16*8*4*2*1$ k) Thr.map, $k=32*16*8*4*2*1$.



a

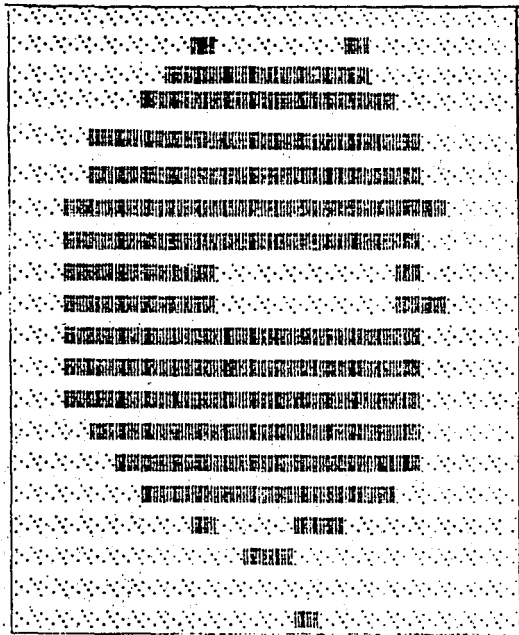
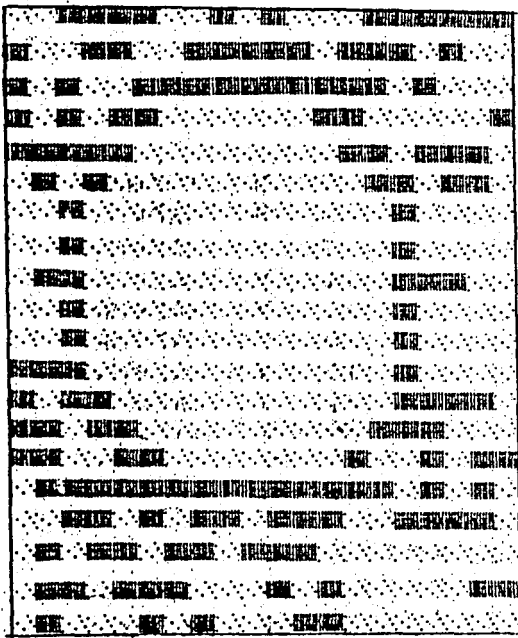


b-c

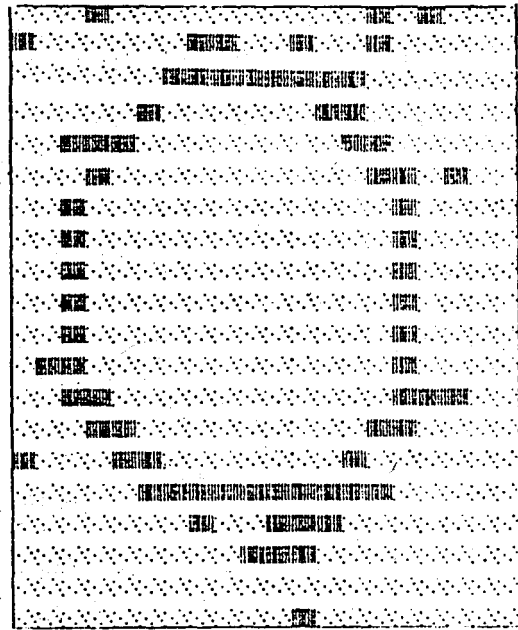
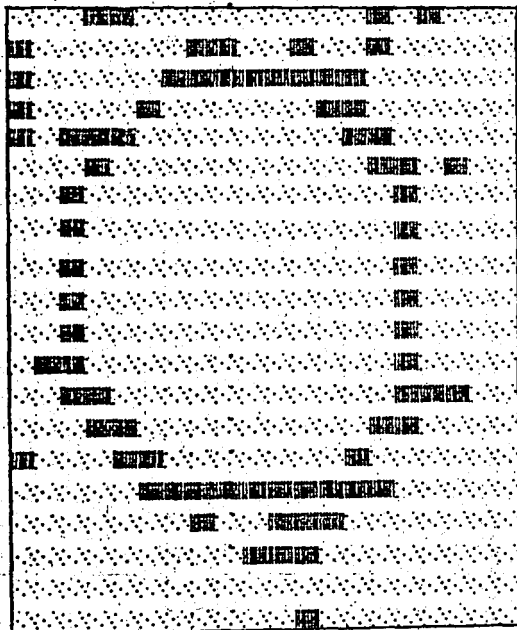


d-e

Fig 4.23 Simulation results of Rosenfelds algorithm with fixed thr. using a circular edge a)Original image b)Enhan.image,k=1 c)Enhan.image,k=8 d)Enhan.image,k=8*4*2*1. e)En.Im,k=32*16*8*4*2*1



f - g



h - i

Fig 4.23 (Continued) f) Thr. map, k=1 g) Thr. map, k=8 h) thr. map, k=8*4*2*1 i) Thr. map, k=32*16*8*4*2*1

Thresholding is done by a fixed threshold whose value is determined as in the preceding edge detection techniques. In section 4.5 evaluation of each scheme and comparisons will be done. In this project the following about edge detection techniques are examined:

- a) Kirsch masks with fixed threshold without connectivity test.
- b) Kirsch masks with fixed threshold with connectivity test.
- c) Kirsch masks with adaptive threshold without connectivity test.
- d) Kirsch masks with adaptive threshold with connectivity test.
- e) Sobel operator with fixed threshold without post-processing operation.
- f) Sobel operator with fixed threshold with post-processing operation.
- g) Sobel operator with adaptive threshold without post-processing operation.
- h) Sobel operator with adaptive threshold with post-processing operation.
- i) Rosenfeld diff.equations with fixed threshold for $k=1,2,4\dots32$
- j) Rosenfeld diff.equations with fixed threshold for $k=1.2,1.2.4,\dots,1.2\dots32.$

The comparison of each method with all the other methods is very difficult, because

- (1) - The outputs are not in the same form.
- (2) - Different thresholding techniques effect the preprocessing schemes in different ways and cause some difficulties in getting correct decisions.
- (3) - There are more than 20 different methods and they cause very long and useless comparison procedures.

Therefore, each of the preprocessing methods will be evaluated in itself by taking into consideration all thresholding and post-processing methods which

were discussed in the preceding sections.

In order to put the outputs into a standard form, the preprocessing schemes are compared by taking into account only the fixed thresholding technique and neglecting adaptive thresholding and post-processing.

4.4. PERFORMANCE CRITERIA

In this project, various quantitative and qualitative evaluations have been taken into consideration. This performance criteria can be listed as follows.

a) Percentage error versus contrast

For each method, this criterion will be evaluated by calculating the percentage error while increasing the contrast values of the test image. Contrast value, as mentioned earlier is $C = D1 - D3$, where $D1$ and $D3$ are mean of pixel magnitudes in zone 1 and 3, respectively. Contrast values are taken to be 2, 5, 10, 20, 30, 50, 80, 100 for each method. Percentage error is calculated by counting the 1's in the edge map, which are out of the edge region and adding the uncovered rows, i.e. missing edge points, to this. Then

$$E = \% \text{ Error} = \frac{\# \text{ of } 1\text{'s out of the edge reg.} + \# \text{ of uncov. rows}}{\# \text{ of pixels in the whole picture}} \quad (4.11)$$

This criterion is applied to each method both with and without the existence of noise and is also used for the comparison of the above methods.

b) Percentage error versus noise variance

For a given fixed contrast, say $C=20$, percentage of error, E , is calculated for different noise variances, σ . (Example, $\sigma = 0.2, 2, 5, 10, 20, 30, 40, 50, 100$).

c) Parameter P

As suggested by Fram and Deutsch [88], a parameter P can be calculated as one more criterion in order to compare the behaviour of various edge detection schemes with respect to noise. P is calculated by

$$P = \frac{\text{\# of errors in a pure signal}}{\text{\# of errors in a noisy signal}} \quad (4.12)$$

For a given fixed noise variance ($\sigma = 40$), variations of P are calculated as the function of the contrast.

d) Mean square distance of errors

This parameter is calculated by taking the mean of distances of the 1's in the edge map which are out of the edge region.

$$D = \frac{\text{square distance of error 1's to the edge region}}{\text{Total \# of 1's out of the edge region}} \quad (4.13)$$

This parameter is used to decide whether the 1's out of the edge region are near the edge, i.e., are only blurring the edge or not. This criterion is applied to both noisy and pure signals. This criterion can give an idea about the distribution of the errors.

e) Missing edge points

This parameter shows the uncovered rows in the edge region and is applied to both noisy and pure signals.

f) Operation times of each scheme are considered as an important criterion. The complexity of the algorithm is hence taken into account for each scheme.

g) Another parameter is the ability to detect curved edges. Circular images are used for this purpose. Only qualitative evaluation is done for this criterion.

h) Another qualitative evaluation is the ability to detect sharp edges. An edge region for each test image is defined. This region contains two pixels in each row of the picture, but sharp edges must contain only one pixel in each row.

4.5 EVALUATION OF RESULTS

As mentioned earlier, because of long operation time of the computer and the great number of edge detection schemes, all simulations had to be performed by using a minimum number of data (i.e. 20*20 samples), in order to evaluate each performance criterion. Each simulation was only performed three times because of the above reasons.

In the following, each performance criterion is evaluated for each preprocessing scheme with respect to the various postprocessing algorithms and thresholding techniques. In some cases the evaluation is done with the existence of noise. In addition all preprocessing schemes using a fixed threshold are compared with respect to all criteria.

PERCENTAGE ERROR VERSUS CONTRAST

A - SOBEL OPERATOR - Four different types of edge detection schemes using the Sobel operator are evaluated:

- 1 - Sobel operator with fixed threshold and without postprocessing algorithm
- 2 - Sobel operator with fixed threshold and with postprocessing algorithm
- 3 - Sobel operator with adaptive threshold and without postprocessing algorithm
- 4 - Sobel operator with adaptive threshold and with postprocessing algorithm

In Fig (4.24) four different schemes are shown without the existence of noise. It is obvious from Fig (4.24) that increasing the contrast value does

SOBEL OPERATOR

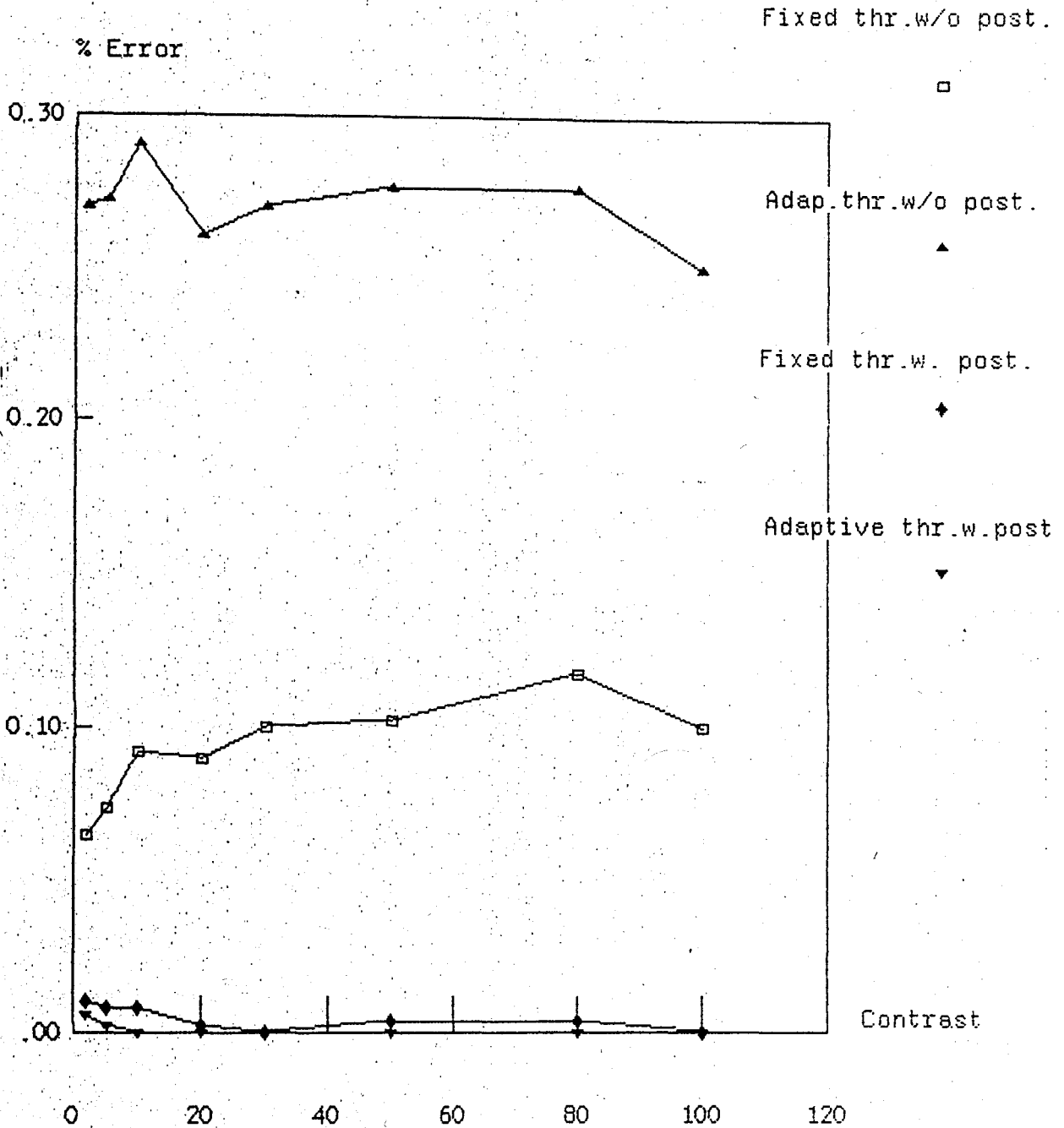


Fig.4.24 % error vs contrast for adaptive & fixed threshold, with and w/o postprocessing, without the existence of noise

not result in a decrease in the percentage error. This can be expected because, when the contrast increases, the number of 1's, i.e., points which are above the threshold value, increases on one side of the edge, while it decreases on the other side. Because the threshold value depends on the local or total average variances of the picture, this criterion does not give any idea about the behaviour of the detection scheme. But, it can be seen that the postprocessing algorithm, i.e., limiting the number of 1's in the picture, gives pretty good results. Fixed threshold gives better result than adaptive threshold without using postprocessing because, the local threshold permits some local edges to appear which are not required in our picture. This postprocessing algorithm can be applied to all other edge detection schemes. But in this study postprocessing was used only for the Sobel operator.

In Fig (4.25) the same criterion is evaluated with the existence of noise. More reasonable results are taken from this analysis. In a noisy environment with $\sigma = 40$, the percentage error decreases with increasing contrast values. This is shown in Fig (4.25). The postprocessing algorithm again gives very good results. In fact, for contrast values greater than 30 the error disappears. That is, there are no 1's except the 1's in the edge region.

For $\sigma = 40$, the noise influences the results for values upto $C=50$. For greater contrast values the noise has no effect on the scheme. Thus, for smaller noise values the effect of the noise disappears at lower contrasts than $C=50$. Again, because we don't want to detect subedges, fixed threshold gives better results than the adaptive threshold.

In Fig (4.26), three pictures (vertical edges), and the detected edges by

SOBEL OPERATOR

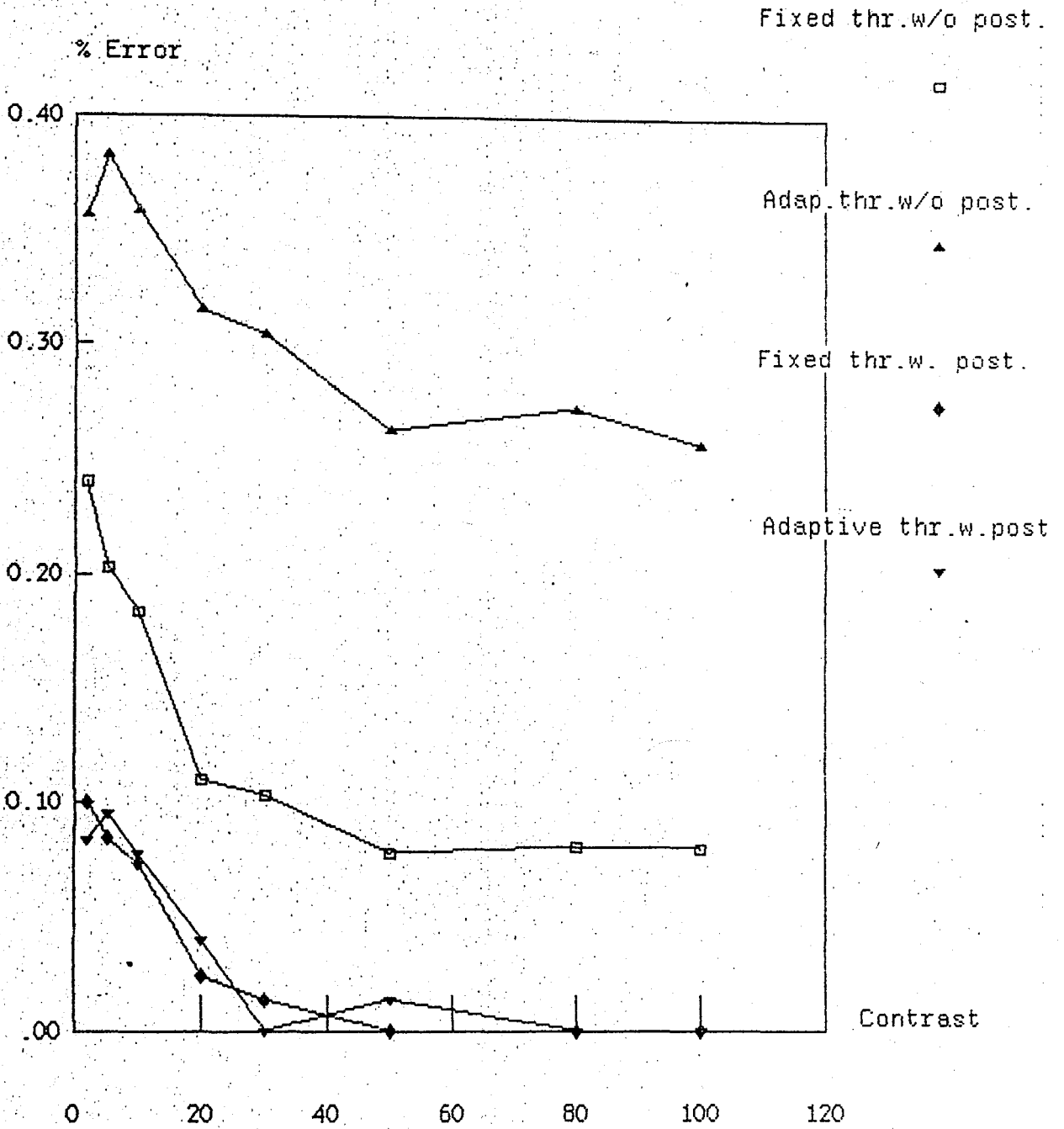
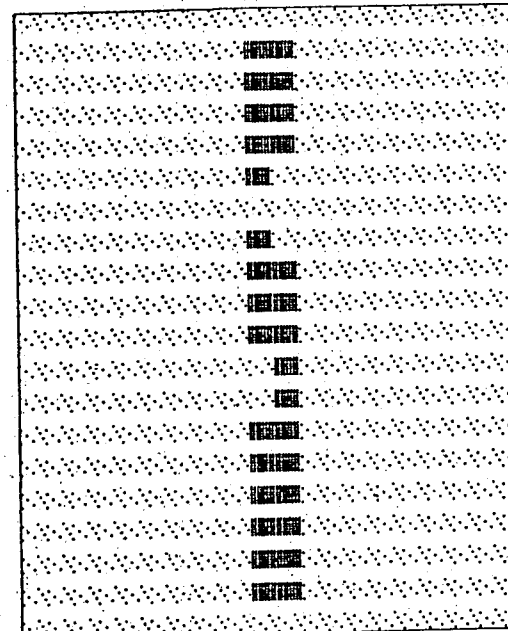
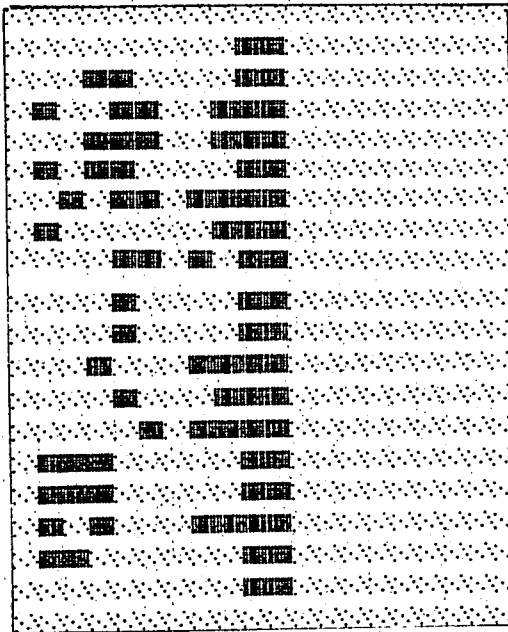
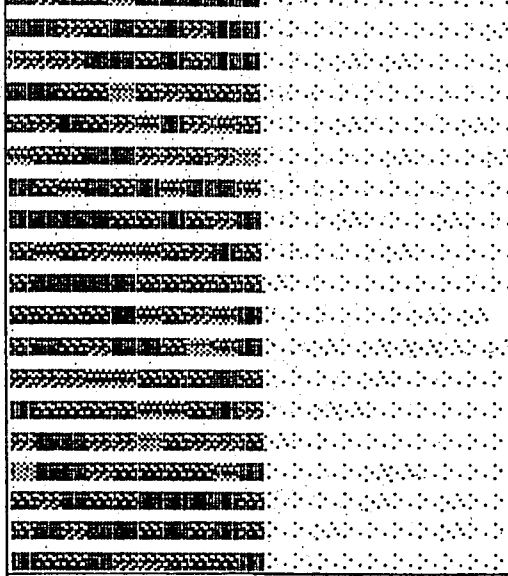


Fig.4.25 % error vs contrast for adaptive & fixed threshold, with and w/o postprocessing, with the existence of noise($\sigma = 40$)



a

Fig 4.26 Simulation result of Sobel operator:Original image, thresholded image with fixed threshold and output of postprocessing operation respectively a)w/o existance of noise,C=20

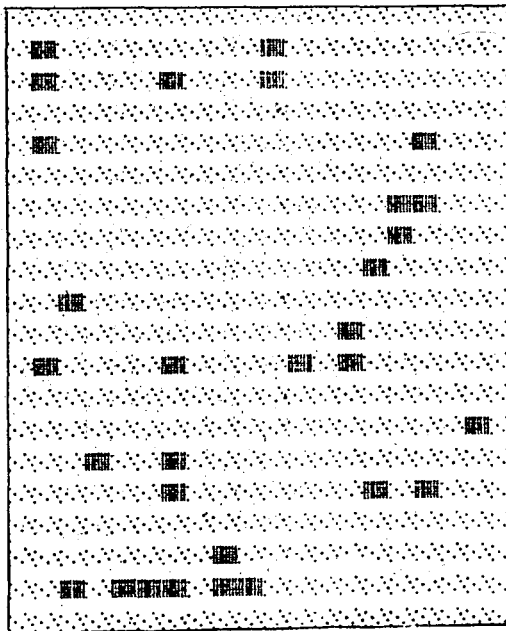
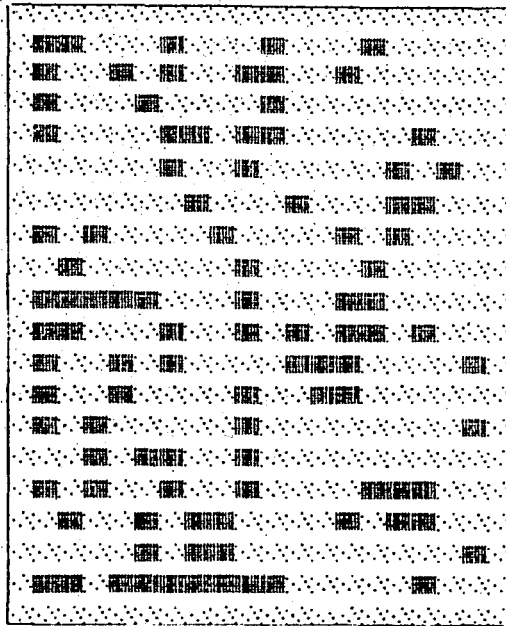
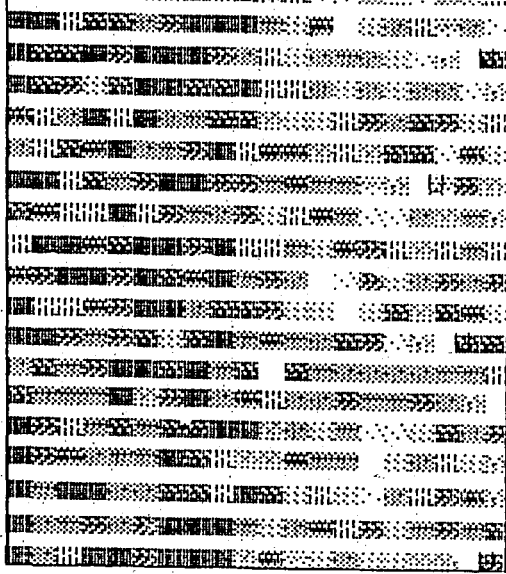


Fig 4.26 (continued) c) with noise, $\sigma=100, C=20$

using fixed threshold, with and without postprocessing procedure (i.e., limitation of 1's) are shown for contrast $C=20$. Fig (4.26a) shows the pure, Fig (4.26b) shows the noisy image with $\sigma=40$, and Fig (4.26c) shows another noisy image with $\sigma=100$.

B - KIRSCH MASKS

Four different types of edge detection schemes are evaluated for Kirsch directional masks.

- 1 - Kirsch masks with fixed threshold and without postprocessing.
- 2 - Kirsch masks with fixed threshold and with postprocessing.
- 3 - Kirsch masks with adaptive threshold and without postprocessing.
- 4 - Kirsch masks with adaptive threshold and with postprocessing.

In Fig (4.27) four different schemes are shown without the existence of noise. As in the case of the Sobel operator, increasing the contrast value does not result in an increase in percentage error because of the same reason. A different postprocessing algorithm, the connectivity test, is used for this scheme as explained in section 4.3. Although this parameter does not give any idea about the individual behaviour of each scheme, one can get a comparative result from Fig (4.27). Connectivity test for fixed threshold gives better results. But the locally adaptive threshold by convolving each pixel with the low pass filter, M_0 (see section 4.3) without connectivity test gives the best result, because the output of the convolution is the sharpest edge, and when the connectivity test is applied to this output some edge points are missed and uncovered rows appear in the edge region. That is why the connectivity test is not good for Kirsch masks. If a fixed threshold is used, better results can be obtained by applying connectivity test but for adaptive thresholding there is no need for postprocessing procedures.

KIRCSH MASKS

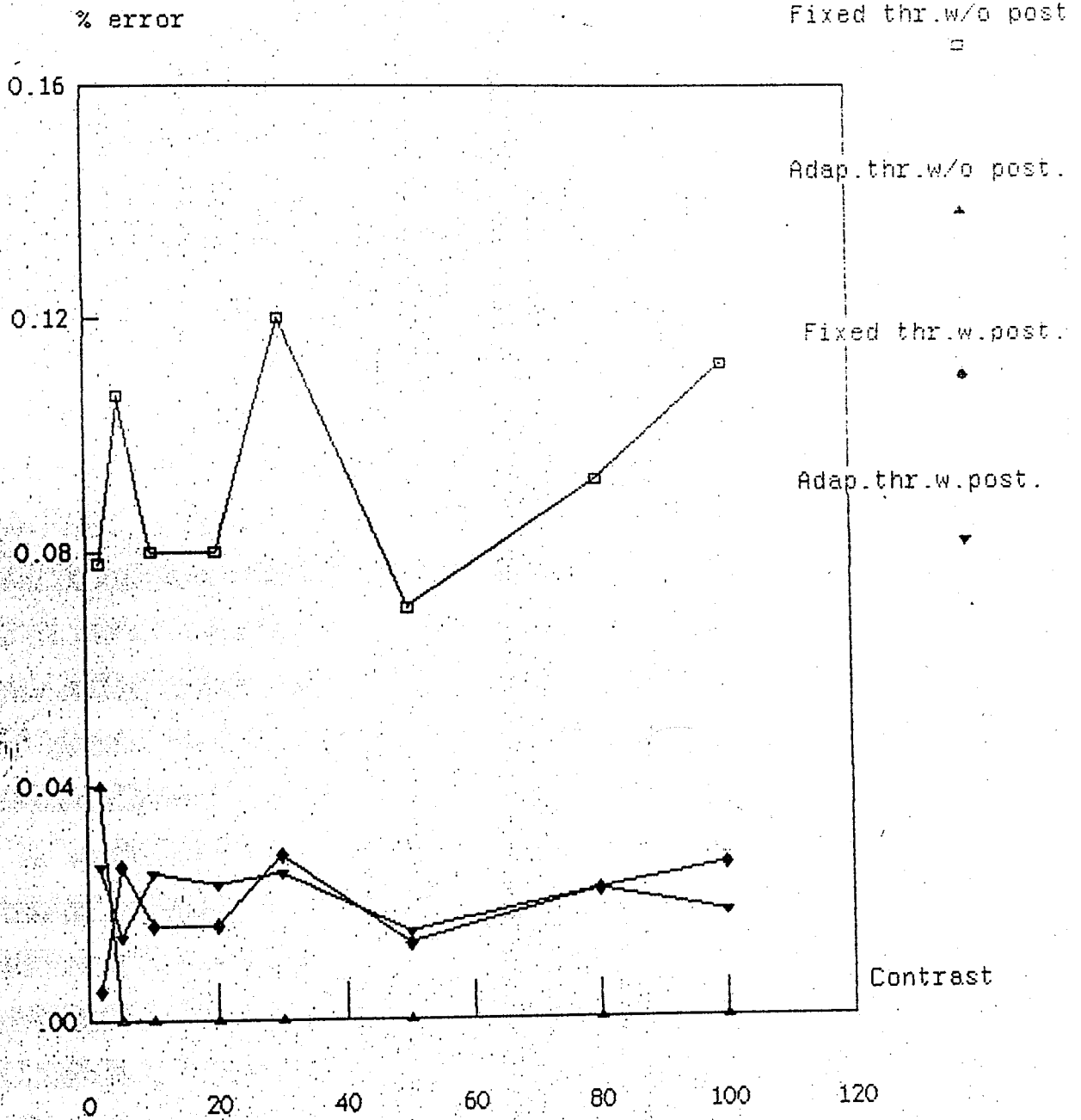


Fig.4.27 % error vs contrast for fixed & adap.threshold, with & w/o postprocessing, without existence of noise

Fig (4.28) shows the percentage error versus contrast with the existence of noise. The behaviour of the schemes can be understood easily from Fig (4.28). As in the case of the Sobel operator, the noise is added to the original picture with the variance $\sigma = 40$. Again, the best result is obtained with an adaptive threshold and without connectivity test. Connectivity test effects the scheme with adaptive thresholding in a negative manner, but for fixed threshold it can be used to get better results.

Fig (4.29) shows three pictures (with vertical edges) with contrast $C=20$, and the detected edges by using fixed threshold without and with postprocessing (i.e., connectivity test) algorithm. Fig (4.29a) shows the pure image, (4.29b) the noisy image with variance $\sigma = 40$, (4.29c) the noisy image with noise variance $\sigma = 100$.

C - ROSENFELD DIFFERENCE EQUATIONS

The simulation of this method is done for each value of k where

$k=1,2,4,\dots,32$ and the products of these, i.e.,

$k=2*1,4*2*1,\dots,32*16*8*4*2*1$. But the evaluation has been done for:

- 1 - Rosenfeld difference equation with fixed threshold, for $k=1$
- 2 - Rosenfeld difference equation with fixed threshold, for $k=8$
- 3 - Rosenfeld difference equation with fixed threshold, for $k=32$
- 4 - Rosenfeld difference equation with fixed threshold, for $k=8*4*2*1$
- 5 - Rosenfeld difference equation with fixed threshold, for $k=32*16*8*4*2*1$

The postprocessing procedure is not used for this scheme. Fig (4.30) shows the results for $k=1,8,32$ and $k=8*4*2*1,32*16*8*4*2*1$ with fixed threshold, without any postprocessing algorithm and without the existence of noise. As

KIRCSH METHOD

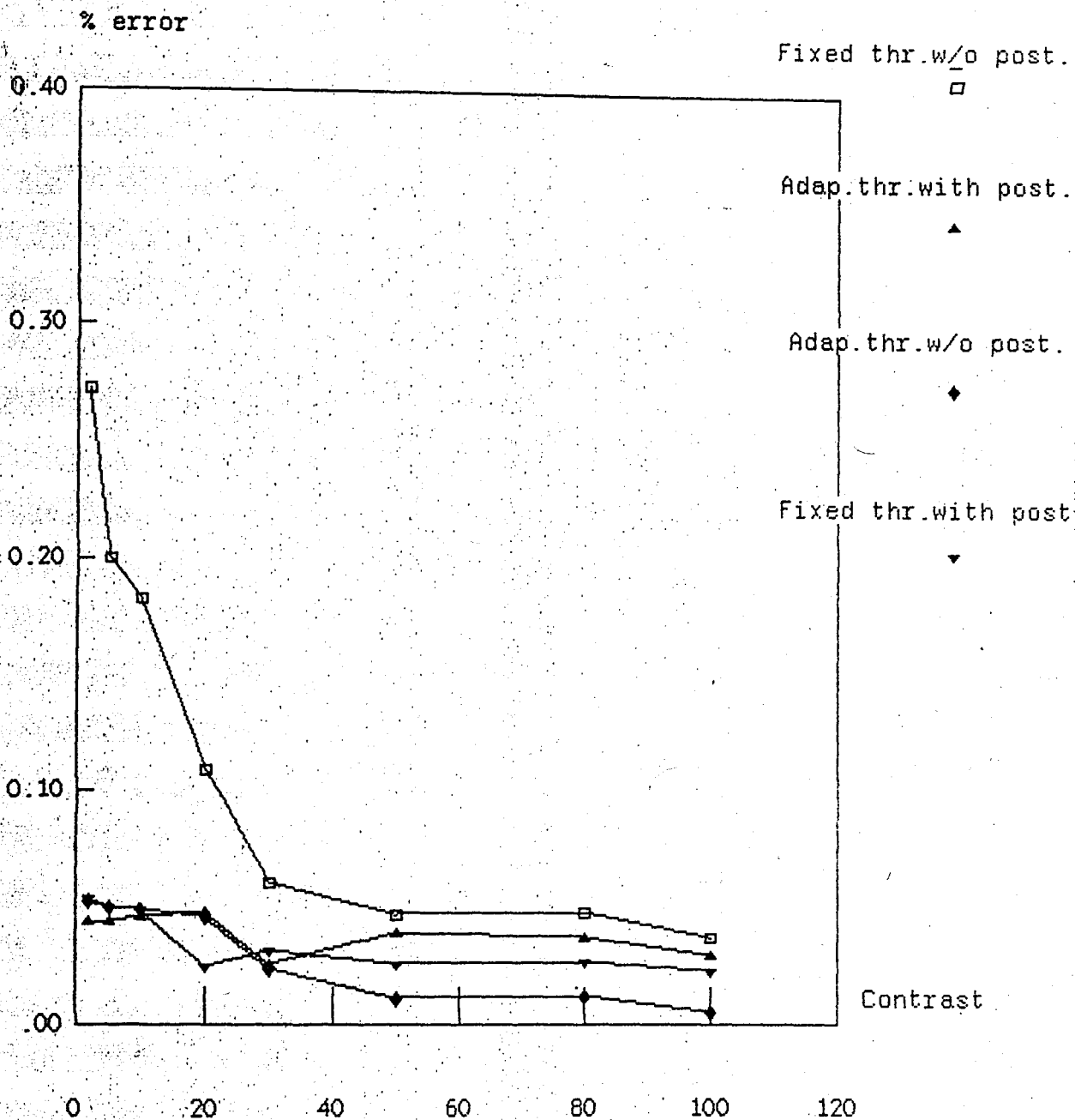
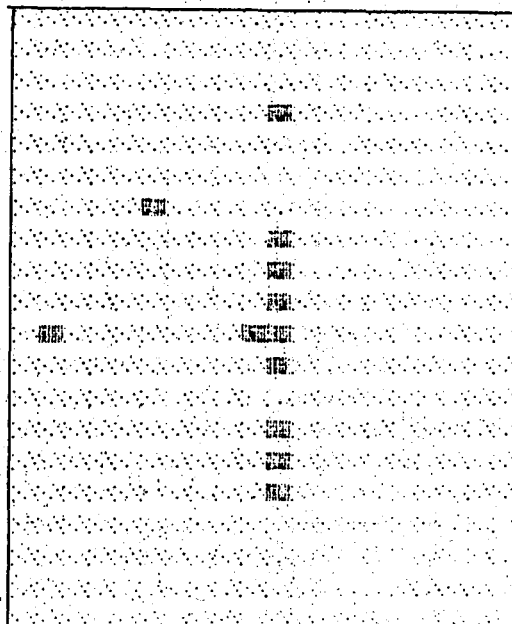
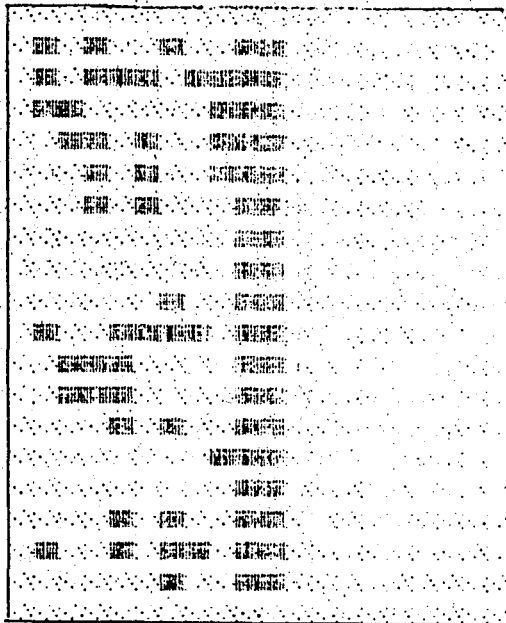
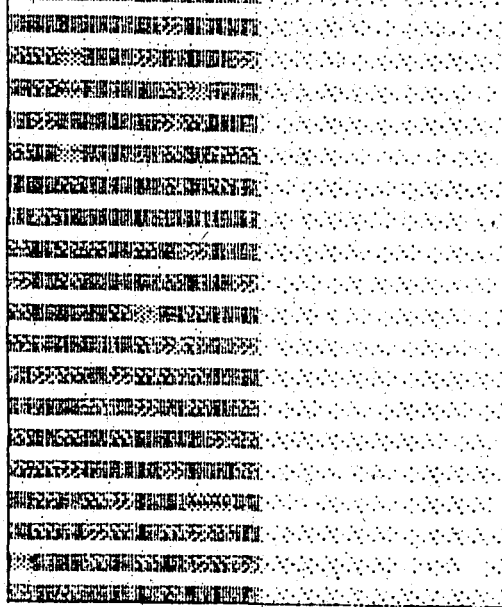


Fig. 4.26 % error vs contrast, with the existence of noise ($\sigma=40$) for fixed & adap. threshold, with & w/o existence of postproc.



a

Fig 4.29 Simulation results of Kirsch masks:Original image, thresholded image with fixed threshold and output of postprocessing operation respectively a)w/o noise, $C=20$ 232

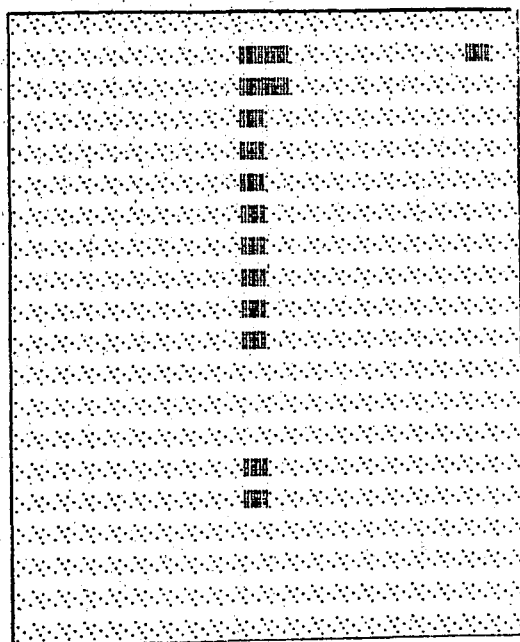
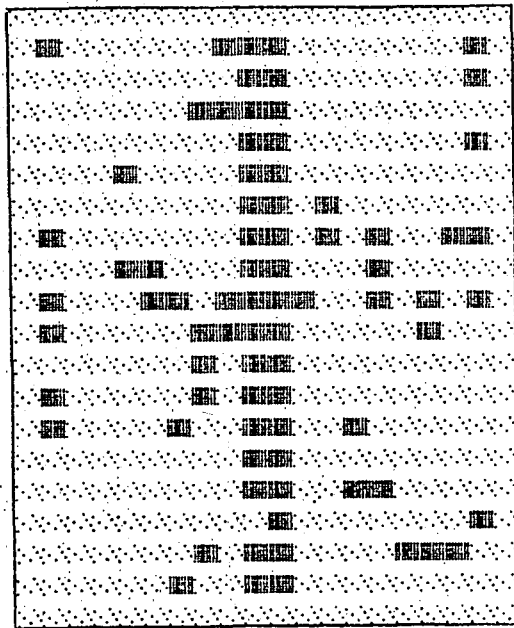
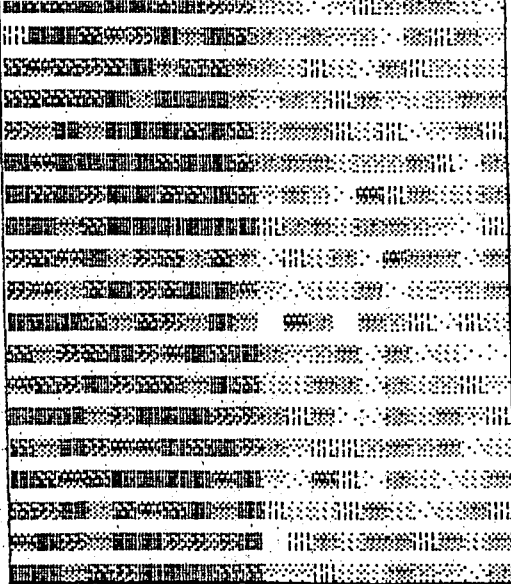


Fig. 4.29 (continued) b) with noise, $\sigma = 40, C = 20$

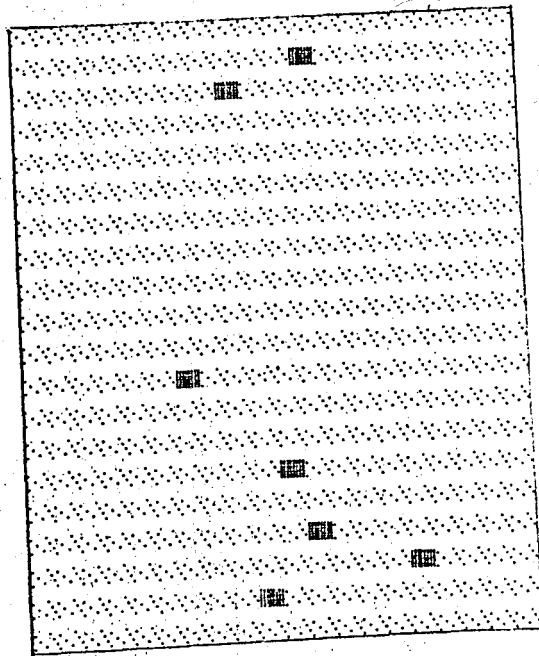
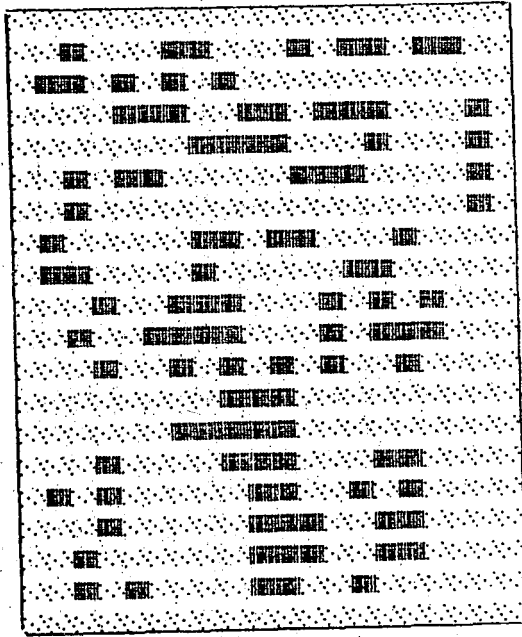
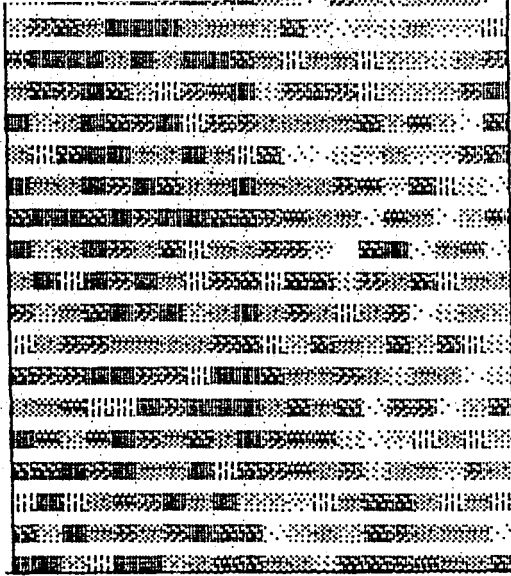


Fig 4.29 (continued) c) with noise, $\sigma=100, C=20$

ROSENFELD DIFF. EQUATIONS

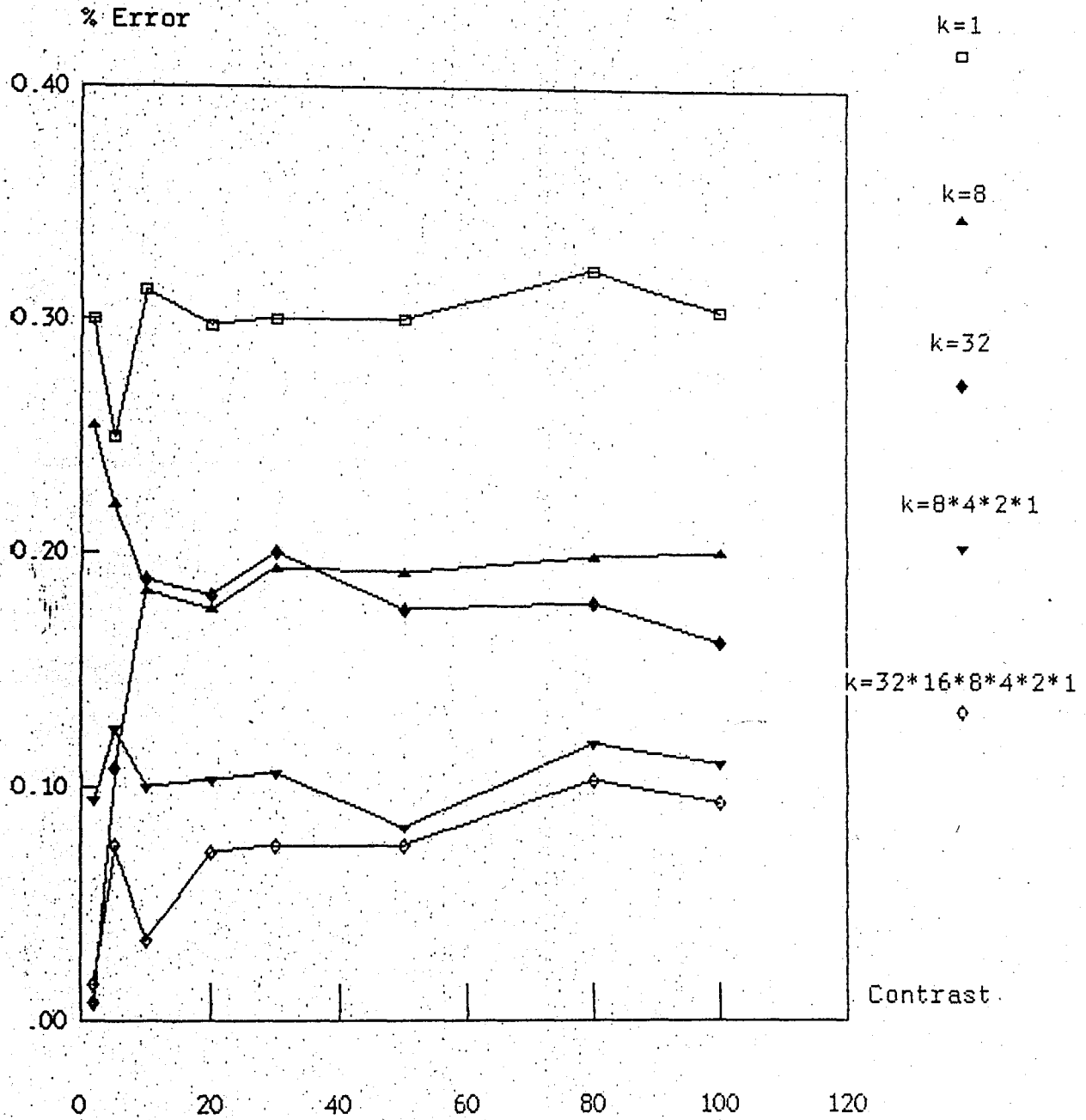


Fig. 4.30 % error vs contrast for fixed threshold,

w/o postprocessing, & without the existence of noise

in the case of the preceding schemes, percentage error is not a function of the contrast values. But a comparison can be done between the alternative schemes. For $k=1$ the worst and for $k=32*16*8*4*2*1$ the best result is obtained as expected.

Fig (4.31) shows the same schemes with the existence of noise ($\sigma=40$). with noise the percentage error decreases versus the increase in contrast. When the two graphs (Fig (4.30) and (4.31)) are compared, the values in (4.31) are worse than the values in (4.30). The noise is still effective even for high contrast values. Again the worst result is obtained when $k=1$ and the best result is obtained when $k=32*16*8*4*2*1$. Fig (4.32) shows three pictures with contrast value $C=20$ and the detected edges by using fixed threshold for $k=8*4*2*1$. Fig (4.32a) shows the pure image, Fig (4.32b) the noisy image with $\sigma=40$, Fig (4.32c) the noisy image with $\sigma=100$. Fig (4.19) and (4.32) illustrate the fact that for large k the edge becomes more and more conspicuous but the larger the k the less precisely localized is the detected edge. If the differences are multiplied together for a range of k , the result tends to yield sharply localized detection of the edges.

D - THE COMPARISON OF THE THREE PREPROCESSING SCHEMES

The comparison is done using only the fixed threshold and without taking any postprocessing algorithm in order to look at each scheme in equal conditions. Evaluation is performed for:

- 1 - Sobel operator with fixed threshold
- 2 - Kirsch masks with fixed threshold
- 3 - Rosenfeld difference equations with fixed threshold for $k=8$
- 4 - Rosenfeld difference equation with fixed threshold for $k=32*16*8*4*2*1$

ROSENFELD DIFF. EQUATIONS

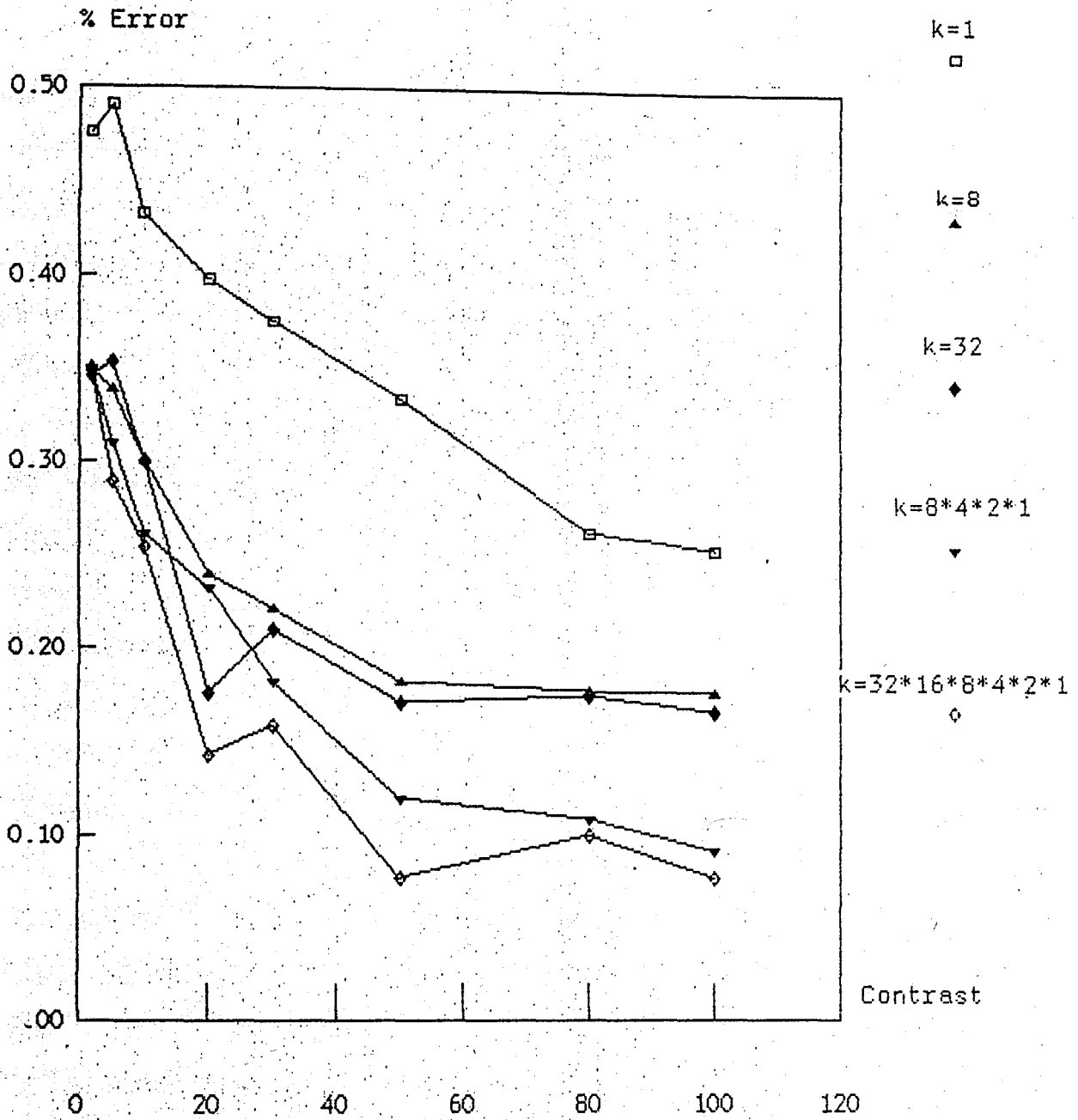
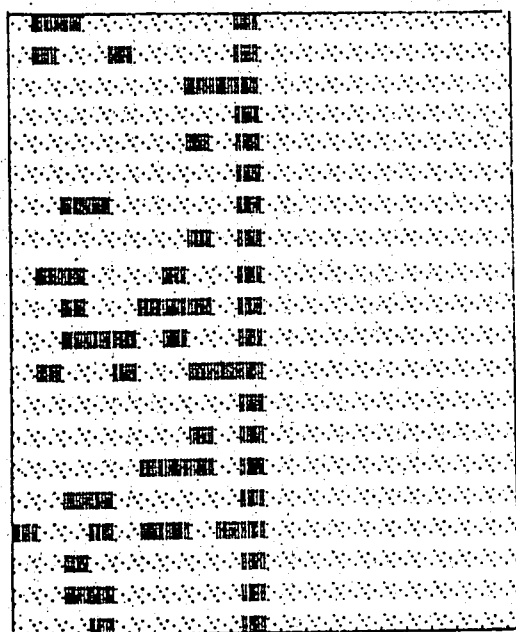
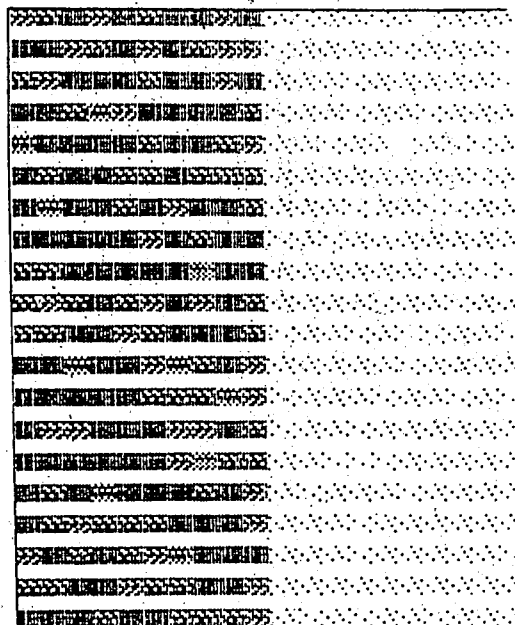


Fig.4.31 % error vs contrast for fixed threshold, and w/o postprocessing, with the existence of noise ($\sigma = 40$)



a

Fig 4.32 Simulation results of Rosenfelds algorithm:Original image, thresholded image using fixed threshold and output of postprocessing operation, respectively a)w/o noise, C=20

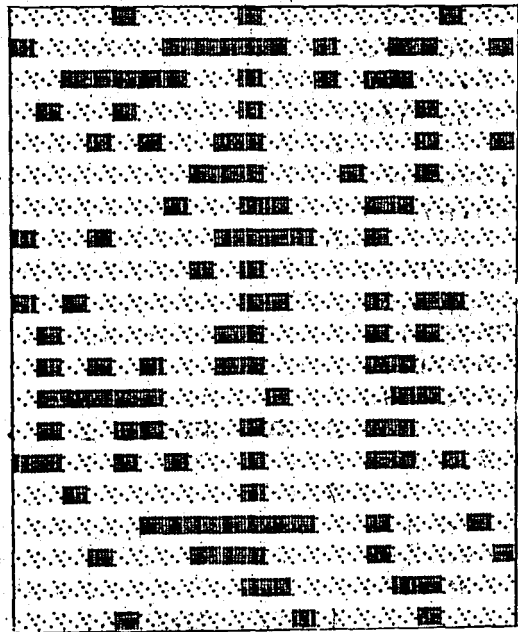


Fig. 4.32 (continued) b) with noise, $\sigma=40$, $C=20$

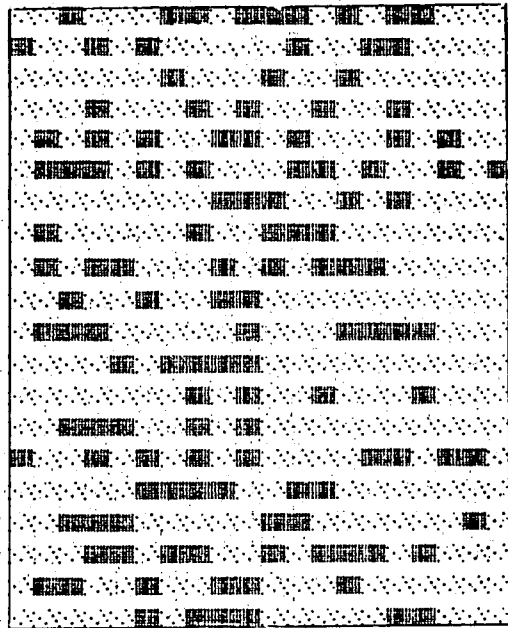
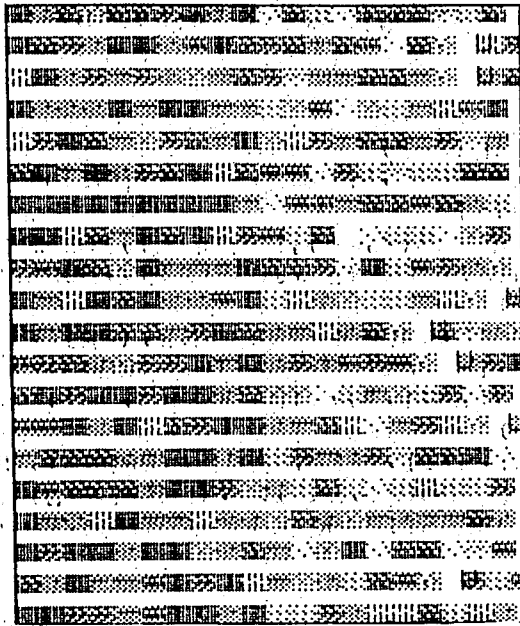


Fig 4.32 (continued) c) with noise, $\sigma=100, C=20$

Although this criterion does not show any steady behaviour, it can be used for comparison purposes. Sobel and Kirsch methods show almost the same characteristics. Rosenfeld's method with $k=8$ is the worst and with $k=32*16*8*4*2*1$ gives the best result (see Fig (4.33)). If Fig (4.34) is observed one can see the behaviour of the four schemes with the existence of noise for $\sigma=40$.

For small contrasts the behaviour of Sobel and Kirsch algorithms are again almost the same. But the effect of noise disappears faster by using Kirsch's method. For greater values than $C=50$ Sobel, Kirsch and Rosenfeld (for $k=32*16*8*4*2*1$) approximately shows the same characteristic but for the contrast values less than 50 Rosenfeld with $k=32*16*8*4*2*1$ cannot give as good results as Sobel and Kirsch methods do. The percentage error is very high for small contrasts. Rosenfeld for $k=8$ gives the worst result. Another disadvantage of Rosenfeld's scheme can also be observed from Fig (4.34). The effect of noise does not disappear even for large contrasts. Besides, the increase of the percentage error in a noisy environment is much greater. This can be seen by making the comparison of Fig (4.33) and Fig (4.34).

REMARKS:

1 - Although percentage error versus contrast is not a steady characteristic for any of these three preprocessing algorithms, however, it can be used for the comparison of different thresholding and postprocessing techniques using one of these preprocessing methods. Also it can be used for the comparison of the preprocessing methods among themselves provided that they are employed under the same conditions.

2 - Percentage error versus contrast with noise contamination puts more into

Comparison between Sobel Operator, Kirsch Masks,

Rosenfeld diff. eqn 1 & 2 for $k=8$ and $k=32 \dots 1$ respectively

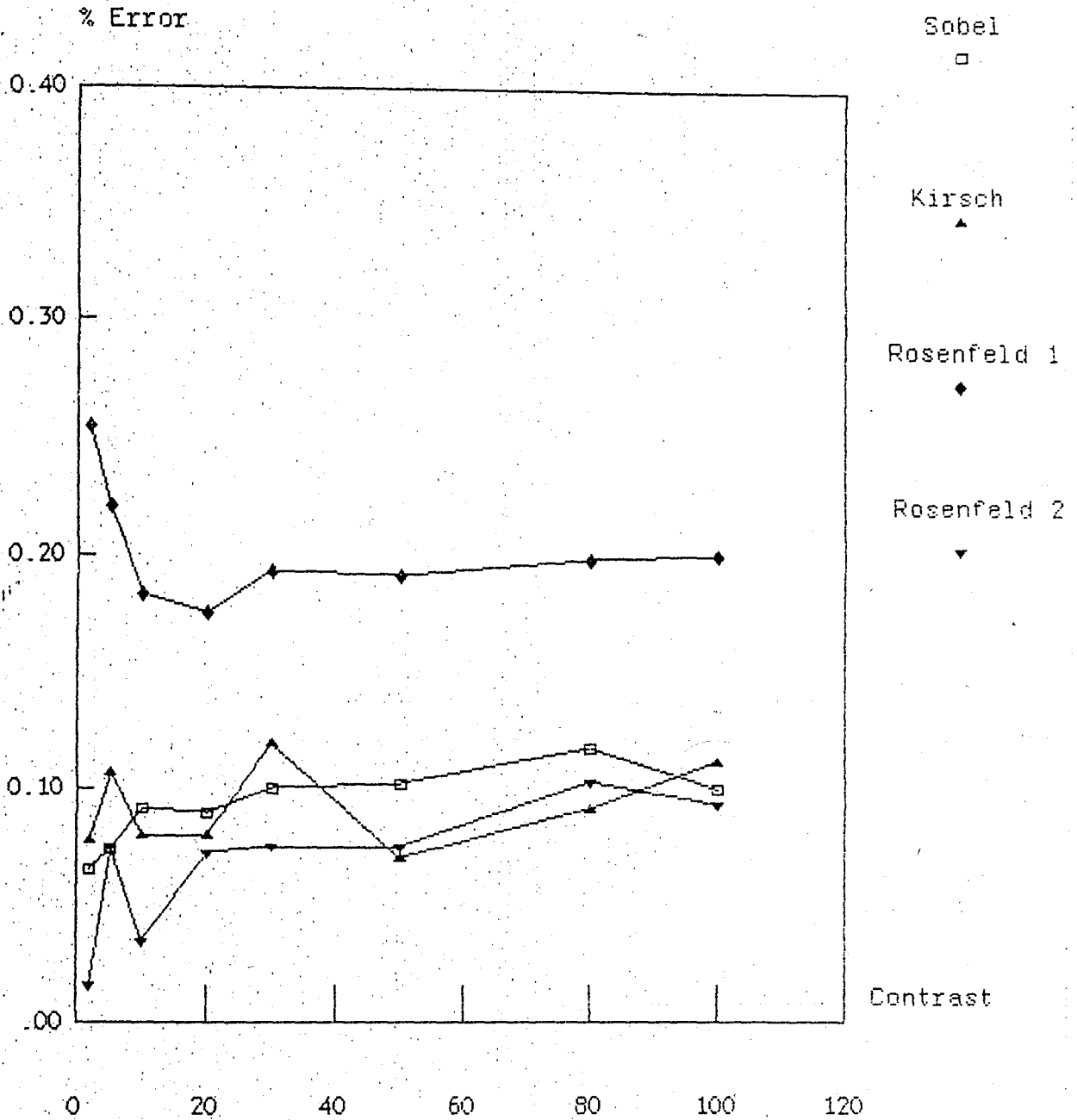


Fig.4.33 % error vs contrast for fixed threshold and

without postprocessing and without the existence of noise

Comparison between Sobel Operator, Kirsch Masks,

Rosenfeld diff. eqn 1 & 2 for $k=8$ and $k=32 \dots 1$ respectively

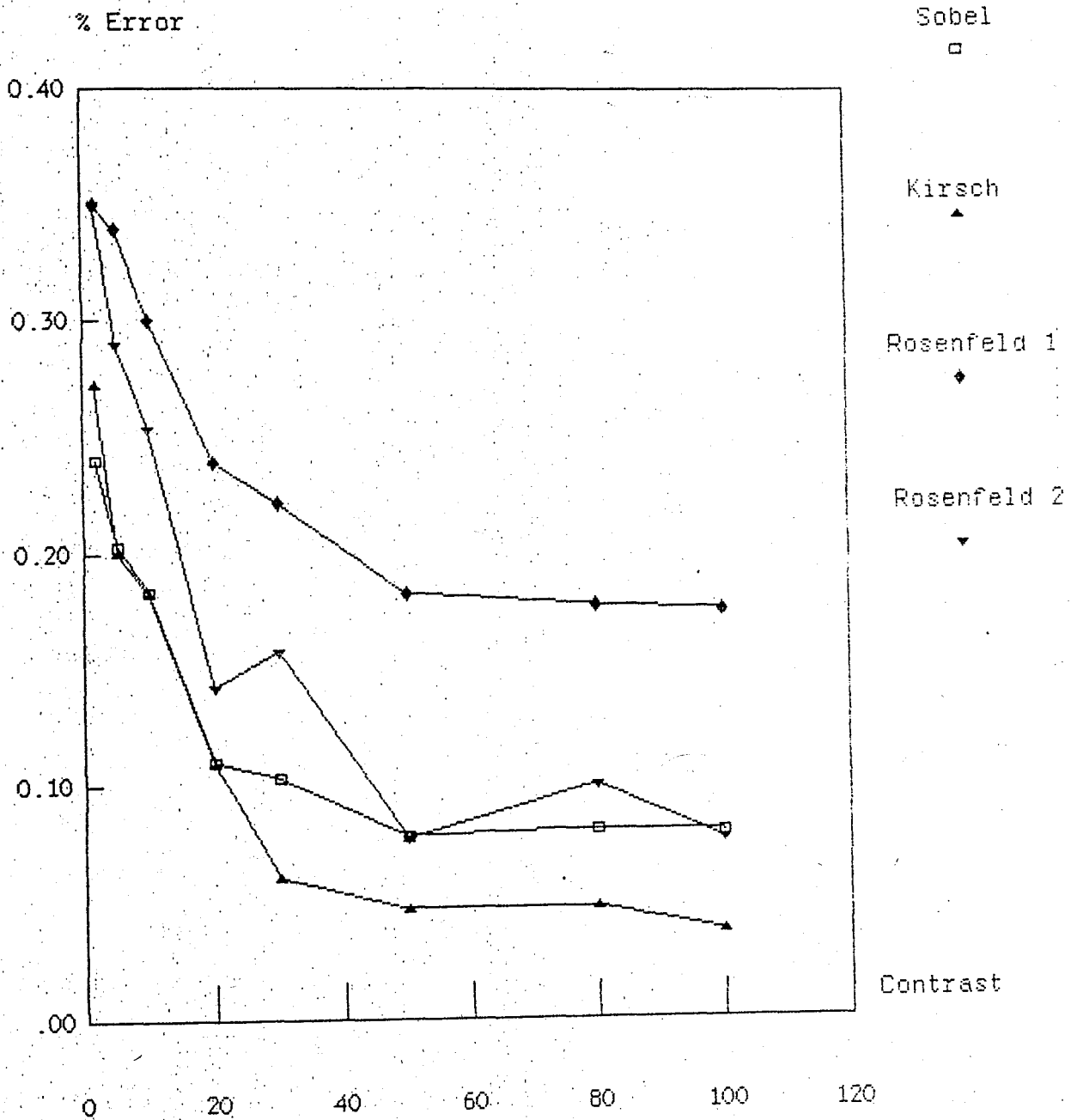


Fig.4.34 % error vs contrast for fixed threshold and without postprocessing and with the existence of noise ($\sigma=40$)

evidence characteristics about the detection schemes.

3 - Postprocessing algorithms, e.g., limiting the number of 1's in the picture can be applied to all schemes. The comparison of the effect of such postprocessing techniques on the various preprocessing algorithms has not been included in this study.

PERCENTAGE ERROR VERSUS NOISE VARIANCE

Evaluation is done for the following nine noise variance values: $\sigma = .2, 2, 5, 10, 20, 30, 40, 50, 100$. The contrast is fixed at $C=20$ for all simulations.

A - SOBEL OPERATOR

Four different types of edge detection schemes are evaluated for the Sobel operator just as has been done for the preceding criterion. The results are shown in Fig (4.35).

It can be seen from Fig (4.35) that the noise variance is directly proportional to the percentage error at a fixed contrast. Because the adaptive threshold permits the appearance of subedges which do not belong to the picture, the results are worst especially if postprocessing operation is not used. Up to $\sigma = 30$ (with $C=20$) the percentage error deteriorates slowly but for values greater than $\sigma = 30$ the increase becomes sharper. The best result is obtained when using adaptive thresholding with postprocessing operation.

Fig (4.36) shows the results of the three simulations using a fixed

SOBEL OPERATOR

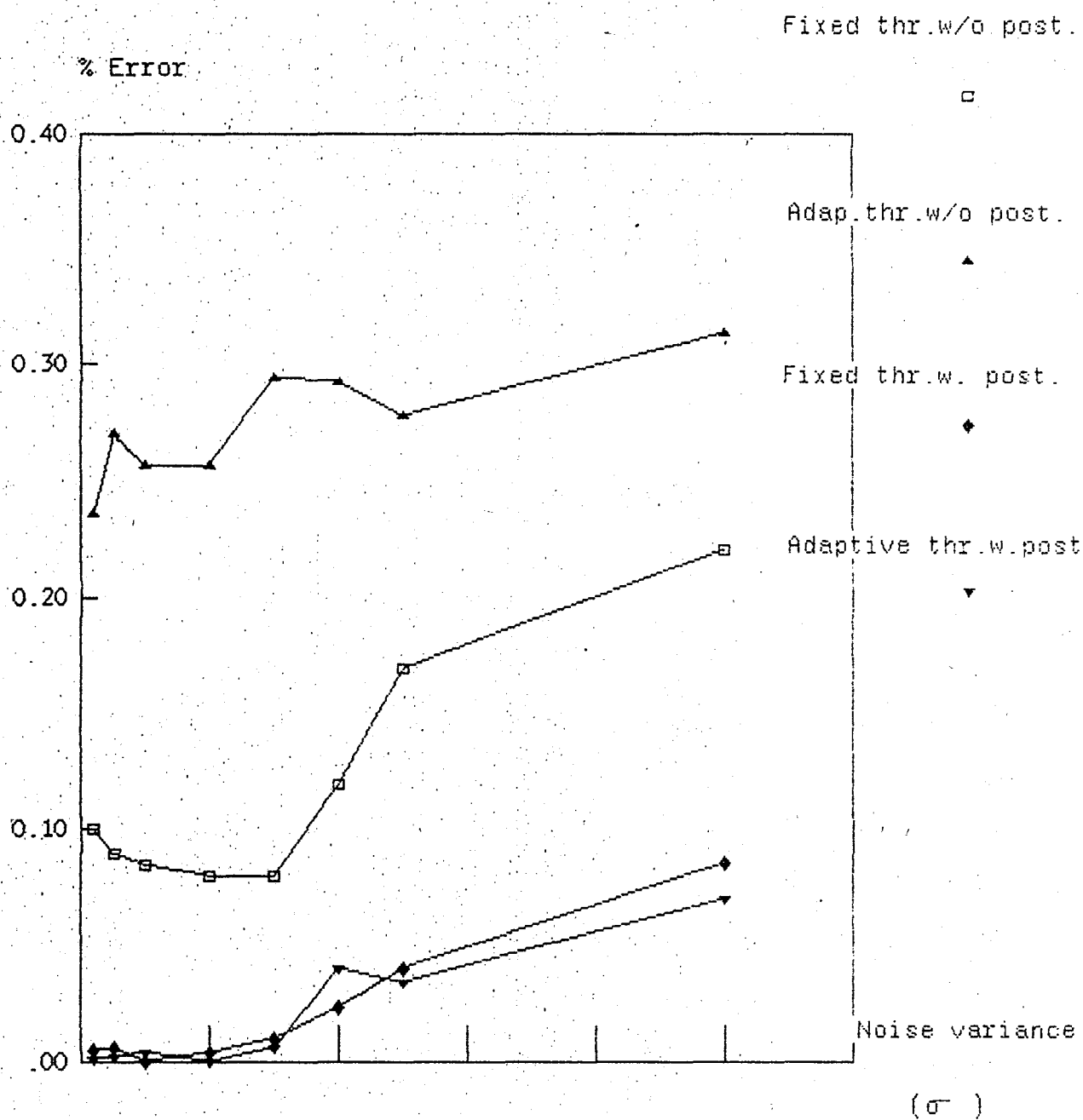


Fig.4.35 % error vs noise variance, σ for adaptive & fixed threshold, with and w/o postprocessing

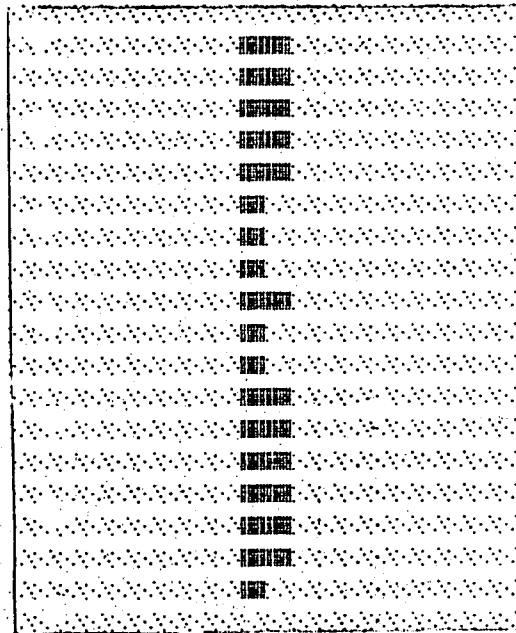
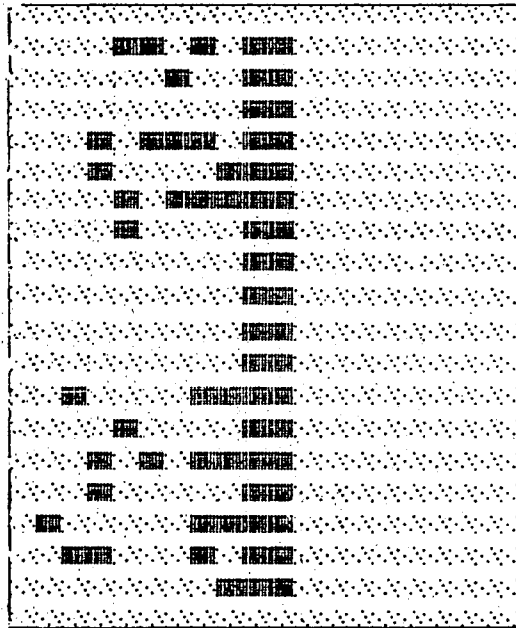
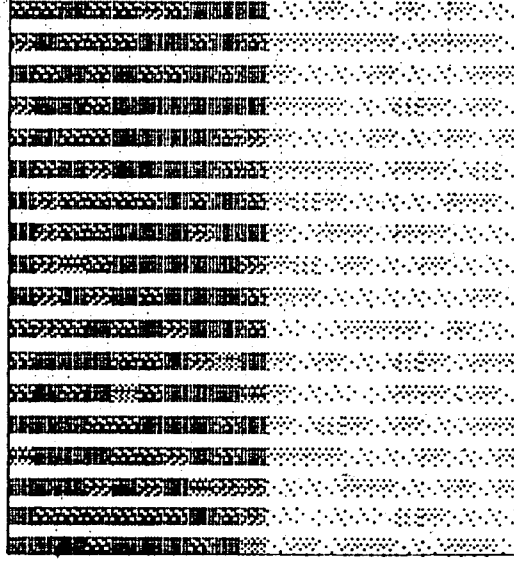


Fig 4.36 Simulation results of Sobel operator:Original image,thresholded image with fixed threshold and output of postprocessing operation, respectively a)with noise, $\sigma=5$, $C=20$

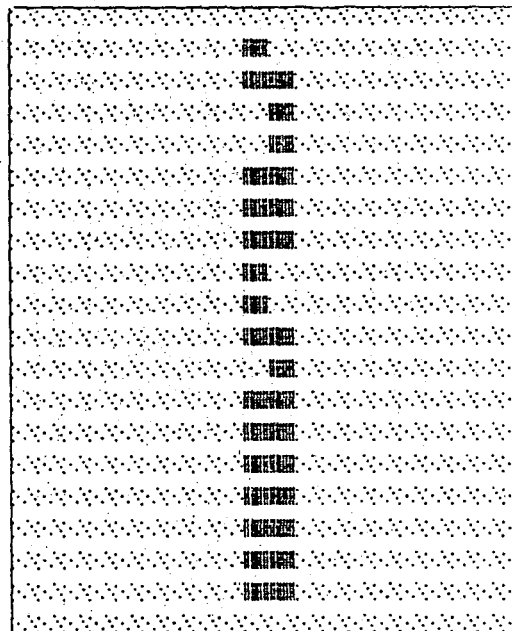
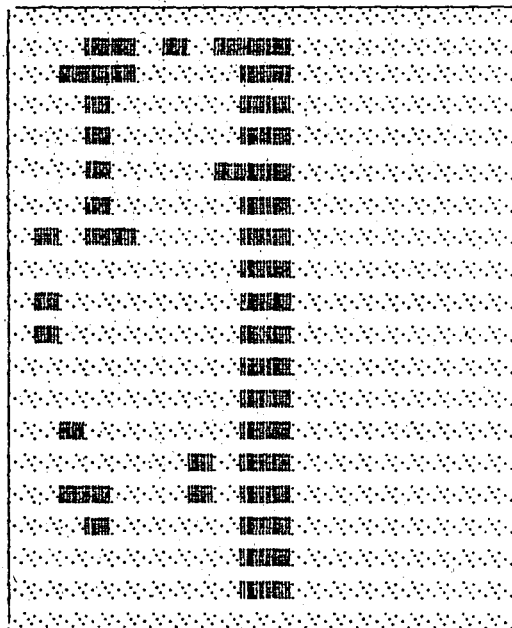
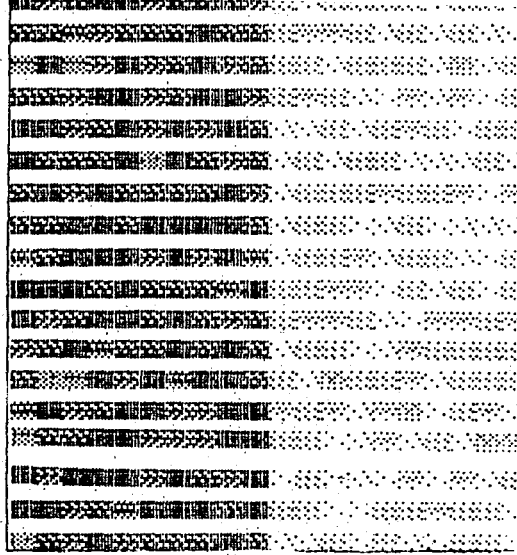


Fig. 4.36 (continued) b) with noise, $\sigma=10, C=20$

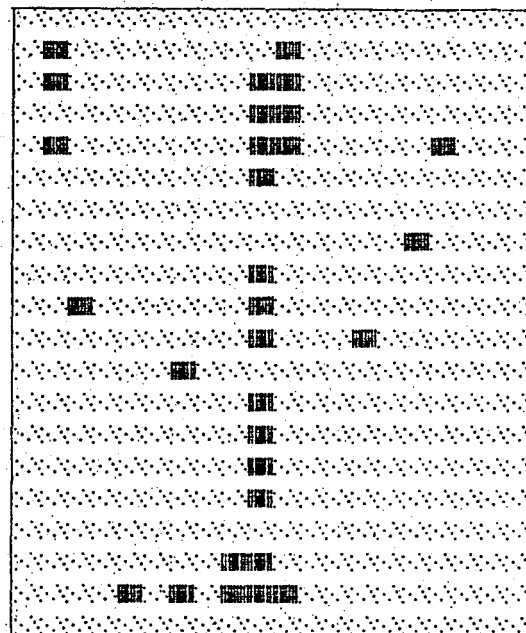
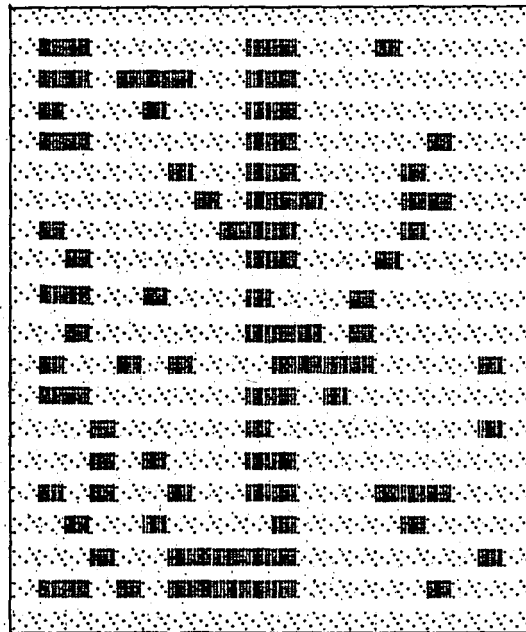
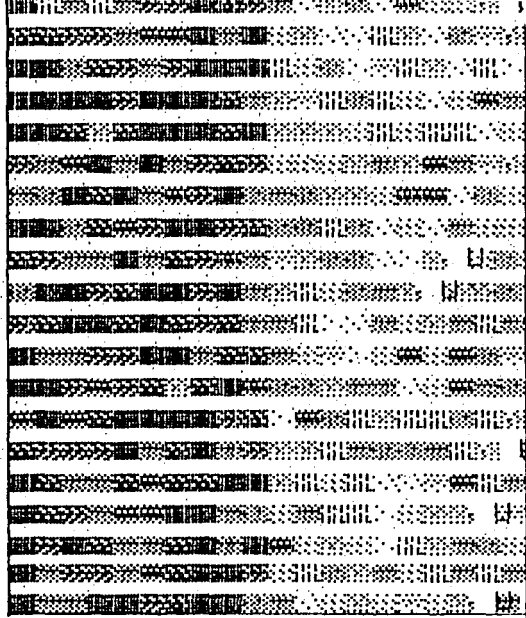


Fig 4.36 (continued) c) with noise, $\sigma=50, C=20$

threshold, one with noise variance $\sigma = 5$, the second with noise variance $\sigma = 10$ and the last one with $\sigma = 50$. In the first two pictures, edges can be easily gathered but in the third one the edge points can not be recognized from the output. For $\sigma = 100$ the result gets worse as shown in Fig (4.26).

B - KIRSCH MASKS

Again the same four schemes are used for the simulations. Fig (4.37) shows the results of how percentage error changes with respect to the changes in noise variance. The worst result is obtained if a fixed threshold is used without using connectivity test. If connectivity test is used with fixed threshold the result of the scheme is very similar to the one with adaptive threshold and without using the connectivity test. For small variances of noise adaptive thresholding without using postprocessing gives the best result. For this criterion, it can be said that for fixed thresholding a postprocessing algorithm is needed in order to get better results, but for adaptive thresholding there is no need for any postprocessing algorithm. Fig (4.38) shows three simulation results with noise variance $\sigma = 5, 10, 50$. The effect of noise and connectivity test can be easily seen from the figure. For $\sigma = 100$ the result gets worse as shown in Fig (4.29).

C - ROSENFELD DIFFERENCE EQUATIONS

For this method only five different schemes are evaluated, although all simulation results for each value of k , which is explained in section 3.3 and 4.3, have been obtained in order to get simpler and more understandable graphs as in the case of the preceding criterion.

KIRCSH MASKS

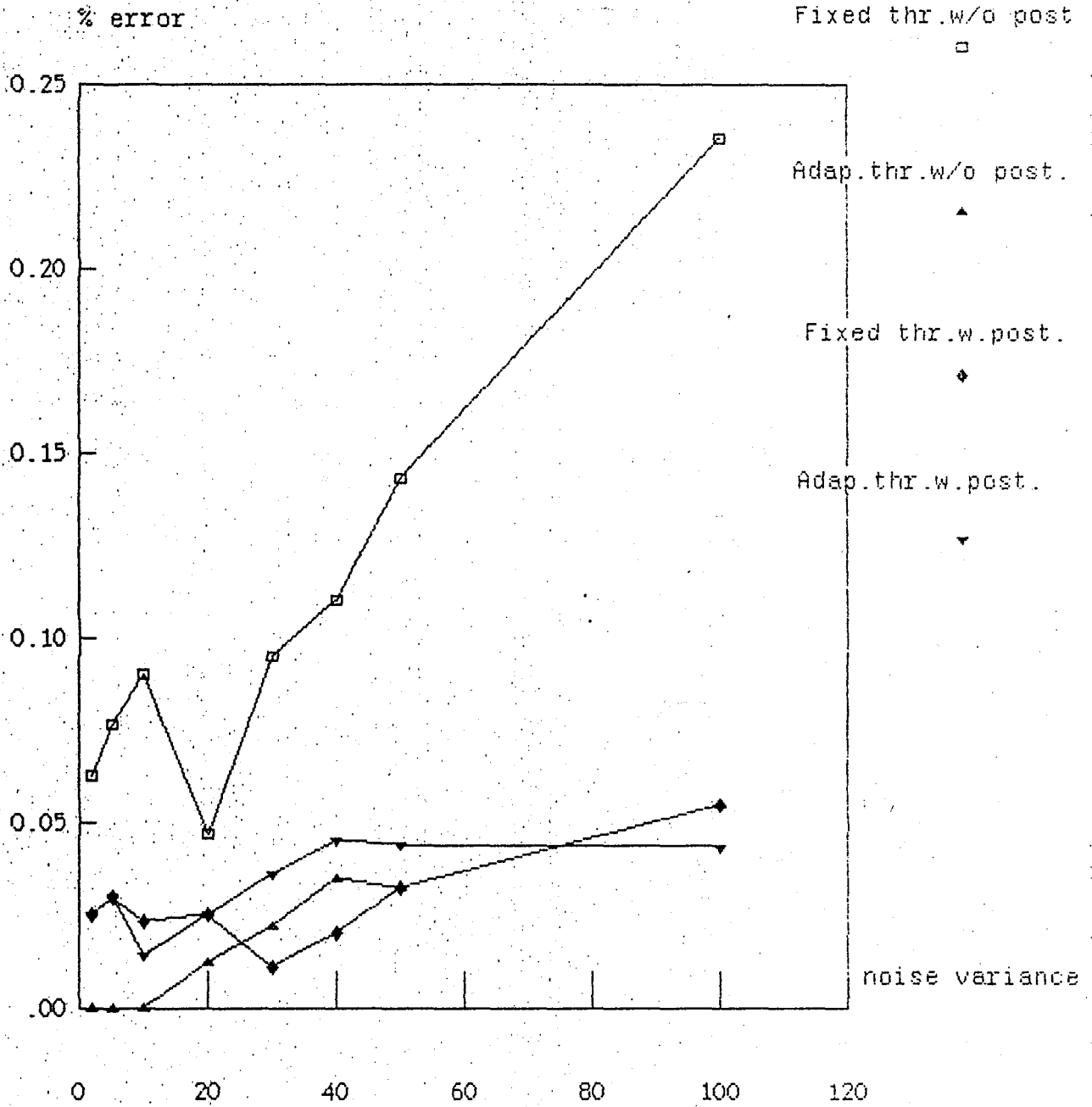


Fig.4.37 % error vs noise variance for fixed & adap.threshold, with & w/o postprocessing

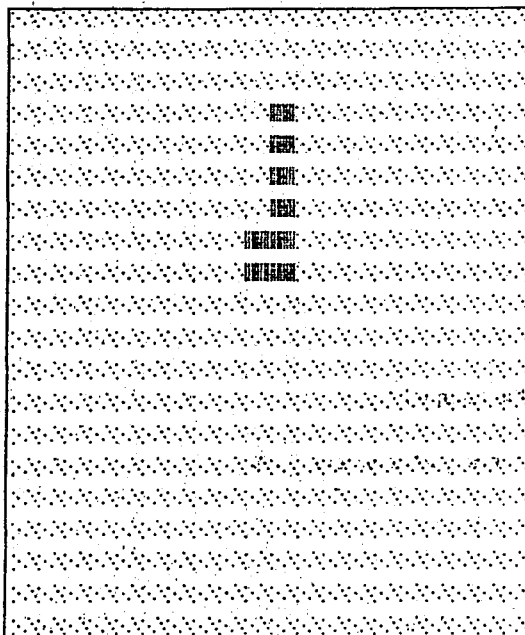
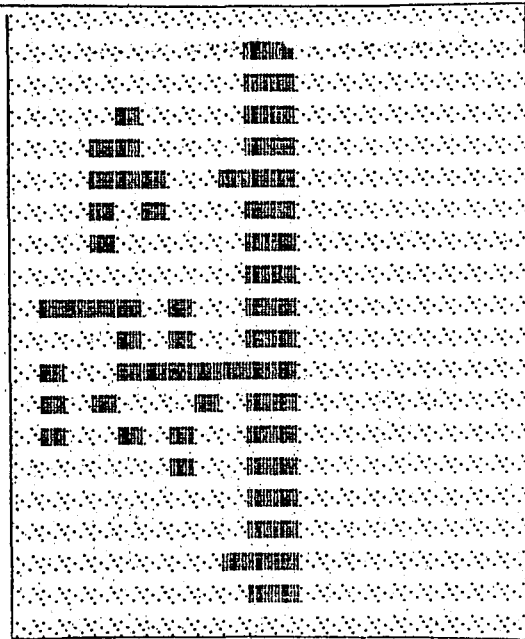
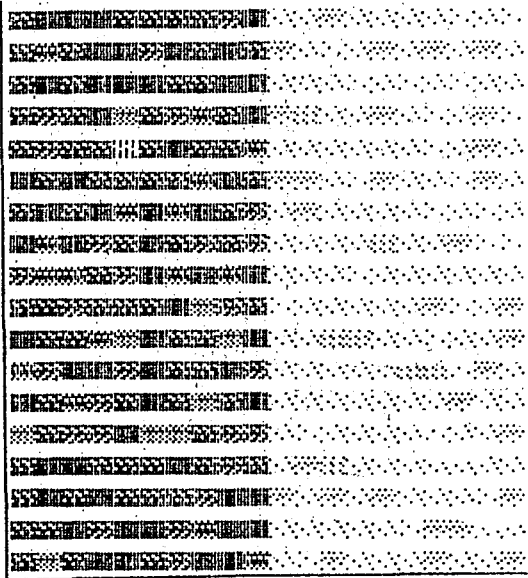


Fig 4.38 Simulation results of Kirsch masks: Original image, threshold image using fixed threshold and output of postprocessing operation respectively a) with noise, $\sigma=5, C=20$

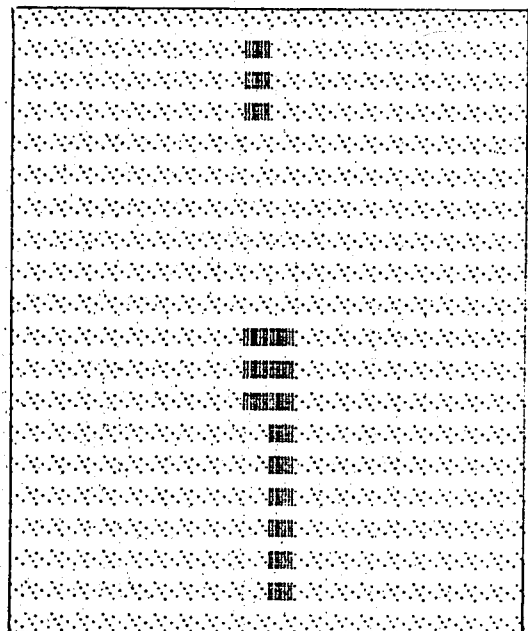
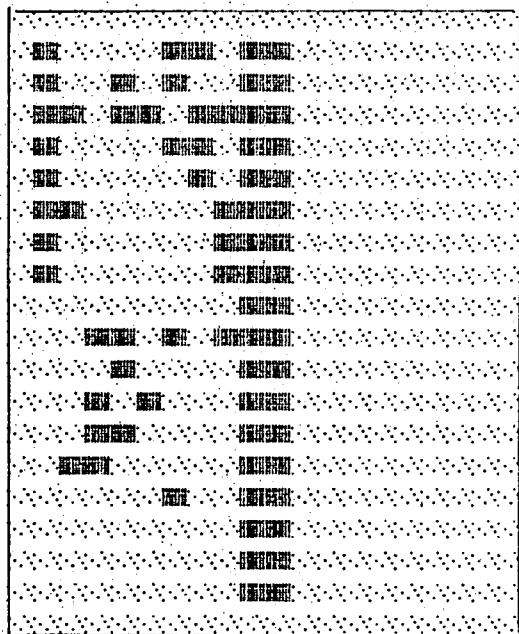
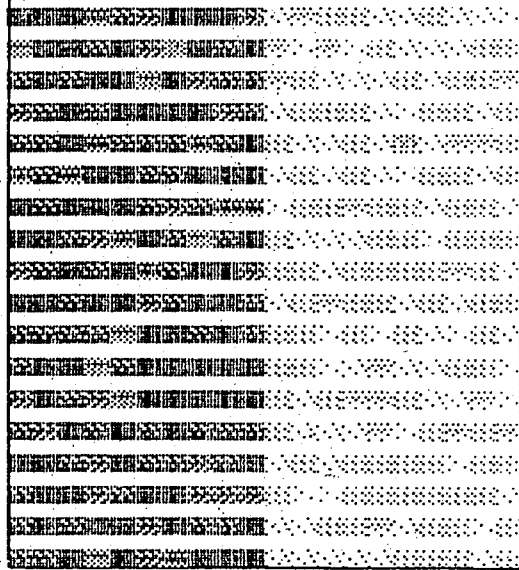


Fig 4.38 (continued) b) with noise, $\sigma=10, C=20$

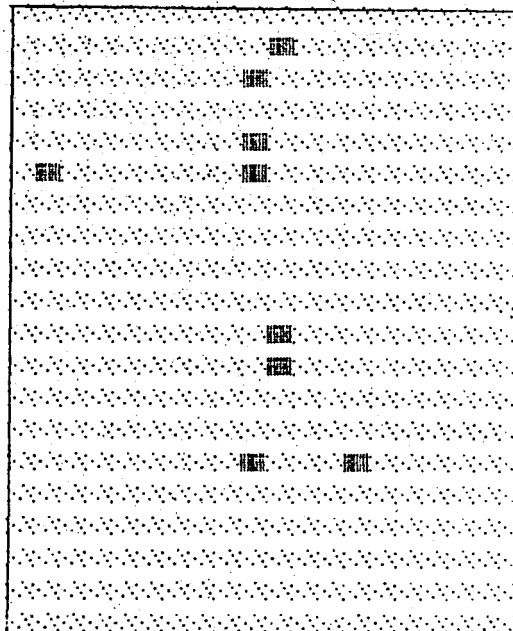
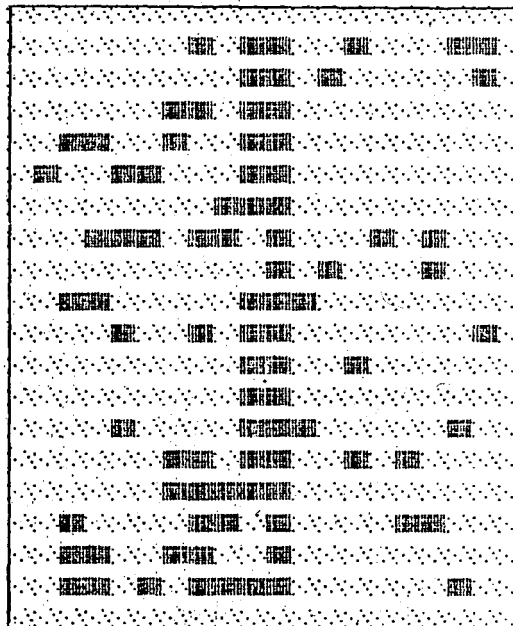
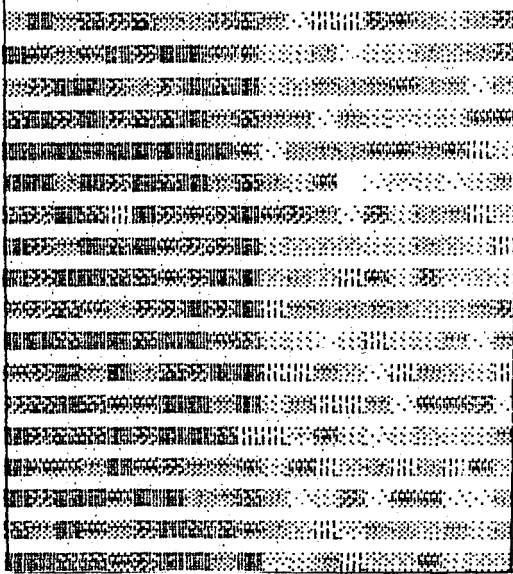


Fig 4.38 (continued) c) with noise, $\sigma=50, C=20$

The evaluation has been done by using only fixed threshold and for the values of $k=1, 8, 32, 8*4*2*1, 32*16*8*4*2*1$. No postprocessing algorithm has been used for this scheme. Fig (4.39) shows the simulation results for percentage error versus noise variance σ at a fixed contrast. For simulations $C=20$ is used as contrast value. As easily understood from the figure better results are obtained for larger values of k and the products of the differences give even better results. Even small noise variances are effective in the graph. The best result is obtained by the product of differences for $k=32$ (i.e., $32*16*8*4*2*1$).

Fig (4.40) shows the three simulation outputs with $\sigma = 5, 10, 50$. Noisy images and the outputs with fixed threshold for $k=8$ and $k=32*16*8*4*2*1$ can be seen from Fig (4.40). If Fig (4.32) is examined one more time, it is obvious that the result gets worse when $\sigma = 100$.

D - THE COMPARISON OF THE THREE PREPROCESSING SCHEMES

The comparison of the three schemes is done by only using a fixed threshold and without taking any postprocessing algorithm provided they are employed under the same conditions. Evaluation has been done for Sobel operator, Kirsch masks, Rosenfeld difference equations for $k=8$ and $k=32*16*8*4*2*1$ as in the case of the preceding criterion.

It is easily understood from Fig (4.41) that Rosenfeld's schemes do not give results as good as Sobel and Kirsch schemes do, i.e., it is better to use Sobel or Kirsch schemes in a noisy environment, but if the noise variance is large, the three schemes give almost the same response.

ROSENFELD DIFF. EQUATIONS

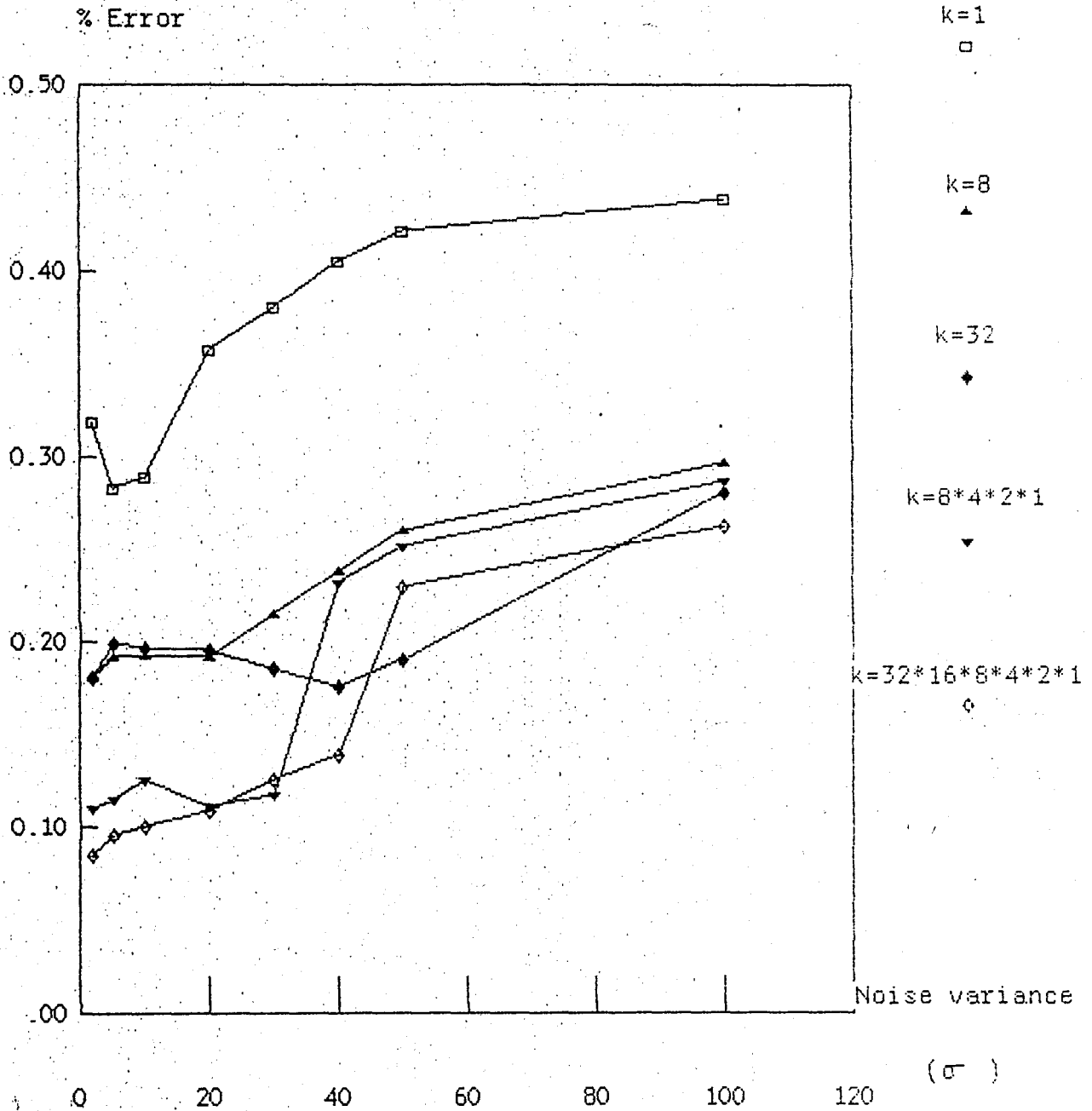


Fig.4.39 % error vs noise variance (σ) for fixed threshold, & w/o postprocessing

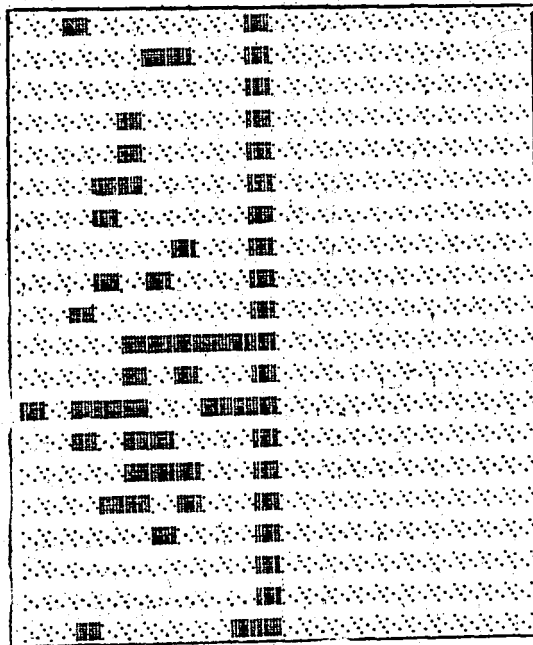
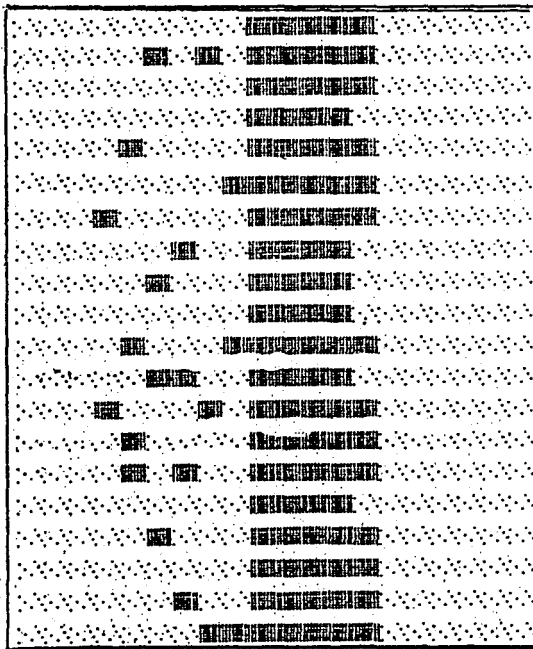
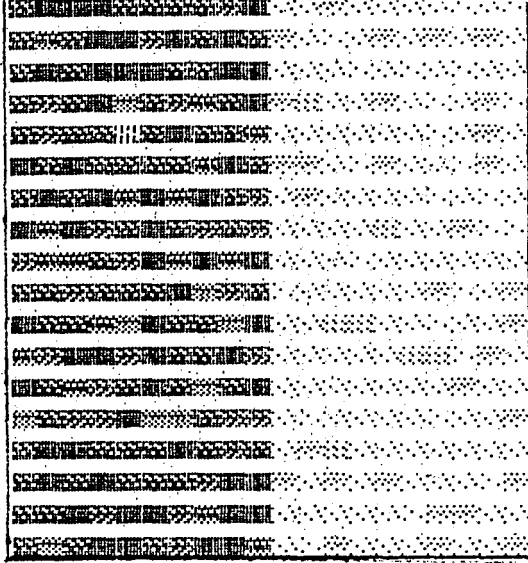


Fig 4.40 Simulation results of Rosenfelds algorithm:Original image, thresholded image with fixed threshold and output of postprocessing respectively a)with noise, $\sigma=5, C=20$

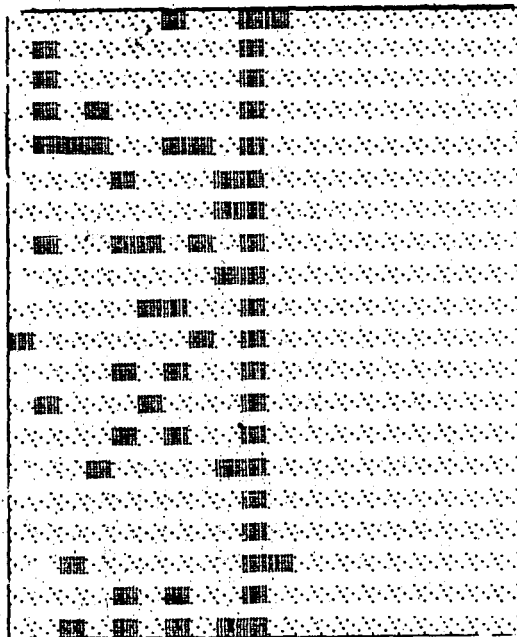
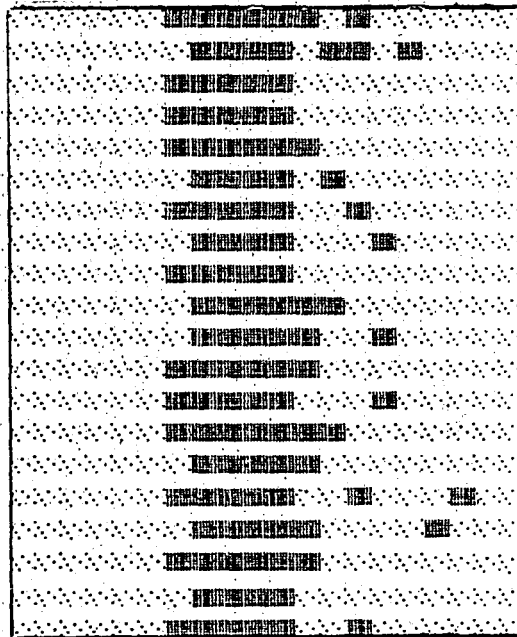
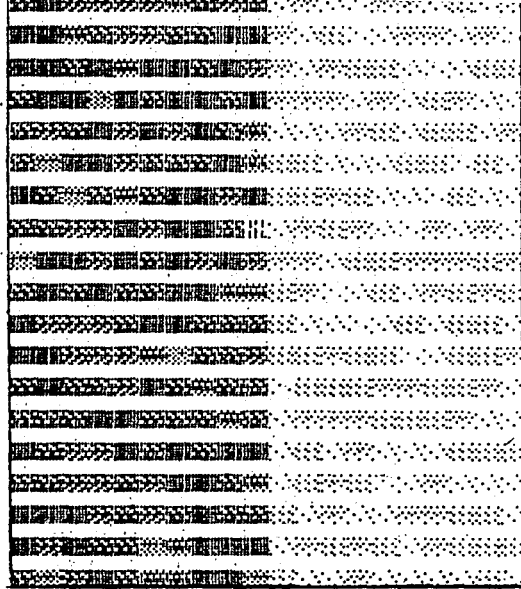


Fig. 4.40 (continued) b) with noise, $\sigma=10, C=20$

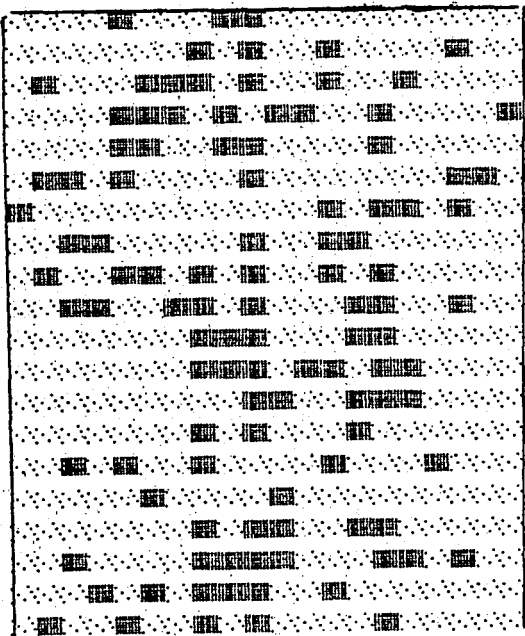
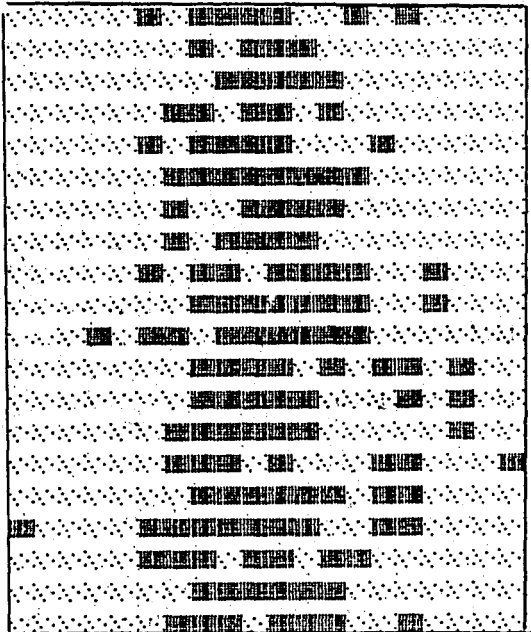
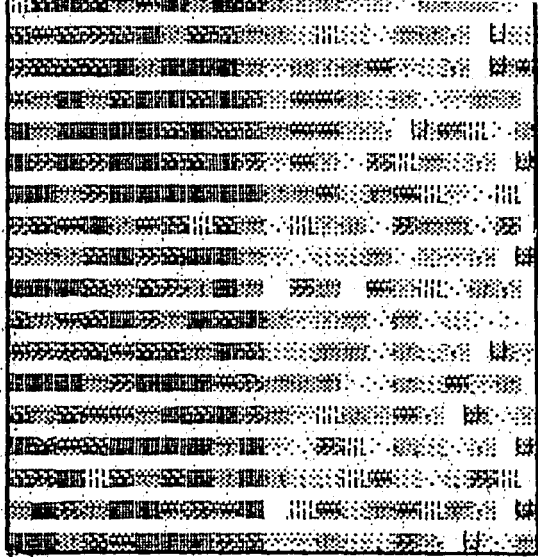


Fig 4.40 (continued) c) with noise, $\sigma=50, C=20$

Comparison between Sobel Operator, Kirsch Masks,

Rosenfeld diff. eqn 1 & 2 for $k=8$ and $k=32 \dots 1$ respectively

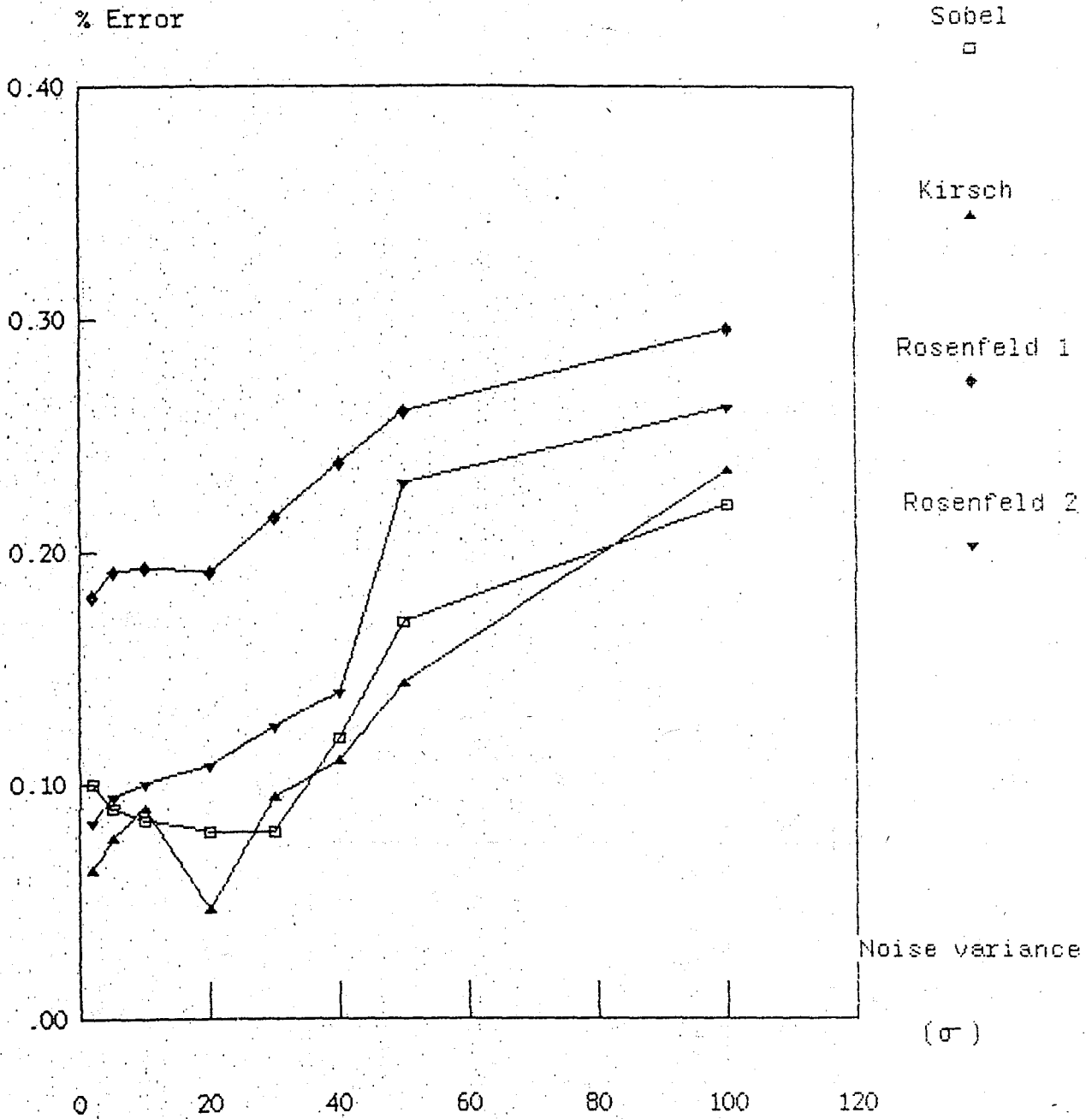


Fig. 4.41 % error vs noise variance $\sigma = 40$ for fixed threshold and without postprocessing

REMARKS

From the simulation results it can be said that the noise variance is a good criterion to see the behaviour of each scheme and to compare the schemes among themselves.

PARAMETER P

The parameter P has been calculated for each method in order to compare the behaviour of the mentioned edge detection schemes. P is calculated by Eqn. (4.31). Simulation runs have been taken for different contrast values once without noise and once with a noise of $\sigma=40$. For each contrast value the percentage error has been calculated for both, with and without the existence of noise. P is the ratio of these two values and is calculated from Eqn(4.12)

A - SOBEL OPERATOR

CONTRAST	FIXED				ADAPTIVE			
	€ ERR. IN PURE SIG		€ ERR. IN NOISY SIG		€ ERR. IN PURE SIG		€ ERR. IN NOISY SIG	
	W/O	WITH	W/O	WITH	W/O	WITH	W/O	WITH
2	24	4	99	40	109	4	144	35
5	32	1.5	83	37	111	0	155	38
10	37	1	77	31	117	1	147	27
20	38	1	47	9	107	0	129	16
30	44	0.4	42	1	108	0	128	0
50	41	0.7	25	1	112	0	106	1
80	47	0.7	26	1	112	1	117	0
100	44	0.5	25	1	110	0	120	1

Table 4.1 The value of P parameters for Sobel Operator with adaptive thr. & fix thr. ,with & w/o existence of noise ($\sigma = 40$)

SOBEL OPERATOR

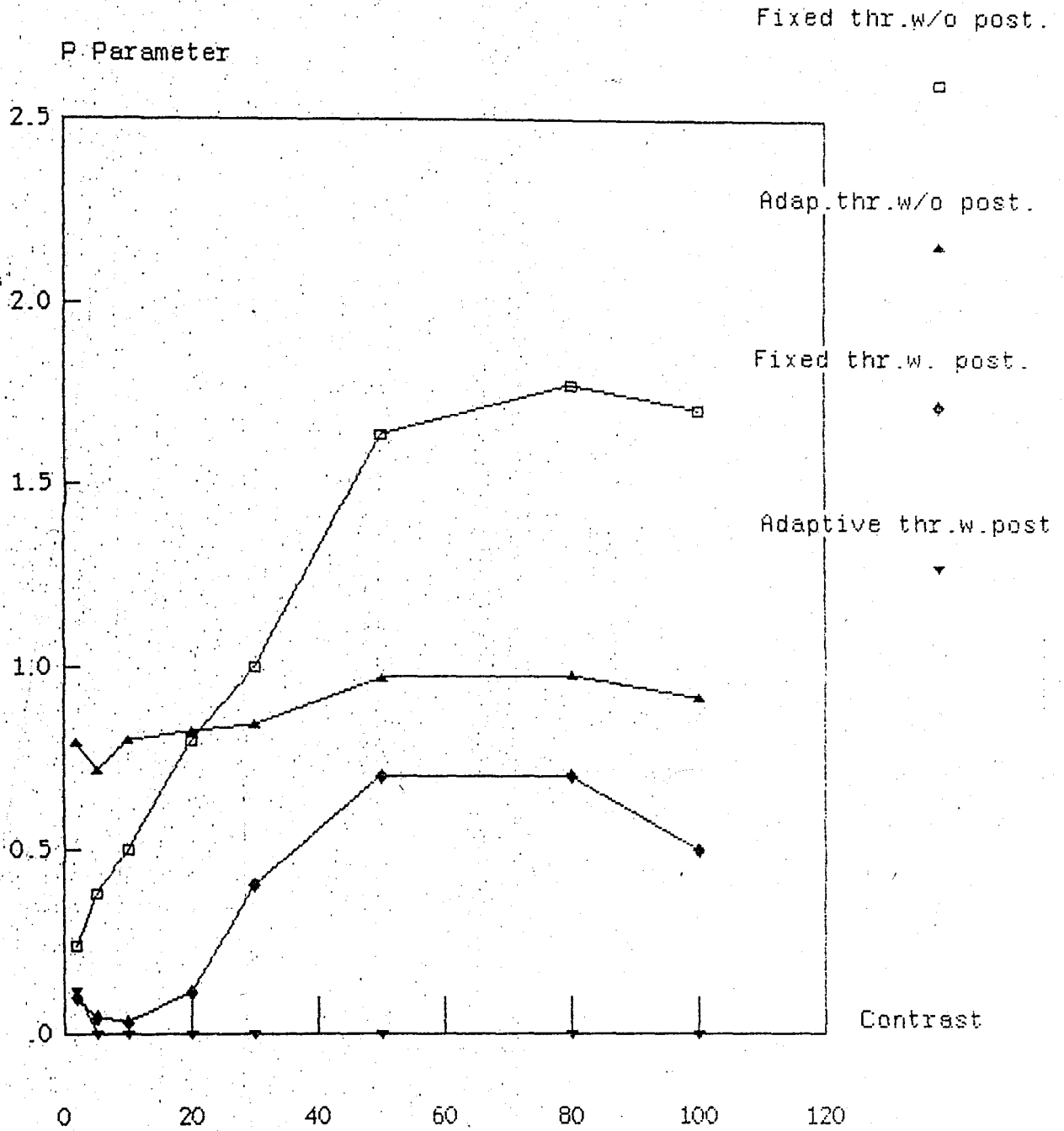


Fig.4.42 P parameter vs contrast for adaptive & .24

fixed threshold, with and w/o postprocessing,
261

KIRSH METHOD

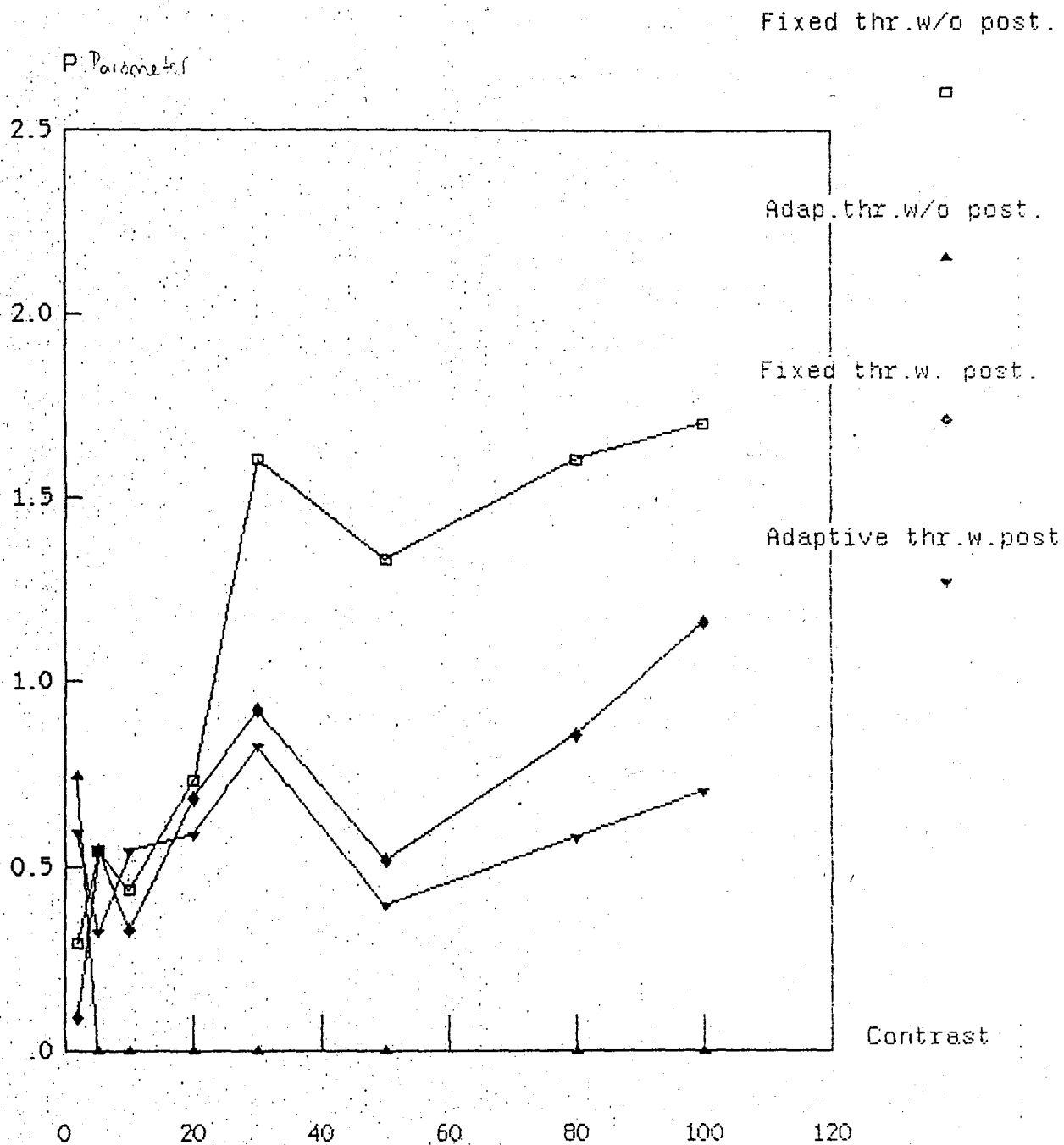


Fig.4.43 P parameter vs contrast for adaptive & fixed threshold, with and w/o postprocessing,

As in the case of the preceding criteria four different types of edge detection schemes are evaluated. The simulation results are shown in Fig (4.42) and Table (4.1). P is an increasing function of the contrast. This can be visualized from Fig (4.42). For the scheme with an adaptive threshold and the postprocessing algorithm, P is always zero, because the percentage error is zero for all contrast values in the pure image. That's why this graph can not give any idea about the variations due to noise. For the scheme with fixed threshold and without postprocessing algorithm, P increases rapidly, because the effect of noise decreases rapidly when the contrast value increases. For small contrasts, the noise is more effective than it is for higher contrasts. The graph of the scheme with fixed threshold and without postprocessing also shows that, the decrease of the noise effect, as well as the increase of percentage error for the pure image are the reason for such large values of P . The graph of the scheme with adaptive threshold and without postprocessing shows that the effect of noise does not change very much for increasing contrast values. The graph for the scheme with fixed threshold and with postprocessing shows that, for increasing contrast the effect of noise decreases but, because the increase of the percentage error for the pure image is less, P in this case does not reach such large values.

B - KIRSCH MASKS

Again, four different types of edge detection schemes are evaluated for Kirsch masks as in the case of the preceding criteria. The simulation results are shown in Fig (4.43) and table (4.2).

CONTRAST	FIXED				ADAPTIVE			
	ε ERR. IN PURE SIG		ε ERR. IN NOISY SIG		ε ERR. IN PURE SIG		ε ERR. IN NOISY SIG	
	W/O	WITH	W/O	WITH	W/O	WITH	W/O	WITH
2	31	3	108	22	11	16	21	18
5	42	11	80	18	0	8	20	18
10	32	6	73	18	0	10	19	18
20	43	8	43	7	0	9	14	17
30	37	8	23	13	0	8	9	10
50	28	5	17	11	0	9	4	15
80	37	9	18	10	0	9	6	12
100	45	10	14	7	0	4	3	11

Table 4.2 The value of P parameters for Kirsch Mask with adaptive thr. & fix thr, with & w/o existence of noise ($\sigma = 40$)

All schemes show approximately the same characteristic. P is an increasing function of contrast for all schemes but the increasing rate changes from one scheme to another. As in the case of the Sobel operator we can not say anything about the scheme with adaptive threshold and connectivity test because of the same reason. For adaptive thresholding with connectivity test the sensitivity to noise is much more than for any other scheme. As the contrast increases P increases slightly. The graph of the scheme with fixed threshold and without connectivity test increases sharply. This sharp increase can be interpreted in the following way. Since for increasing contrast values the number of errors in the pure signal is increasing whereas the number of errors in the noisy signal is decreasing, their ratio, the P parameter, will increase very rapidly. That's why the best result is obtained by the scheme with fixed threshold and with connectivity test.

C - ROSENFELD DIFFERENCE EQUATIONS.

Five different schemes are evaluated for Rosenfeld difference equations as in the case of the preceding criteria. The results of the simulations are given in Fig (4.44) and Table (4.3). All schemes approximately show the same characteristics. For all schemes P is an increasing function of contrast. The rates of increase are also very similar to each other. As it can be seen from Fig (4.44) the noise effects the schemes much more for small contrasts than for greater contrasts.

CONTRAST	k=1		k=8		k=32		k=8.4.2.1		k=32.15...2.1	
	ε ERR PURE SIGNAL	ε ERR NOISY SIGNAL	ε ERR PURE SIGNAL	ε ERR NOISY SIGNAL	ε ERR PURE SIGNAL	ε ERR NOISY SIGNAL	ε ERR PURE SIGNAL	ε ERR NOISY SIGNAL	ε ERR PURE SIGNAL	ε ERR NOISY SIGNAL
	2	121	190	104	139	39	140	7	138	8
5	103	196	87	134	50	124	43	143	8	116
10	127	171	75	121	39	106	75	122	12	105
20	163	159	95	95	81	92	71	70	53	56
30	122	150	76	88	43	73	79	84	30	64
50	120	133	74	72	33	47	69	67	30	32
80	126	105	78	73	49	43	71	69	41	41
100	122	102	81	70	44	37	66	67	35	32

Table.4.3 The value of P parameters for Rosenfeld's Difference Equations with adap.thr.& fixed thr., with & w/o existence of noise ($\sigma = 40$)

D - COMPARISON OF THE THREE PREPROCESSING SCHEMES

As for the preceding criteria, comparison is done only by using fixed thresholding without postprocessing algorithms. Evaluation has been done for Sobel operator, Kirsch masks, Rosenfeld difference equations with k=8 and

ROSENFELD DIFF. EQUATIONS

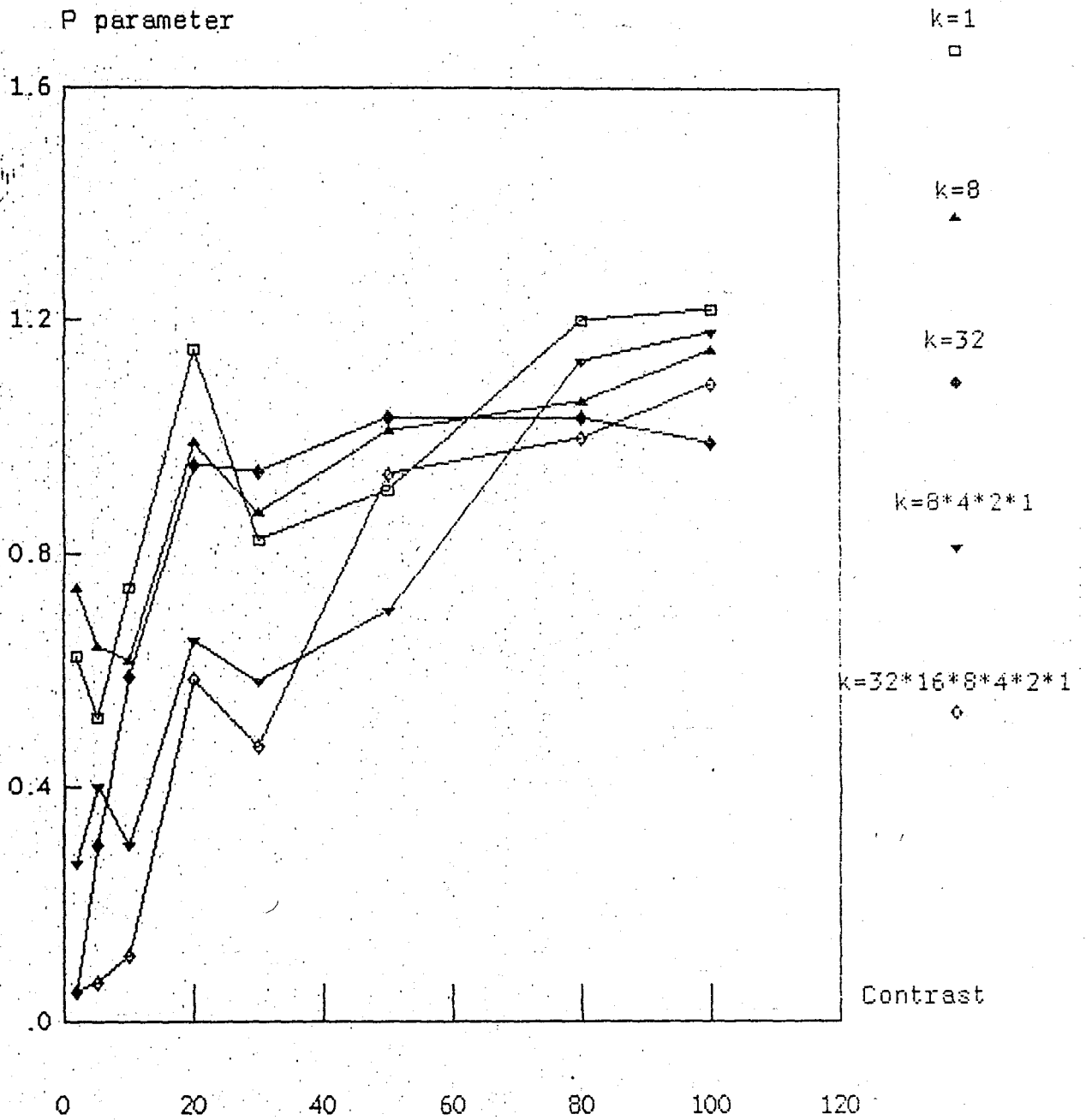


Fig.4.44 P parameter vs contrast for fixed threshold,
and w/o postprocessing

Comparison between Sobel Operator, Kirsch Masks,

Rosenfeld diff. eqn 1 & 2 for $k=8$ and $k=32 \dots 1$ respectively

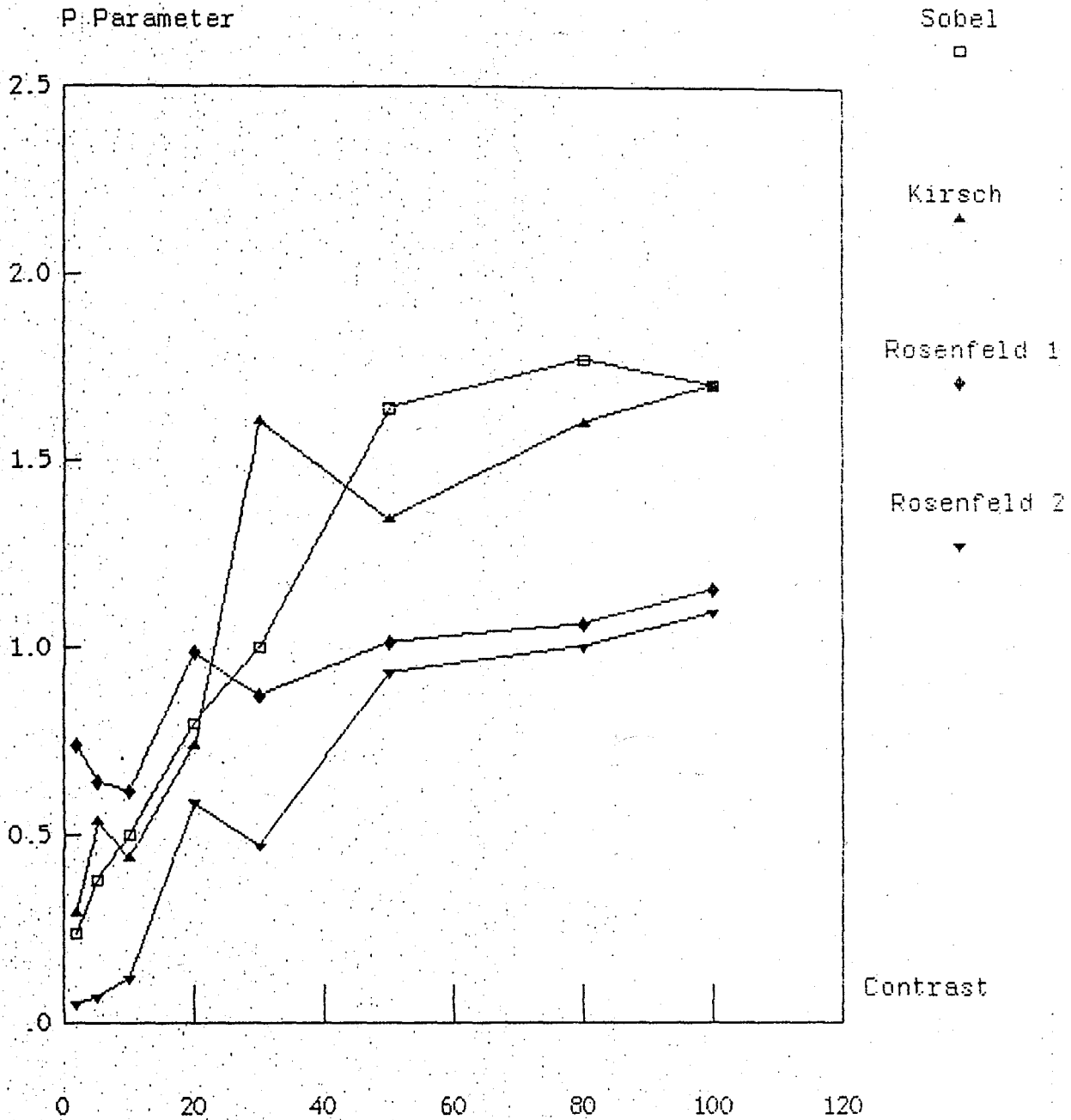


Fig.4.45 P parameter vs contrast for fixed threshold and without postprocessing

$k=32*16*8*4*2*1$. Fig (4.45) shows the results of the simulations.

For small contrasts better results can be obtained by using Rosenfeld products of differences for large k . Since a nonincreasing P function means that the effect of noise does not diminish for increasing contrast values, Rosenfeld's method is not convenient for great contrasts. Since a sharp increase of the parameter P shows not only the diminishing effect of noise but also that the percentage error in the pure signal is increasing, this much of increase is not desired. This makes the Sobel operator more desirable than the Kirsch masks because of their somewhat smoother increase.

MEAN SQUARE DISTANCE OF ERRORS

This parameter is used to decide whether the 1's out of the edge region are near the edge or not, i.e., the edge is blurred or not.

A - SOBEL OPERATOR

The simulation results of four different schemes of the Sobel operator without and with the existence of noise ($\sigma=40$) are shown in Fig (4.46) and (4.47) respectively.

B - KIRSCH MASKS

The simulation results of four different schemes of Kirsch masks w/o and with noise ($\sigma=40$) are shown in Fig (4.48) and (4.49), respectively.

SOBEL OPERATOR

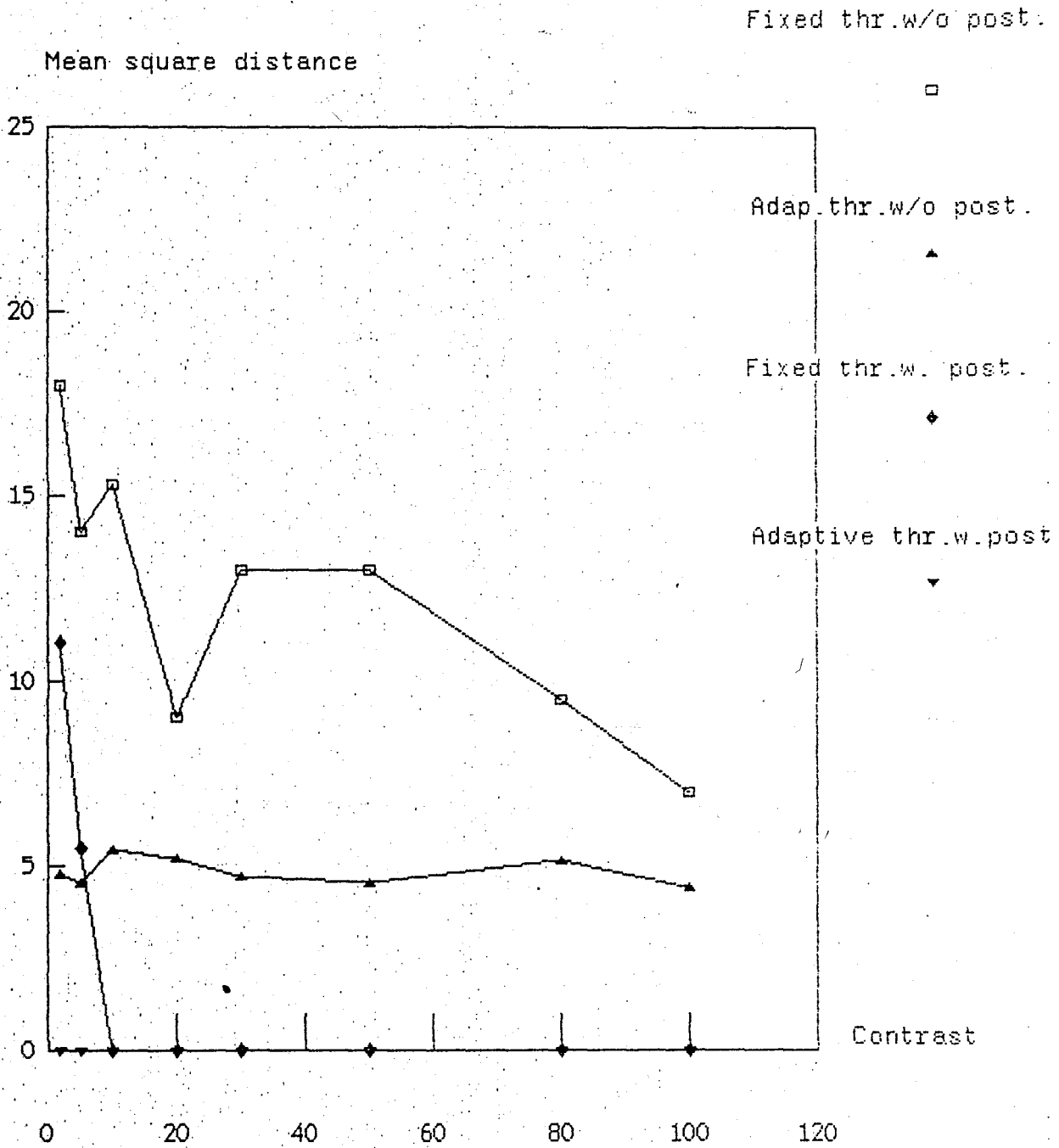


Fig.4.46 Mean square distance vs contrast for adaptive & fixed thresh with and w/o postprocessing, without the existence of noise

SGBEL OPERATOR

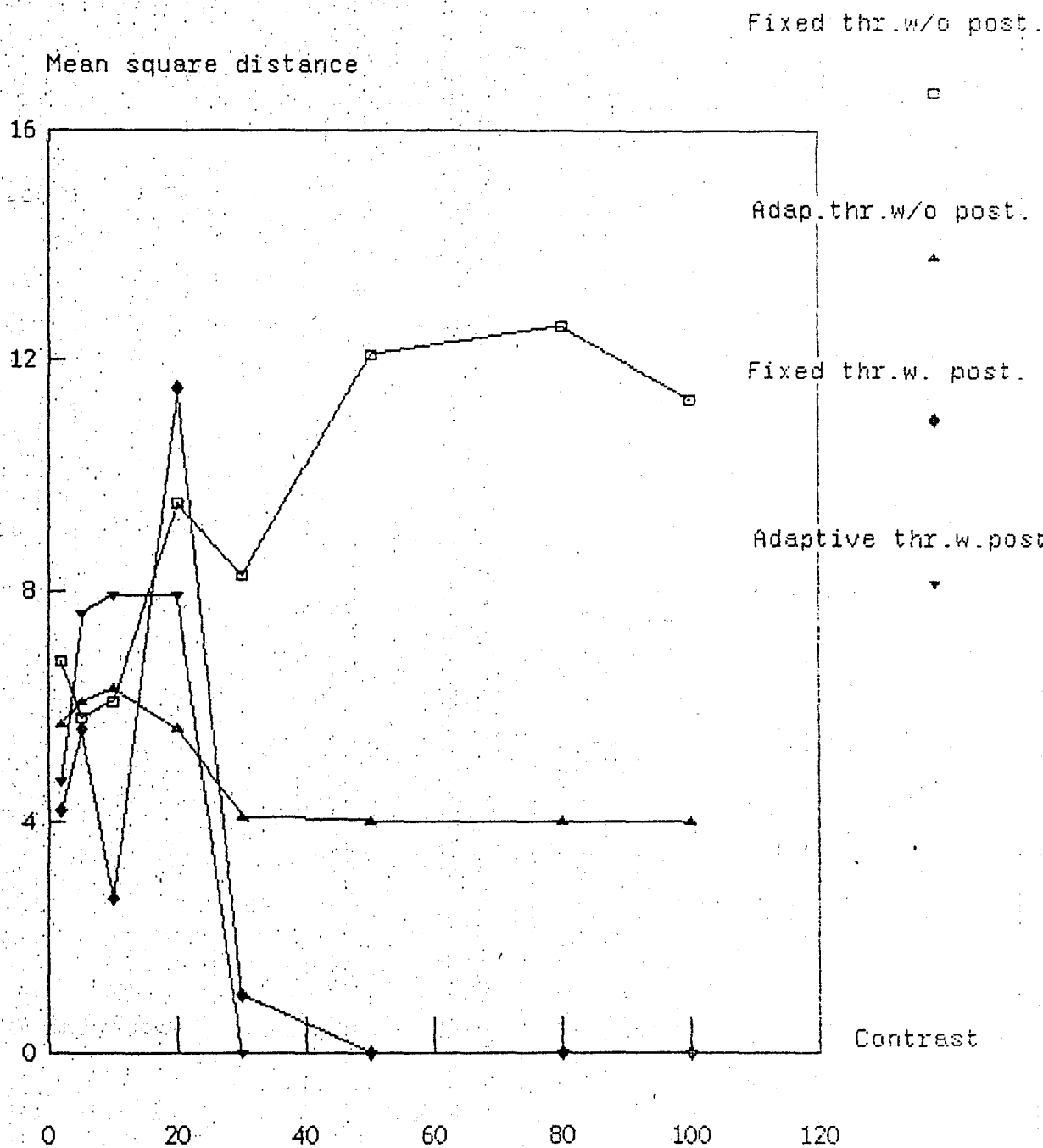


Fig.4.47 Mean square distance vs contrast for adaptive & fixed thres with and w/o postprocessing, with the existence of noise ($\sigma = 40$)

KIRSH METHOD

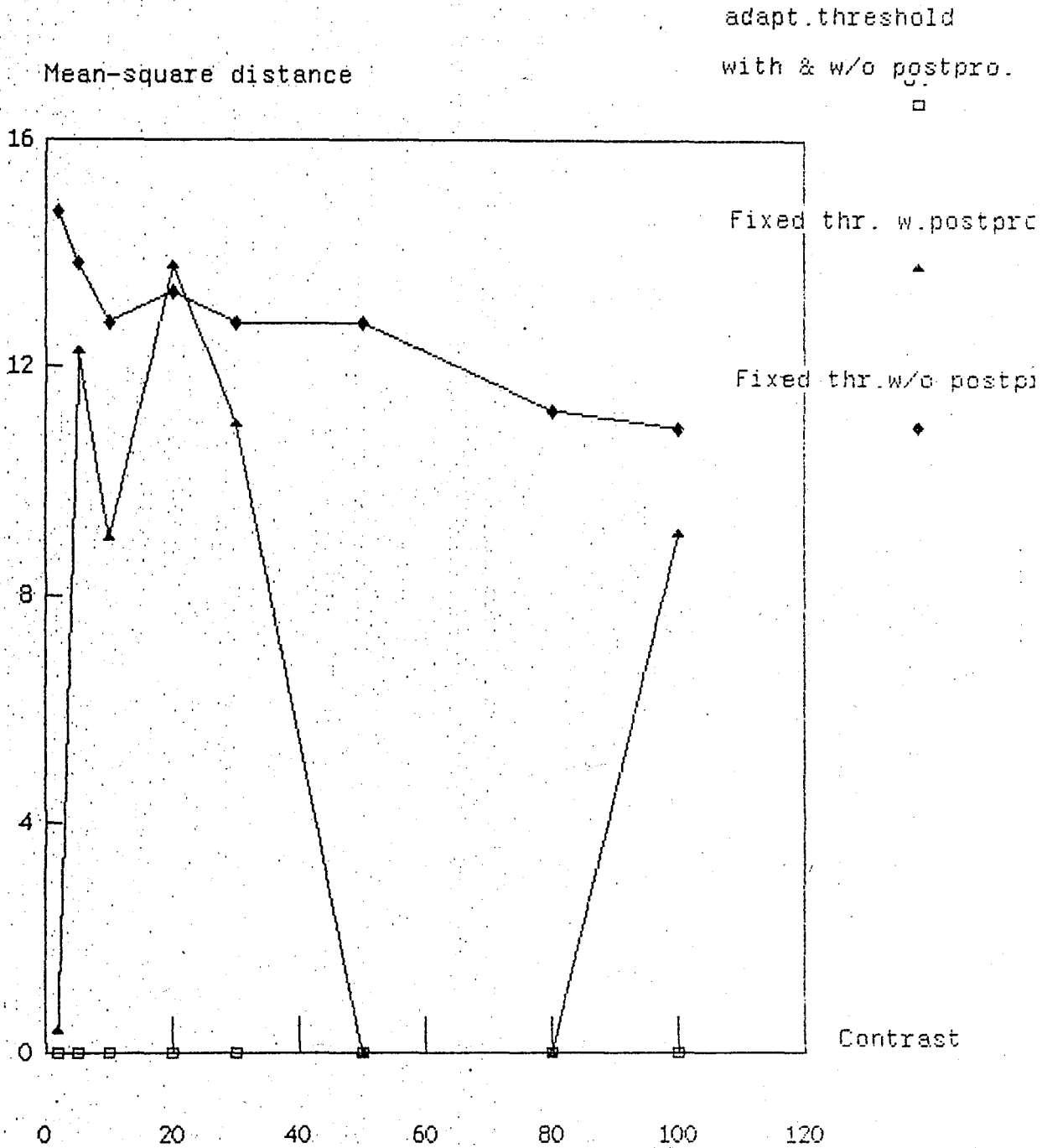


Fig.4.48: Mean square distance vs contrast for fixed & adap. threshold with and w/o postprocessing, w/o the existence of noise

KIRSH METHOD

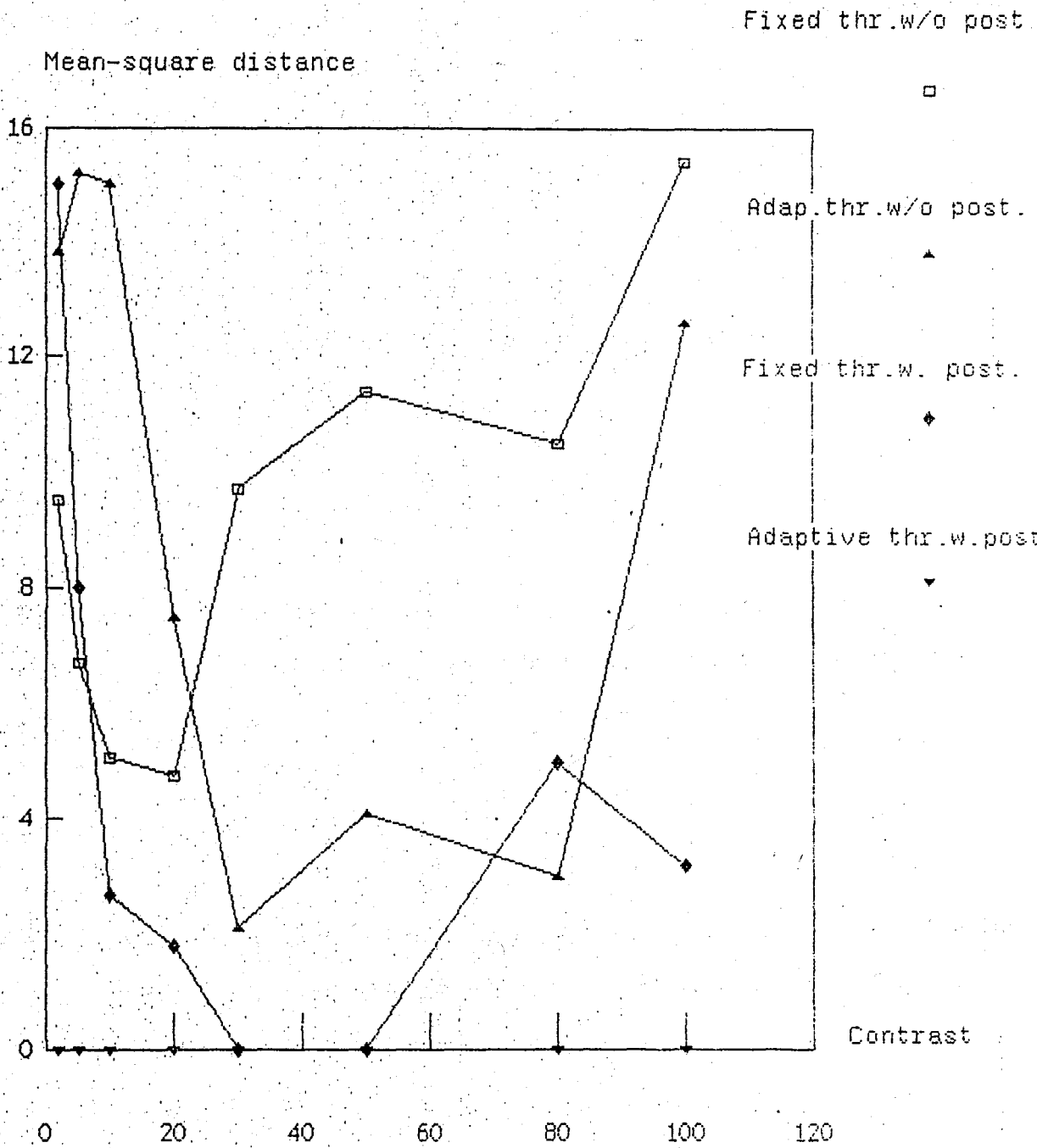


Fig. 4.49 Mean square distance vs contrast for fixed & adap. threshold, with and w/o postprocessing, with the existence of noise($\sigma = 40$)

C - ROSENFELD DIFFERENCE EQUATIONS

The simulation results of the five different schemes of Rosenfeld edge detection method without and with the existence of noise ($\sigma = 40$) are shown in Fig (4.50) and (4.51) respectively.

REMARKS:

It is obvious from Fig (4.46) through (4.51) that, this parameter can not be used for evaluation, because it does not show any consistency and give a reasonable result.

MISSING EDGE POINTS

This parameter shows the uncovered rows in the edge region and gives an idea about the schemes whether they can catch the points in the edge region sufficiently or not.

A - SOBEL OPERATOR

Fig (4.52) shows the ratio of missing edge points per total number of points in an image as a function of contrast for four different types of Sobel operator schemes without the existence of noise.

The figure shows that there are no missing edge points when the postprocessing algorithm is not used. But, if it is used then the scheme with adaptive thresholding gives a better result with respect to this

ROSENFELD DIFF. EQUATIONS

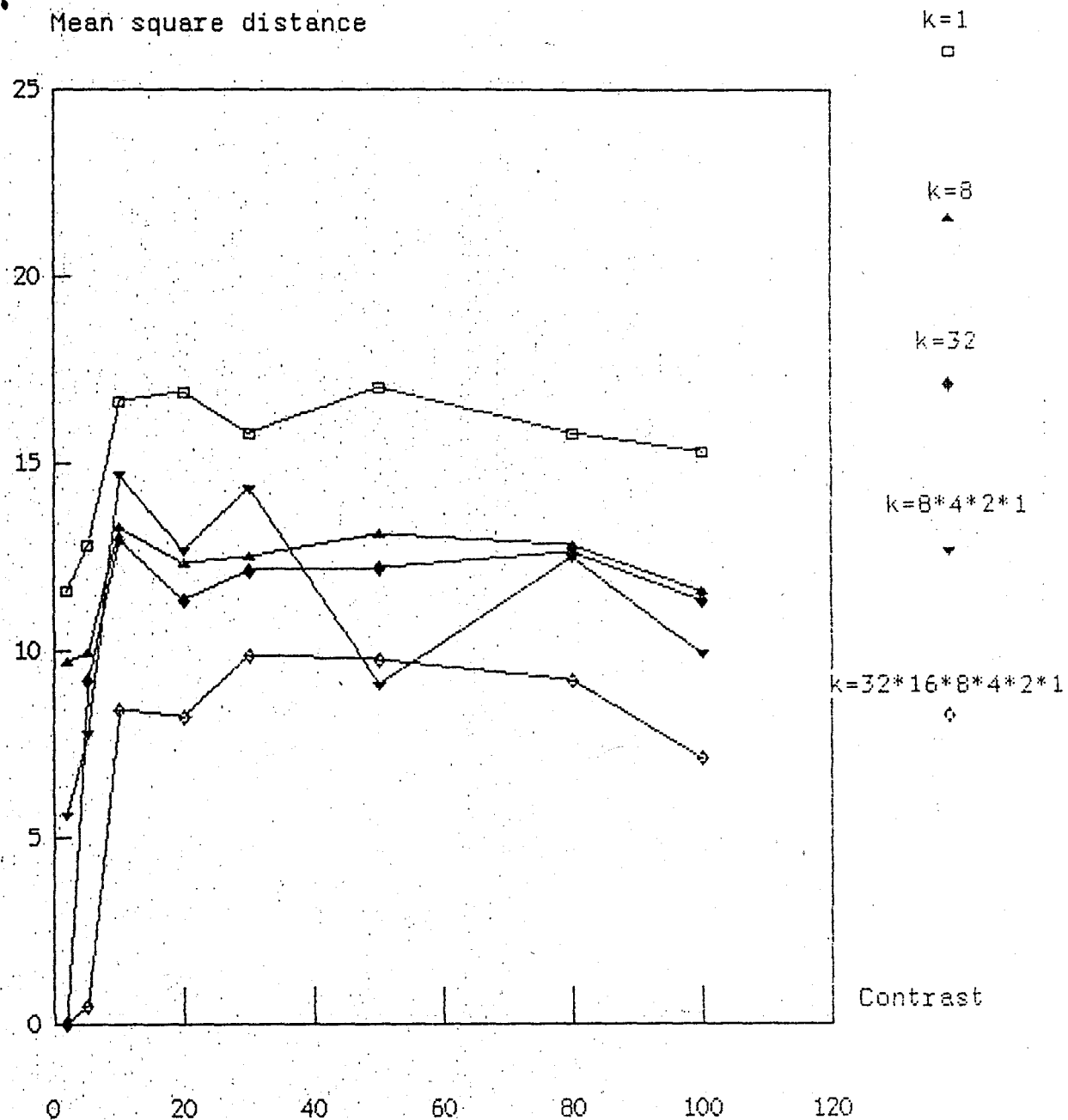


Fig 4.50 Mean square distance vs. contrast for fixed threshold and without postprocessing algorithm and w/o noise.

ROSENFELD DIFF. EQUATIONS

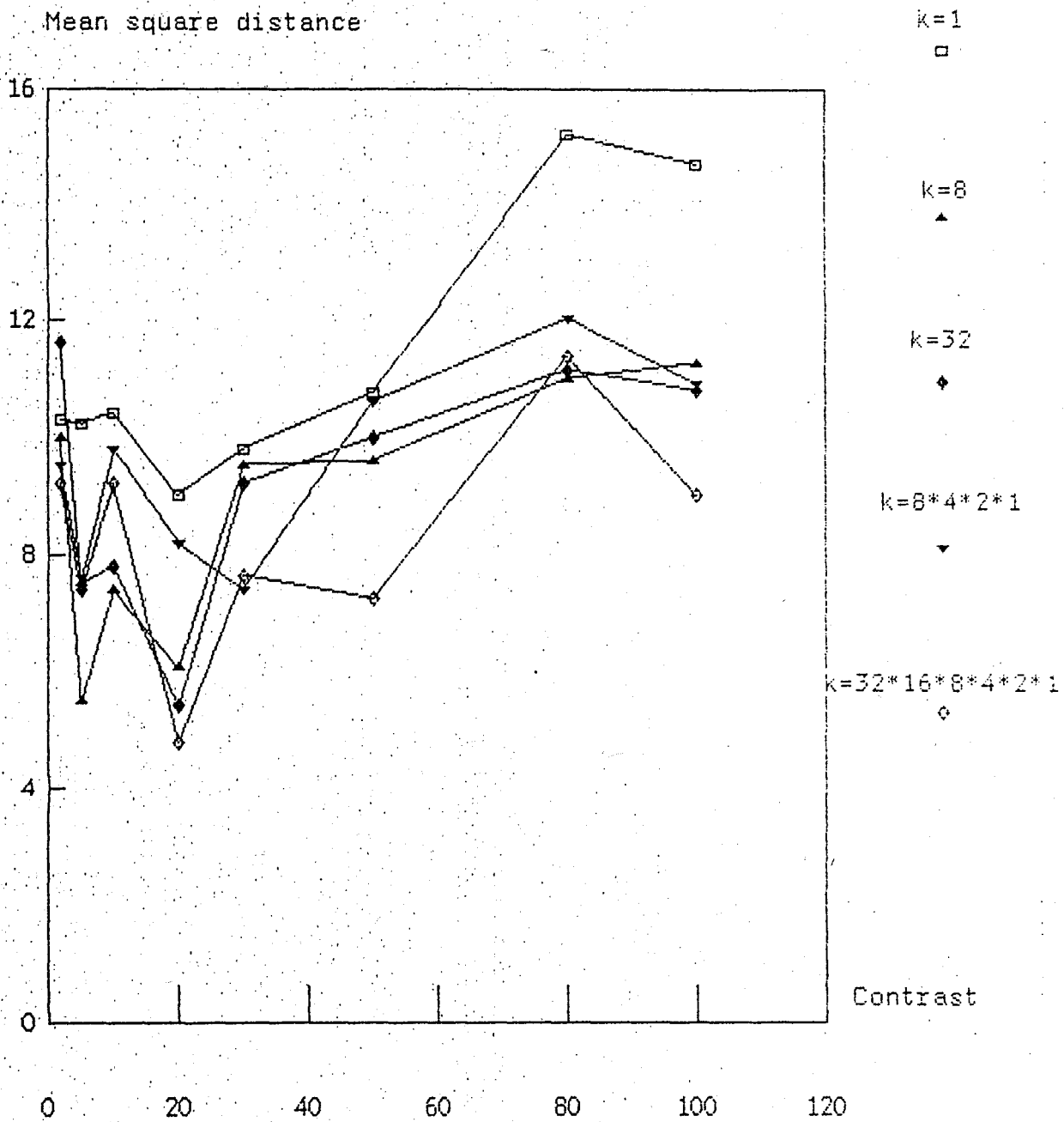


Fig 4.51 Mean square distance vs. contrast for fixed threshold and without postprocessing algorithm and with noise, $\sigma=40$,

SOBEL OPERATOR

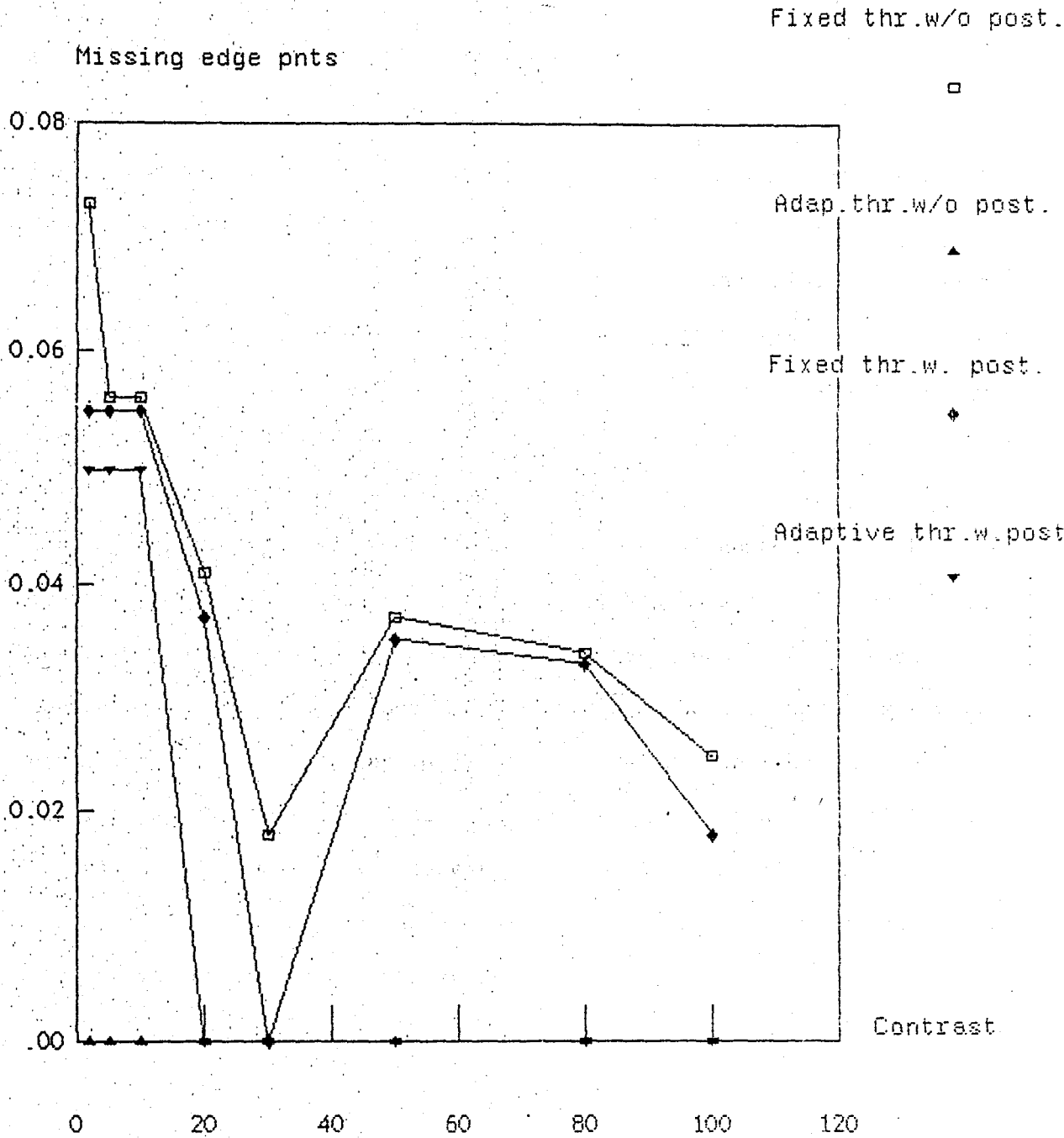


Fig.4.52 Missing edge pnt vs contrast for adaptive & fixed threshold with and w/o postprocessing, without the existence of noise

criterion. Fig (4.53) shows the results of the same criterion with the existence of noise. The comparison of Fig (4.52) and (4.53) indicates that the number of missing edge points increases when noise is added to the picture. Since for large contrasts the noise loses its effect the graphs in the above mentioned figures have approximately the same shape.

B - KIRSCH MASKS

Fig (4.54) and (4.55) show the ratio of the missing edge points per total number of points in the picture as a function of contrast for four different types of schemes of Kirsch masks without and with the existence of noise, respectively. Although the connectivity test sharpens the edges it causes some edge points to be missed. That means, that for both with and without the existence of noise, the schemes without connectivity test give better results with respect to this criterion. When there is noise, especially for low contrasts, edges are totally missed. For small contrasts the best result is obtained by using the scheme with fixed threshold and with connectivity test. In the upper part of the contrast domain the schemes without connectivity test catch almost all edge points whereas the schemes with connectivity test miss many points of the edge region. The worst result is obtained by the scheme with adaptive threshold and with connectivity test because adaptive threshold (convolving with an LPF, M_0) itself, gives the sharpest edge. If the connectivity test is applied to this output, it is obvious that some edge points will be missed.

SOBEL OPERATOR

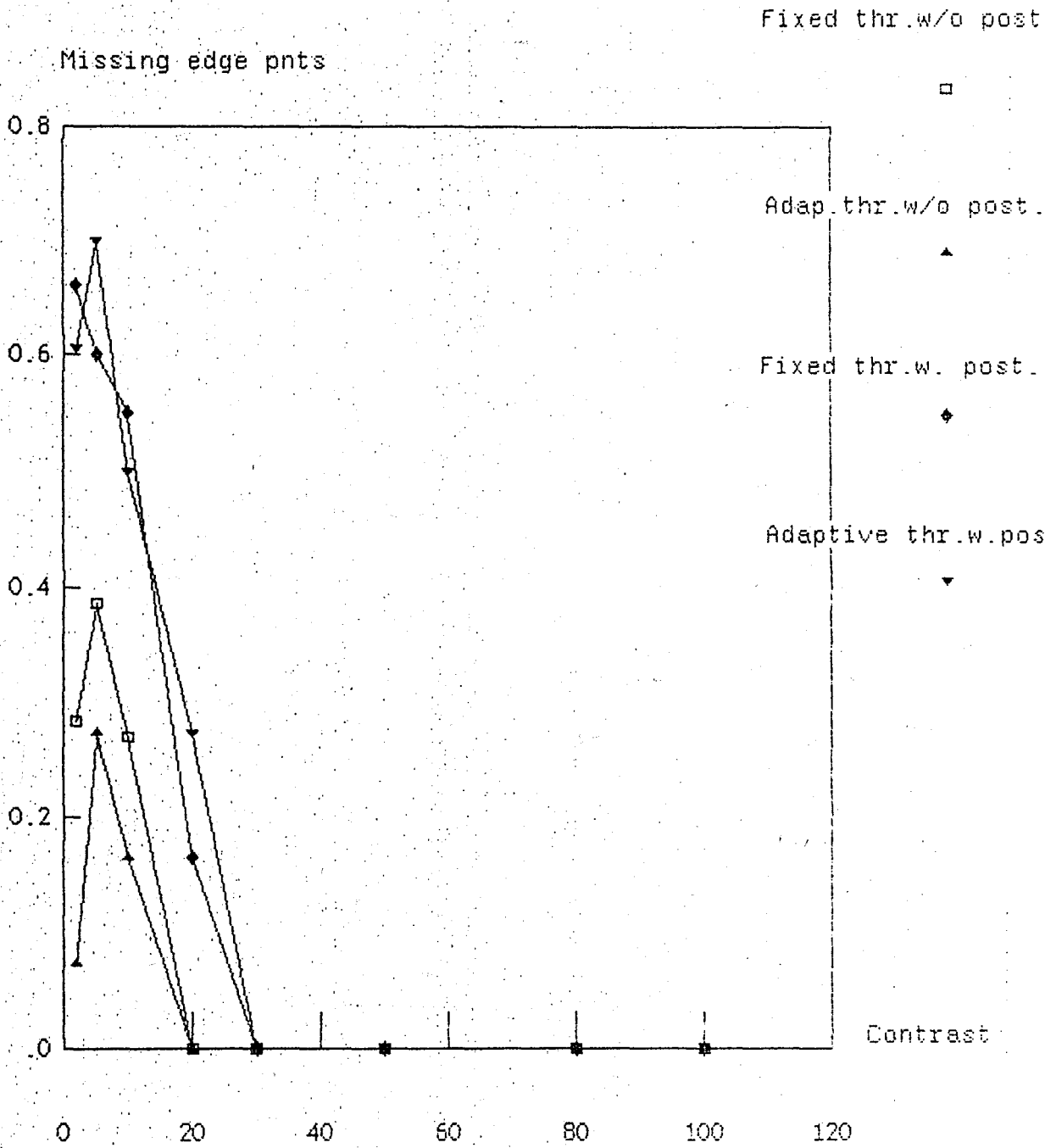


Fig.4.53 Missing edge pnt vs contrast for adaptive & fixed threshold with and w/o postprocessing, with the existence of noise ($\sigma=40$)

KIRCSH MASKS

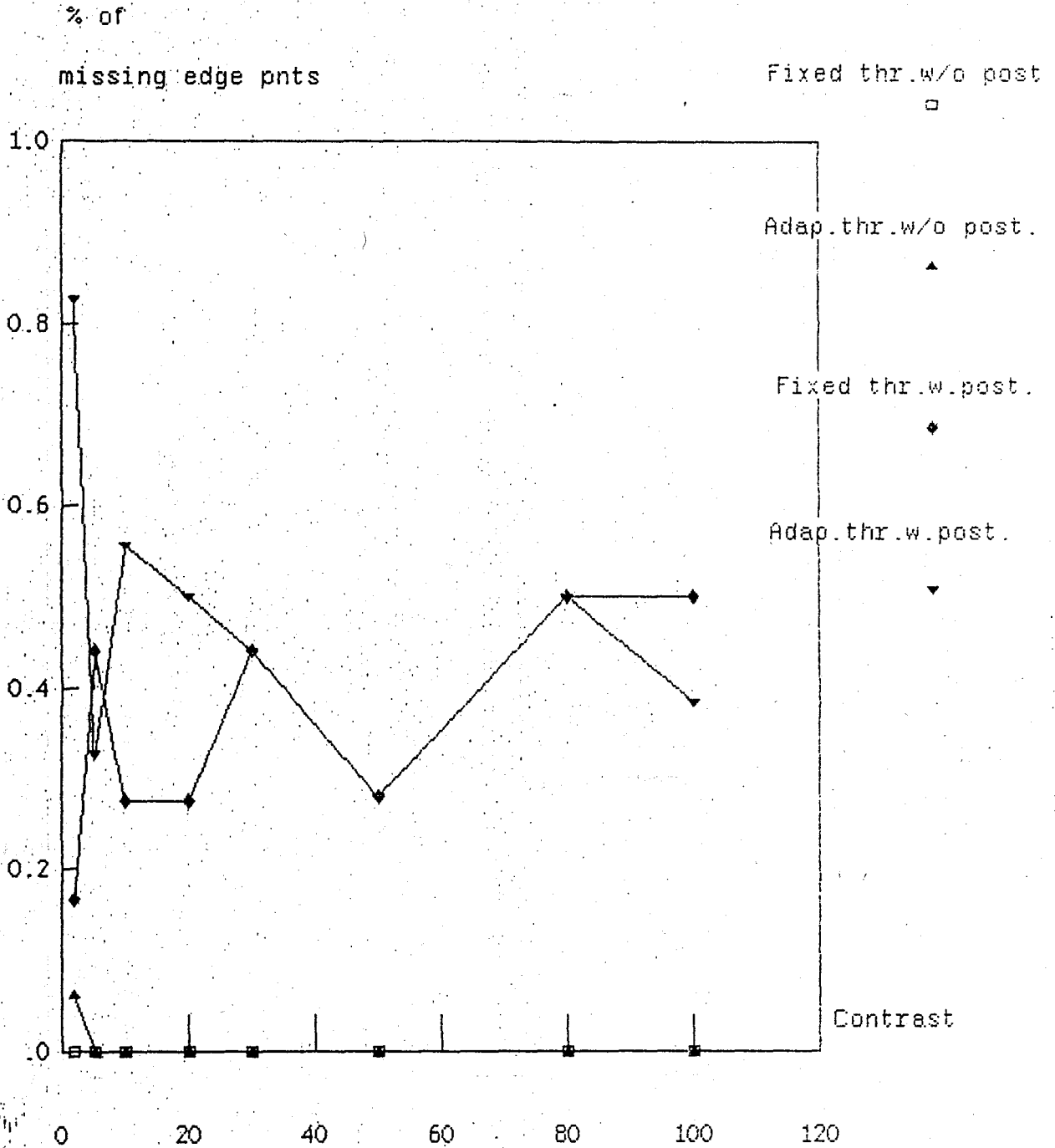


Fig.4.54 % of missing edge pnts vs contrast for fixed & adap.thr. with & w/o postprocessing, without existence of noise

KIRCSH MASKS

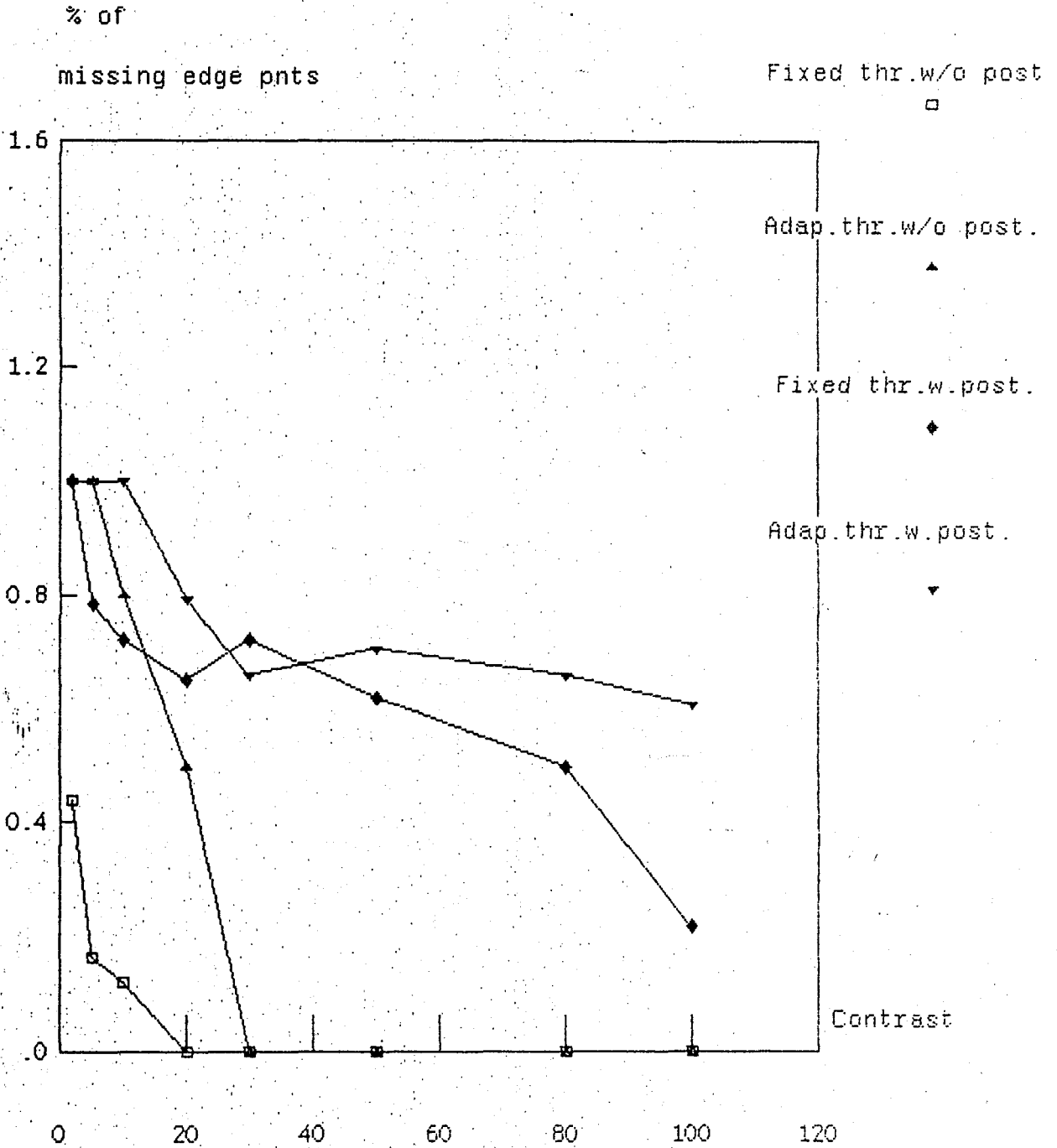


Fig.4.55 % of missing edge pnts vs contrast for fixed & adap.thr. with & w/o postprocessing, with the existance of noise ($\sigma = 40$)

C - ROSENFELD DIFFERENCE EQUATIONS

Fig (4.56) and (4.57) shows the outputs of the same criterion for five different schemes using Rosenfeld difference equations without and with the existence of noise, respectively. As shown in Fig (4.56) and (4.57) this result gives good results for this criterion. All five schemes catch all the edge points when there is no noise and for small contrasts some of the edge points are missed.

D - THE COMPARISON OF THE THREE PREPROCESSING SCHEMES

Fig (4.58) and (4.59) show the graphs of the percentage missing edge points as a function of contrast for the discussed preprocessing schemes. Fixed thresholding without postprocessing is used. If there is no noise then there are no missing edges points for any of the schemes. As it can be expected, for contrasts up to $C=30$ the noise is more effective than it is for greater contrast values. Therefore, edge points are missing in the lower contrast domain whereas there all edge points are covered in the upper part. For greater noise values it can be expected that the effectiveness threshold of $C=30$ will also rise.

REMARKS

1) This criterion can not be used alone to evaluate the performance of the edge detection schemes, a scheme may catch a lot of nonedge points besides of all edge points. Although all edge points are covered what sounds good, such a result is not desired. Percentage error versus contrast or percentag

ROSENFELD DIFF. EQUATIONS

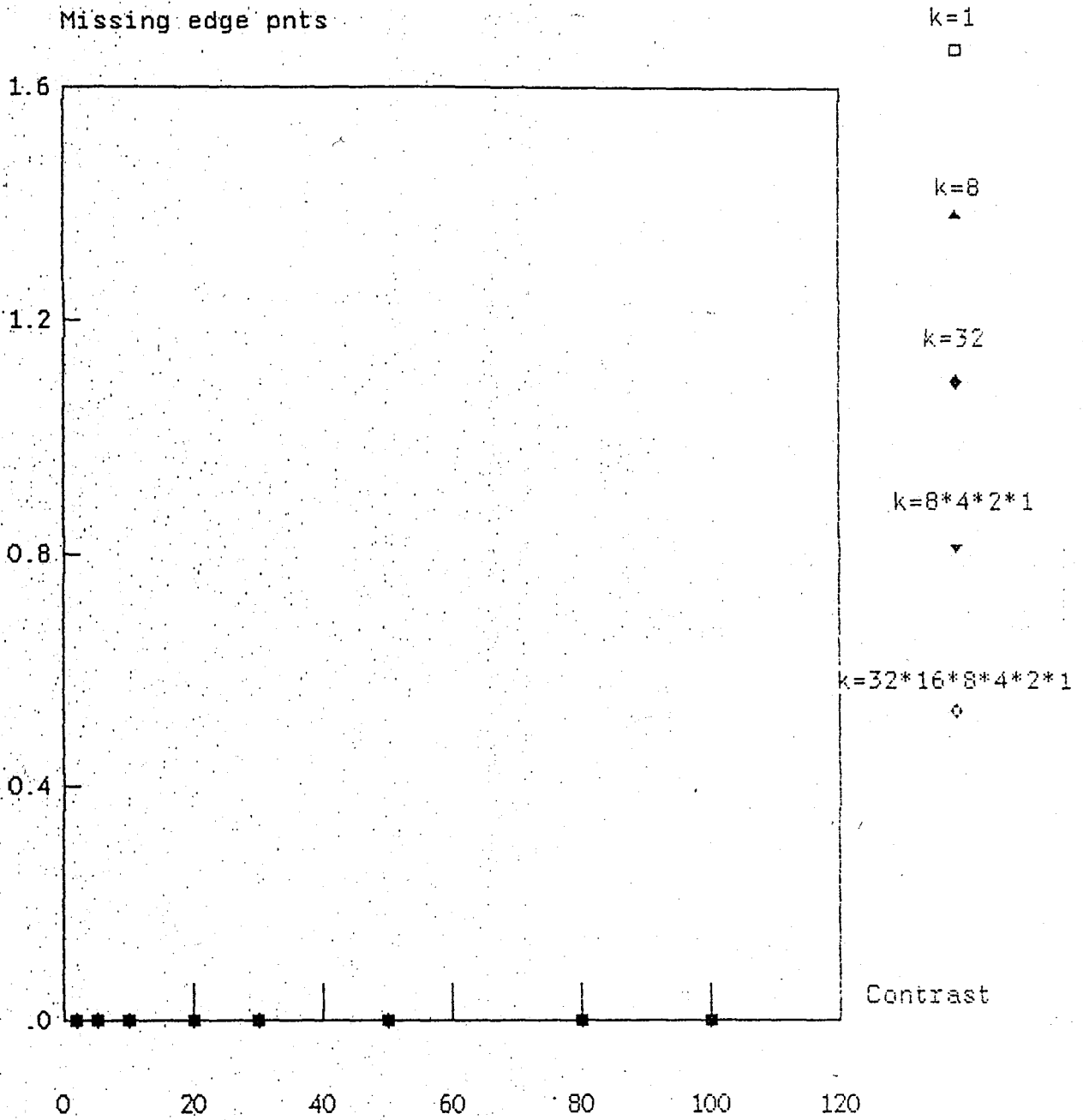


Fig:4.56 Missing edge points vs contrast for

fixed threshold, & w/o postprocessing without existence of noise

ROSENFELD DIFF. EQUATIONS

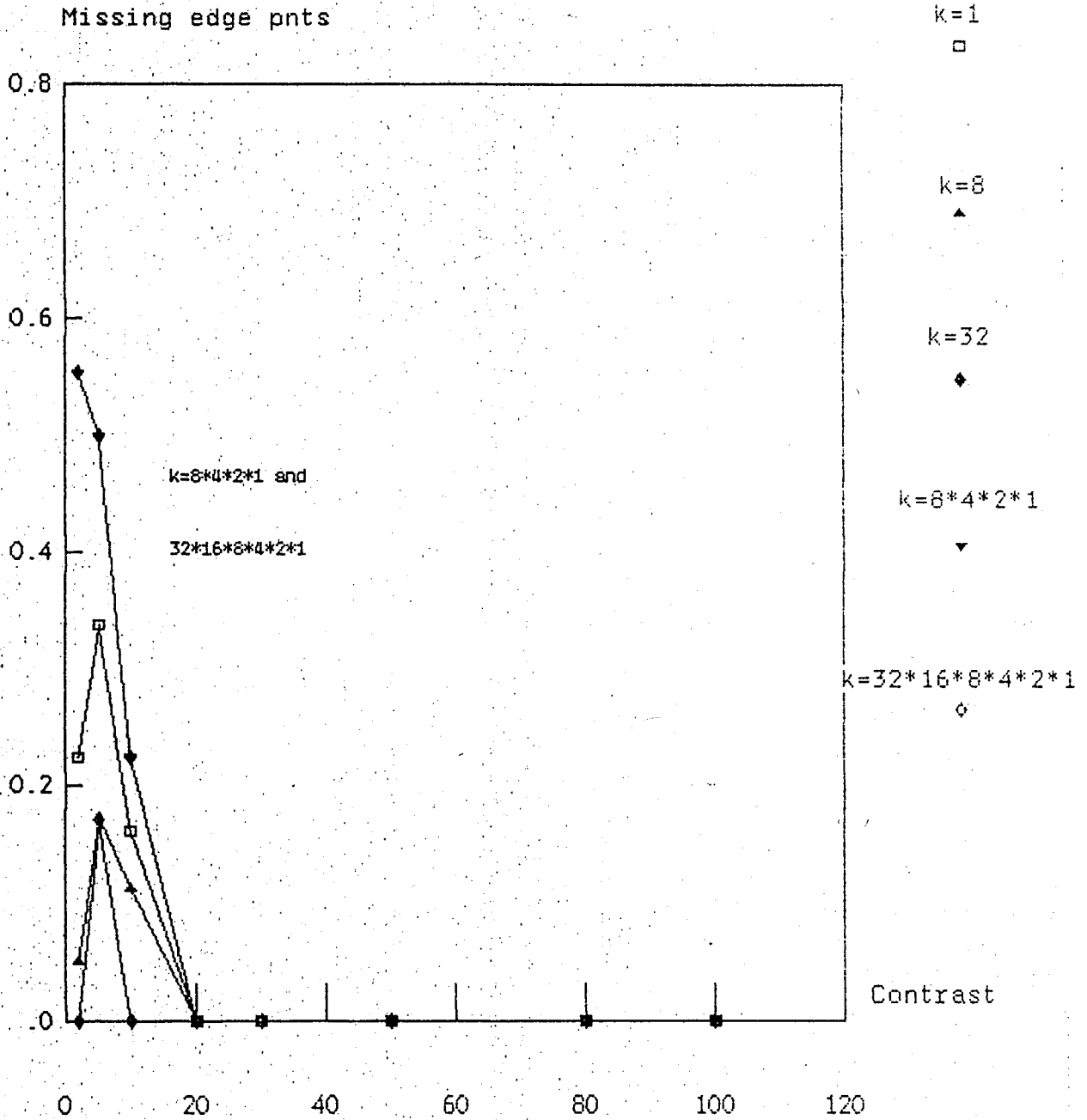


Fig. 4.57 Missing edge points vs contrast for fixed threshold, & w/o postprocessing with existence of noise ($\sigma = 40$)

Comparison between Sobel Operator, Kirsch Masks,

Rosenfeld diff. eqn 1 & 2 for $k=8$ and $k=32 \dots 1$ respectively

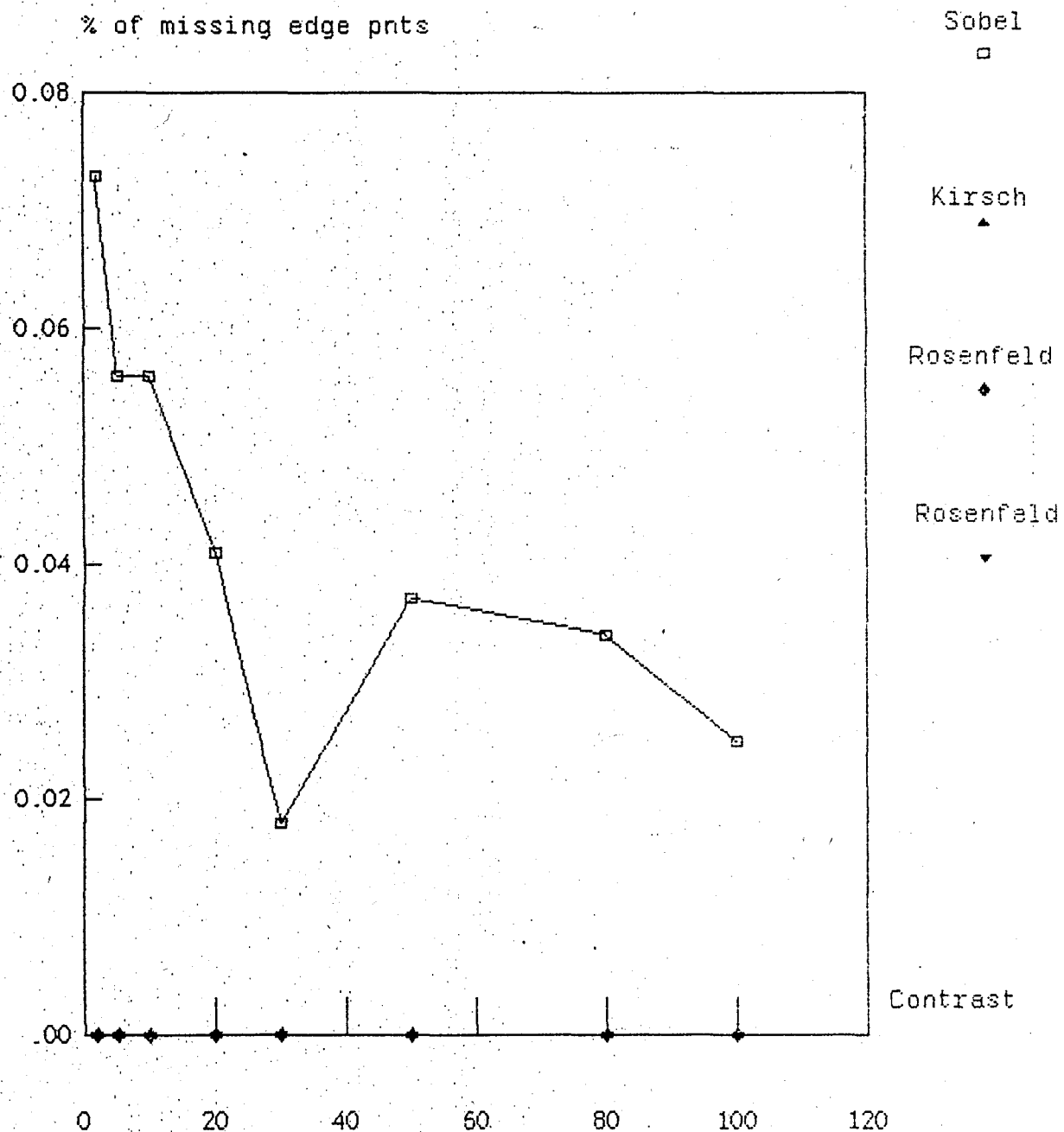


Fig. 4.58 Missing edge pnts vs contrast for fixed threshold

and without postprocessing and without the existence of noise

Comparison between Sobel Operator, Kirsch Masks,

Rosenfeld diff. eqn 1 & 2 for $k=8$ and $k=32 \dots 1$ respectively

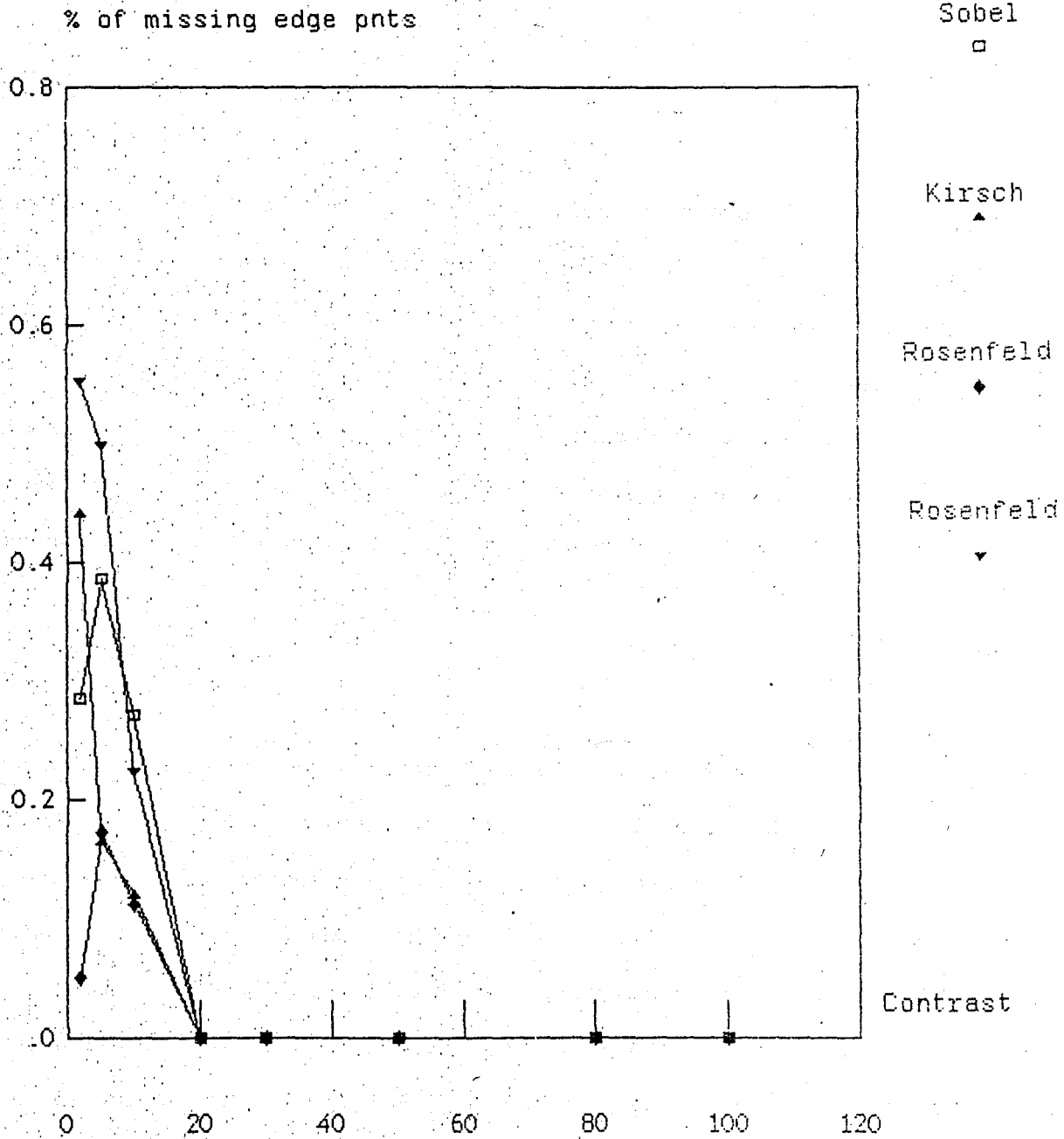


Fig. 4.59 Missing edge pnts vs. contrast for fixed threshold

and without postprocessing and with the existence of noise ($\sigma=40$)

error versus noise variance should be taken into consideration together with the missing edge points to evaluate the schemes.

2) This criterion gives better results when no postprocessing is used, because while sharpening the edges obtained by preprocessing, the postprocessing procedure may end up in losing some real edge points.

OPERATION TIME OF THE SCHEMES

The operation time of the schemes is considered as one of the performance criteria. The computer used for this study has an 8-bit processor and has very low operation speed. Because of memory limitations, disk accesses are required in many parts of the program. This also causes long operation times. In our case the speed of the printer and the disk accesses are not considered in order to make the evaluation independent from the computer. The number of operation during one run are calculated for each scheme.

A - SOBEL OPERATOR

For only the preprocessing algorithm $21(N*N)$ operations are required, where $N*N$ is the dimension of the test image. If a fixed threshold is used $16(N*N)$ operations, if an adaptive threshold is used then $25(N*N)$ operations must be added to the number of the preprocessing operations. For the postprocessing operation $18(N*N)+LIM+3(L*L)$ more operations are required, where LIM is the number limit, L is the number of 1's in the whole picture. Then, for the four different schemes using Sobel operator, the number of operations required can be summarized as follows.

1 - Sobel operator with fixed threshold without postprocessing:

$$O = 21(N*N)+16(N*N) = 37(N*N) \quad (4.14)$$

where O is the total number of operations.

2 - Sobel operator with fixed threshold and with postprocessing:

$$\begin{aligned} O &= 21(N*N)+16(N*N)+18(N*N)+LIM+3(L*L) \\ O &= 55(N*N)+LIM+3(L*L) \end{aligned} \quad (4.15)$$

3 - Sobel operator with adaptive threshold and without postprocessing:

$$O = 21(N*N)+25(N*N) = 46(N*N) \quad (4.16)$$

4 - Sobel operator with adaptive threshold and with postprocessing:

$$\begin{aligned} O &= 21(N*N)+25(N*N)+18(N*N)+LIM+3(L*L) \\ O &= 64(N*N)+LIM+3(L*L) \end{aligned} \quad (4.17)$$

B - KIRSCH MASKS

For only the postprocessing algorithm $162(N*N)$ operation are required. If a fixed threshold is used $16(N*N)$ operations, if an adaptive threshold, i.e., convolving with LPF, M_0 is used $26(N*N)$ operations must be added to the number of the preprocessing operations. For postprocessing operation which is the connectivity test in the case of the Kirsch masks, $45(N*N)$ operations are required in addition of the number. For the four different schemes using Kirsch masks, the number of operations required can be summarized as

follows.

1 - Kirsch masks with fixed threshold and without postprocessing:

$$O = 162(N*N)+16(N*N) = 178(N*N) \quad (4.18)$$

2 - Kirsch masks with fixed threshold and with postprocessing:

$$O = 162(N*N)+16(N*N)+45(N*N) = 223(N*N) \quad (4.19)$$

3 - Kirsch masks with adaptive threshold and without postprocessing:

$$O = 162(N*N)+26(N*N) = 188(N*N) \quad (4.20)$$

4 - Kirsch masks with adaptive threshold and with postprocessing:

$$O = 162(N*N)+26(N*N)+45(N*N) = 233(N*N) \quad (4.21)$$

C - ROSENFELD DIFFERENCE EQUATIONS

The value of k is very important for Rosenfeld's edge detection schemes. The generalized formula for the number of operations of difference equations:

$$O = (k+1)(N*N)+kM(N*N) \quad (4.22)$$

where k is the order of difference and M is calculated by the formula

$$k = 2^B \quad (4.23)$$

$$M = B+1 \quad (4.24)$$

Then, for example if $k=1$,

$$O = 2(N*N)+(N*N) = 3(N*N) \quad (4.25)$$

If $k=8$ $O = 9(N*N)+32(N*N) = 41(N*N)$ (4.26)

If products of the differences are used, the number of operations can be calculated by the formula:

$$O = (18k+1)(N*N)+kM(N*N) \quad (4.27)$$

For example, if $k=4$ (i.e., $4*2*1$), then

$$O = (18*4+1)(N*N)+4*3(N*N) = 85(N*N) \quad (4.28)$$

If $k=32$, $O = (18*32+1)(N*N)+32*6(N*N) = 769(N*N)$ (4.29)

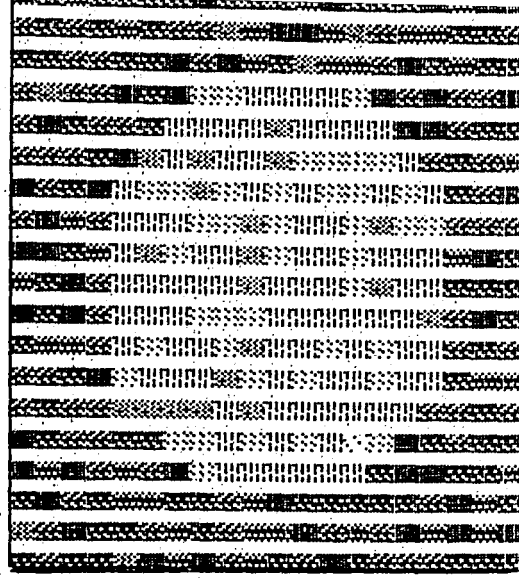
If the above equations from (4.14) through (4.29) are examined, the smallest number of operations corresponds to the Sobel operator scheme which uses fixed threshold and the largest one corresponds to Rosenfeld's products of differences for $k=32$. Adaptive thresholding always adds a considerably large number of operations. Postprocessing algorithms also add large number of operations.

ABILITY TO DETECT CURVED EDGES

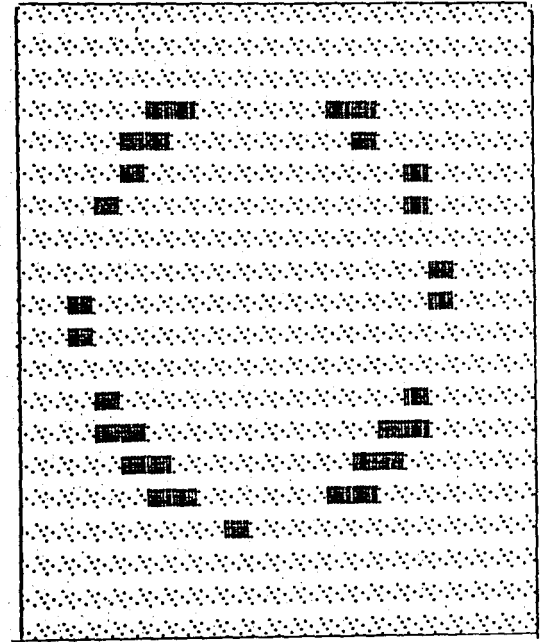
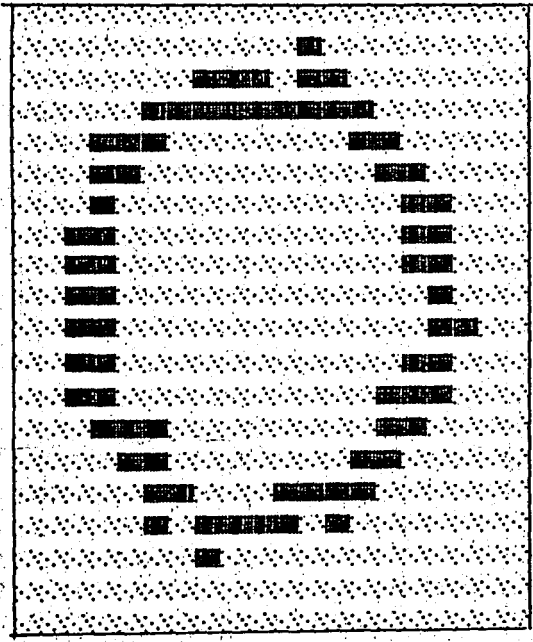
Fig (4.60) through (4.65) show the behaviour of the edge detection schemes on curved edges, e.g., on circular shapes. The best method for detecting curved edges is the method of Kirsch without connectivity test. But when connectivity test is applied, some edge points are lost, then the shape of the edge can not be realized properly. Rosenfeld's schemes give good results at higher values of k and for products of differences. If differences are used without taking the products of them, the larger the value of k the wider the edge region gets, then the shape of the edge is lost. This is shown in Fig (4.62) and (4.65). Sobel operator can also give better results by using postprocessing algorithm as limiting the number of 1's. If the limit is chosen properly all 1's which are outside of the edge region, can be eliminated and all 1's which are inside of the edge region can be saved.

ABILITY TO DETECT SHARP EDGES

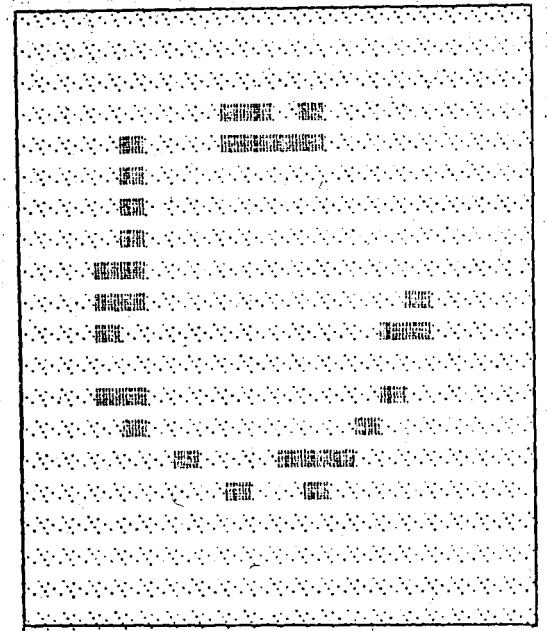
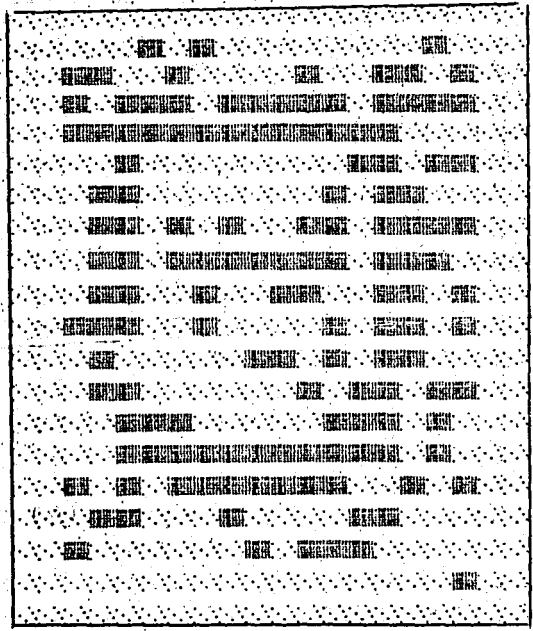
Fig (4.66) through (4.68) show all edge detection schemes with fixed and adaptive threshold without and with postprocessing algorithms when a vertical image with contrast $C=30$ is used without the existence of noise. It is obvious that, Kirsch masks with adaptive threshold without using any postprocessing algorithm gives the most exact and sharpest edges. Rosenfeld difference equations give good results when the large values of k and the products of differences are used.



a

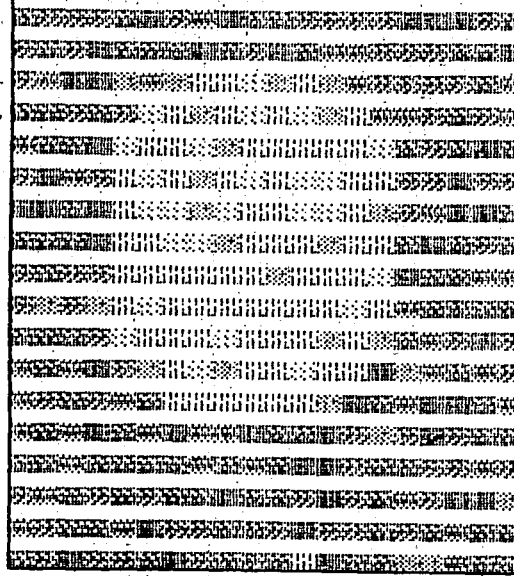


b

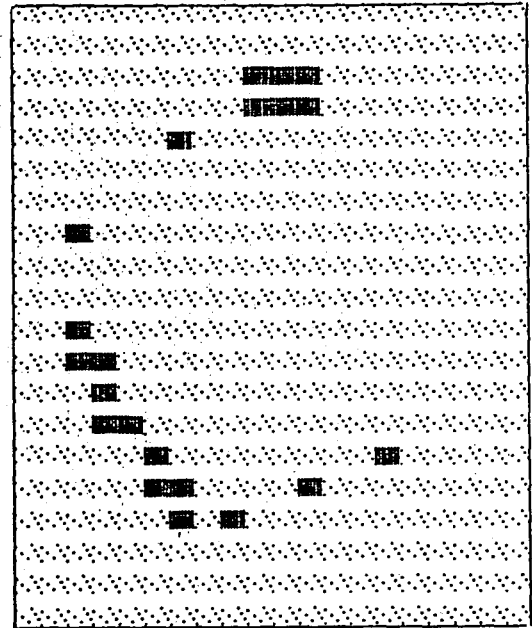
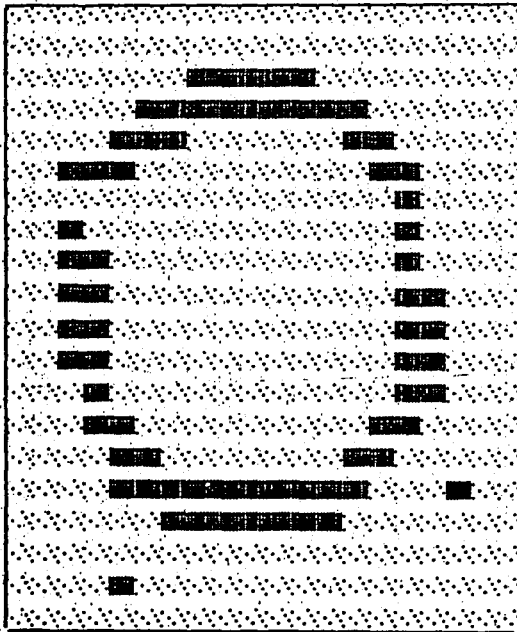


c

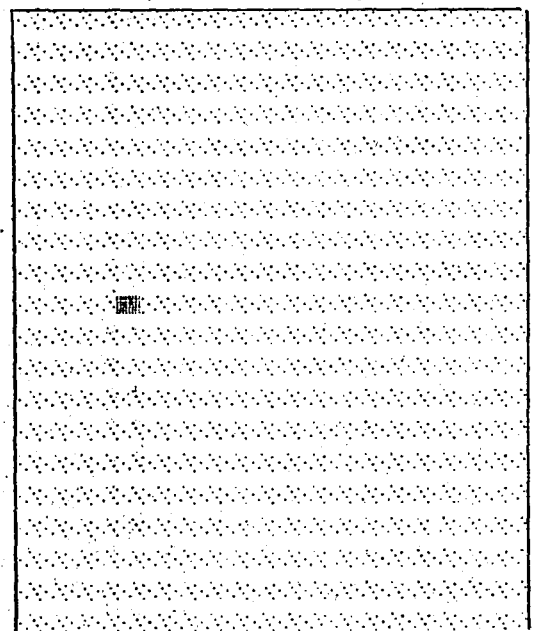
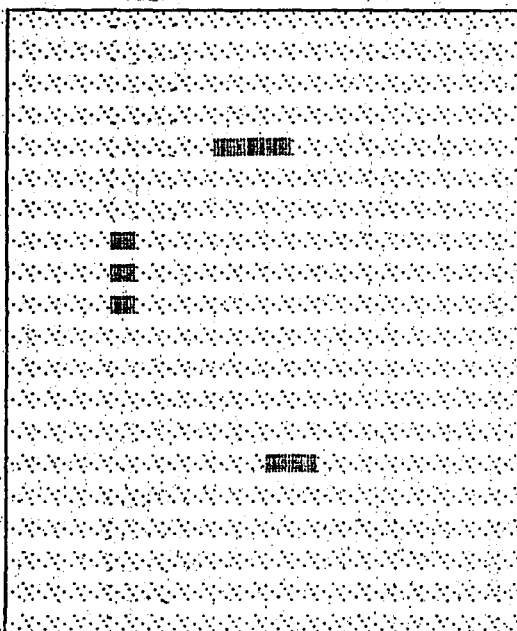
Fig 4.60 Behaviour of the edge detection scheme using Sobel operator on curved edges with $C=2$ a)Original image b)Fixed thr.image and output of postpro.alg.c)Adapt.thr.image and output of postpro.algor



a

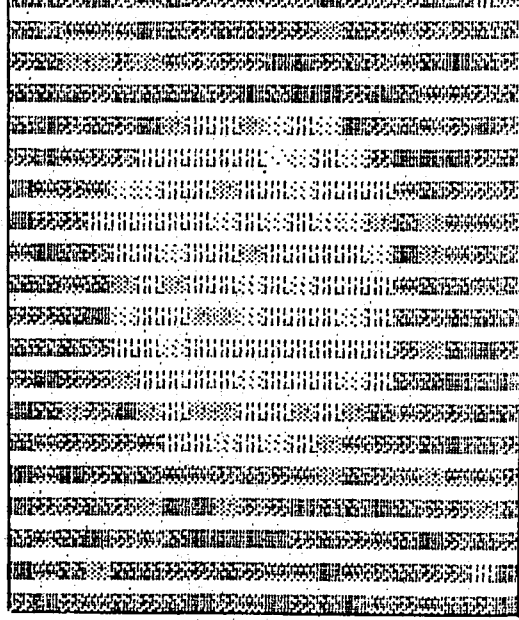


b

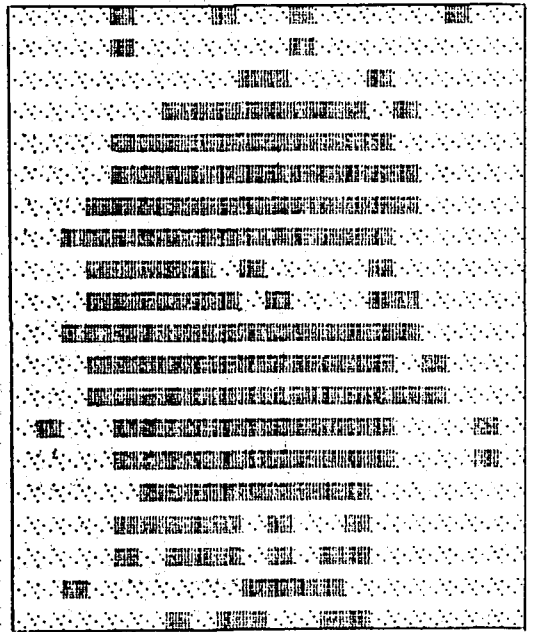
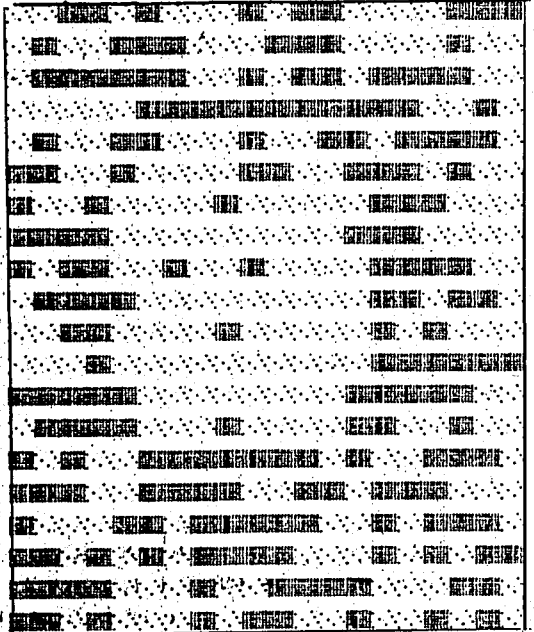


c

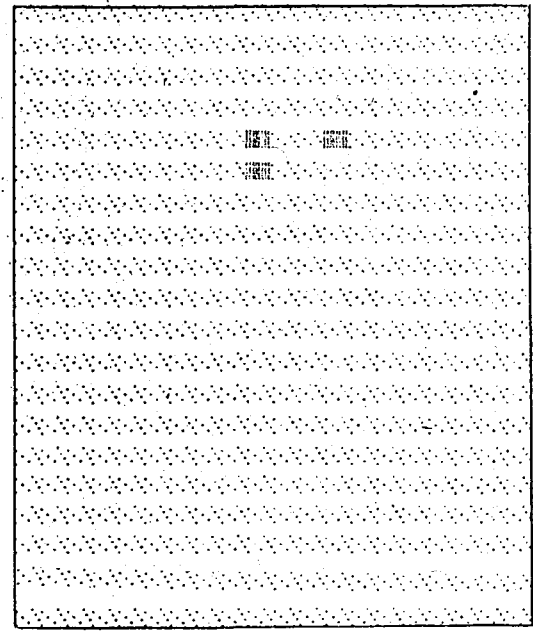
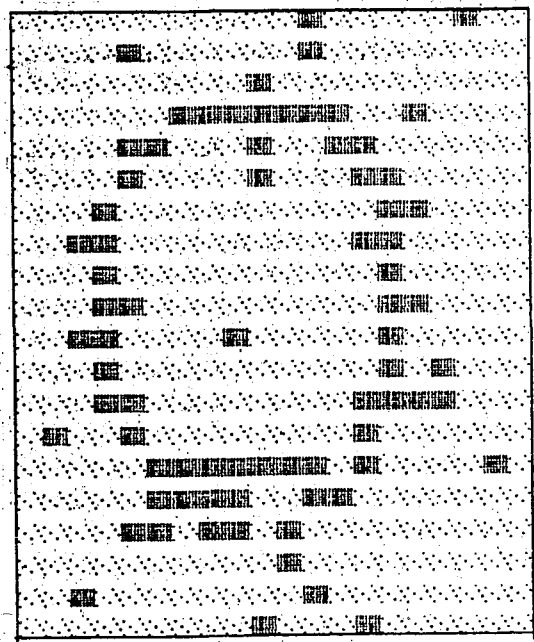
Fig 4.61 The behaviour of Kirsch masks on curved edges with $C=2$
 a)Original image b)Fixed thr.image and output of postpro.algor.
 c)Adaptive thr.image and output of postproc.algor.



a

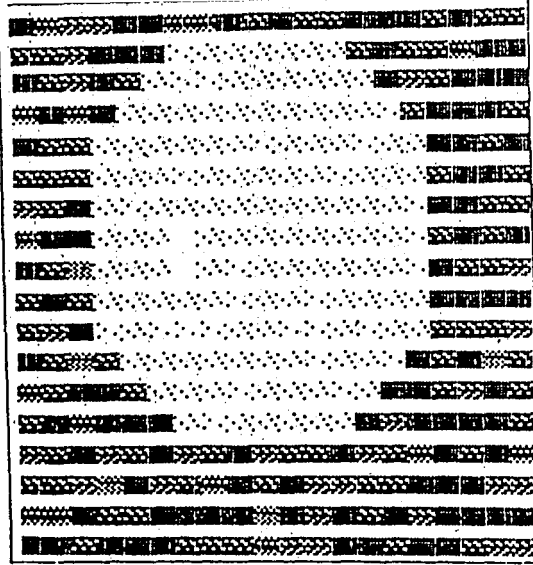


b,c

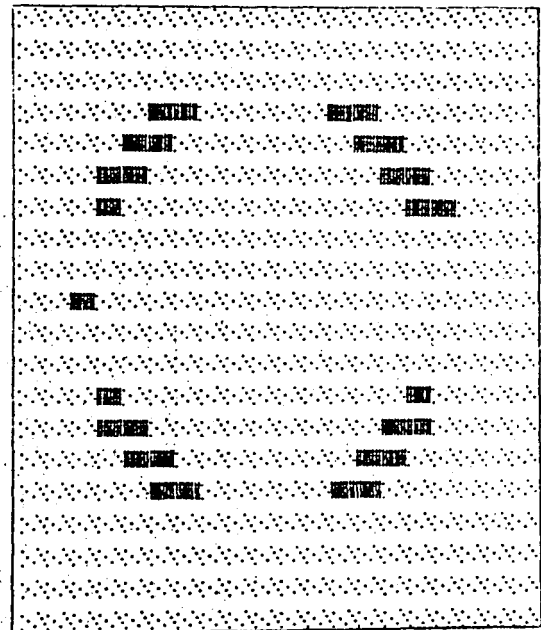
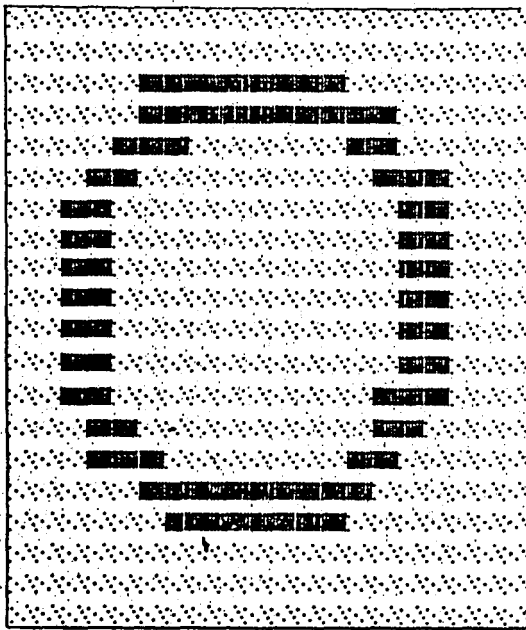


d,e

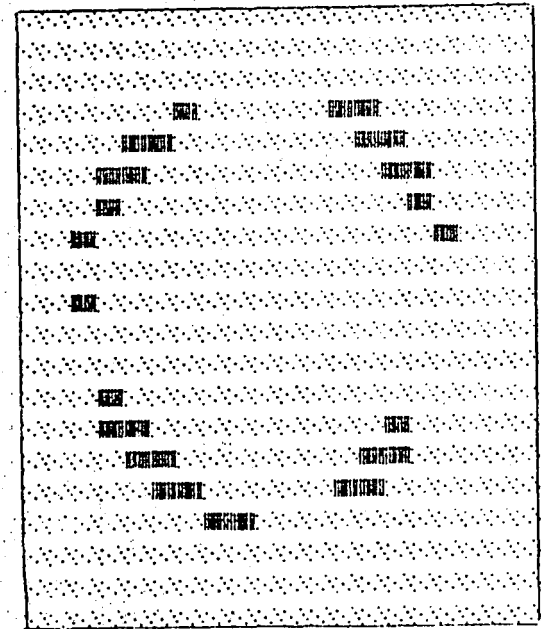
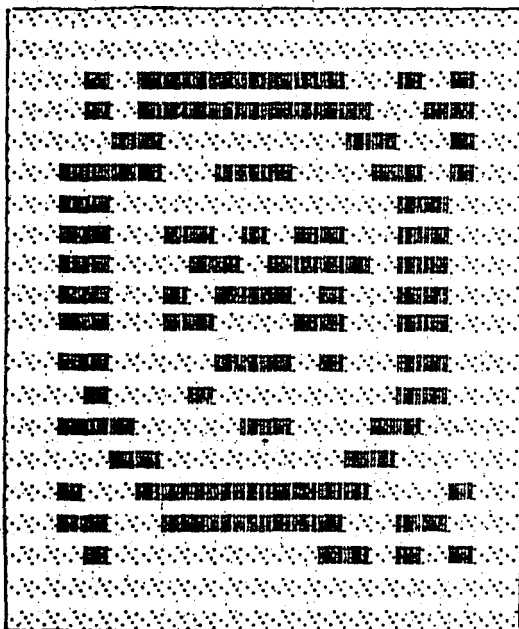
Fig 4.62 The behavior of Rosenfelds algorithm on curved edges with C=2
 a)Original image b)Fixed thr.image for k=1 c)k=8 d)k=8*4*2*1
 e)k=32*16*8*4*2*1



a

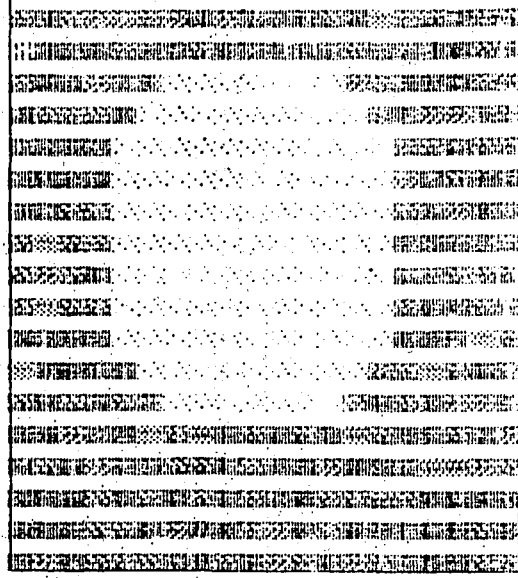


b

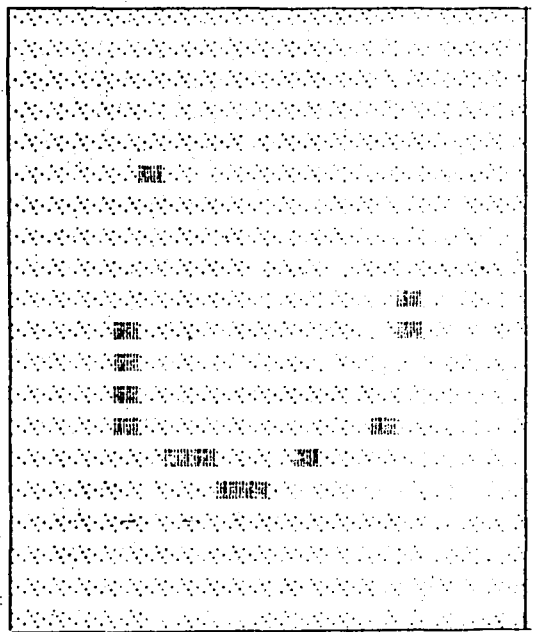
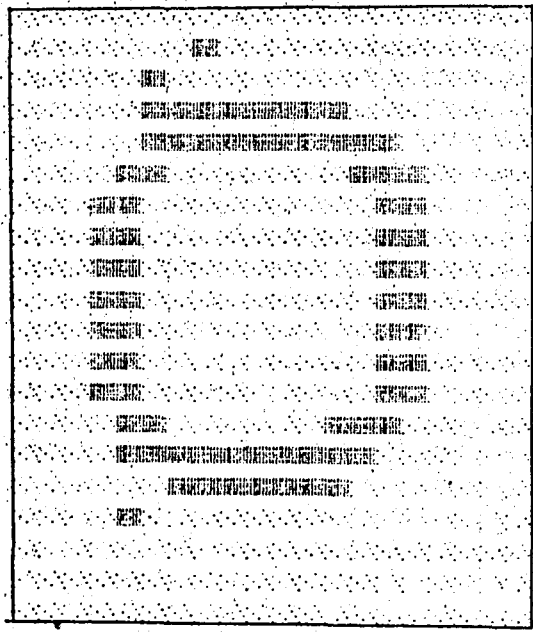


c

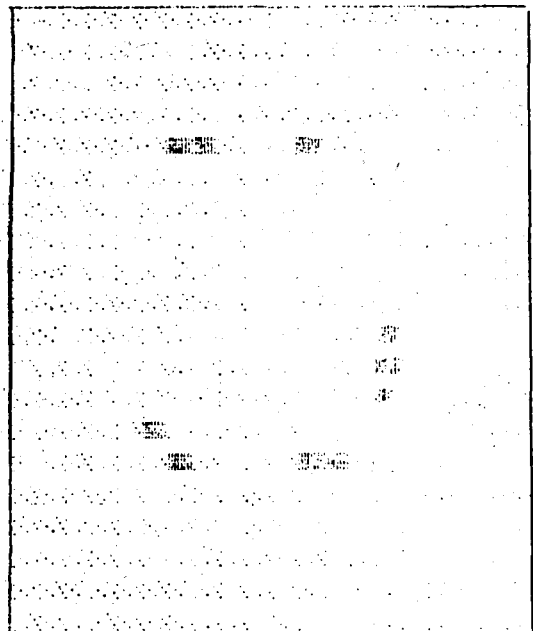
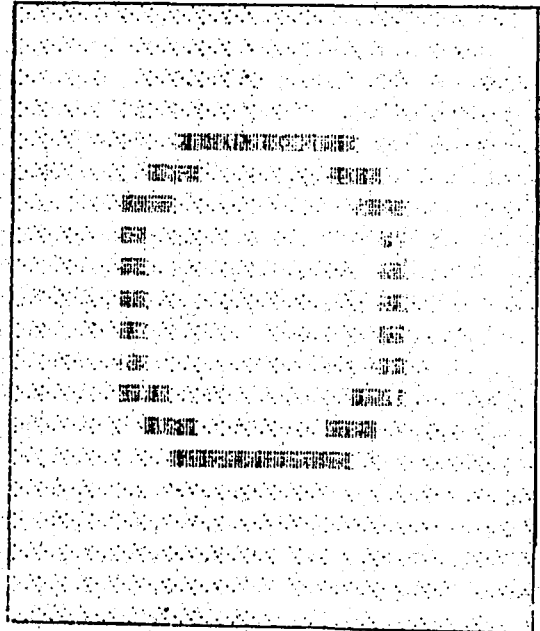
Fig 4.63 The behaviour of the Sobel operator on curved edges with $C=20$
 a)Original image b)Fixed thr.image and output of postproc.algor.
 c)Adaptive thr.image and output of postproc.algor.



a

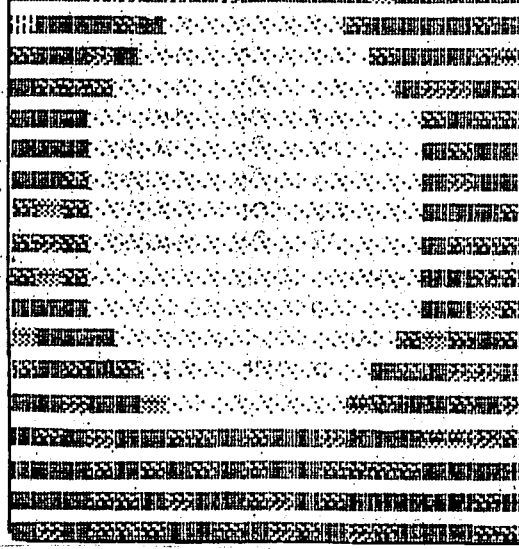


b

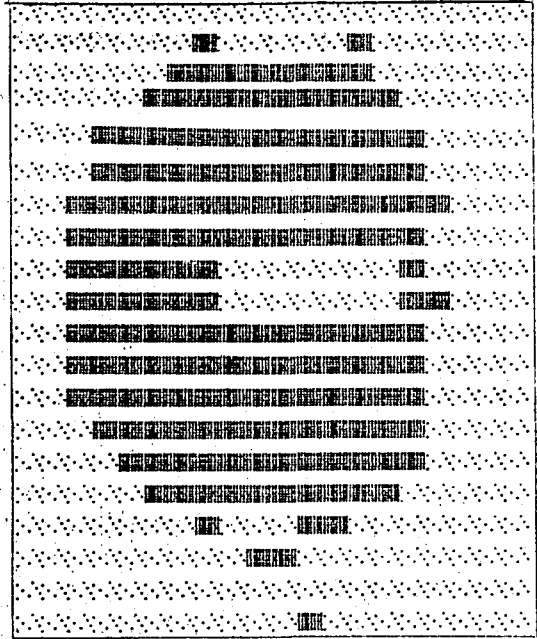
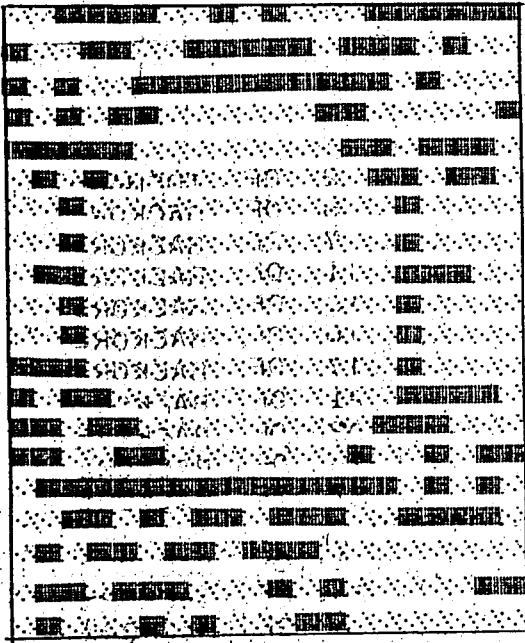


c

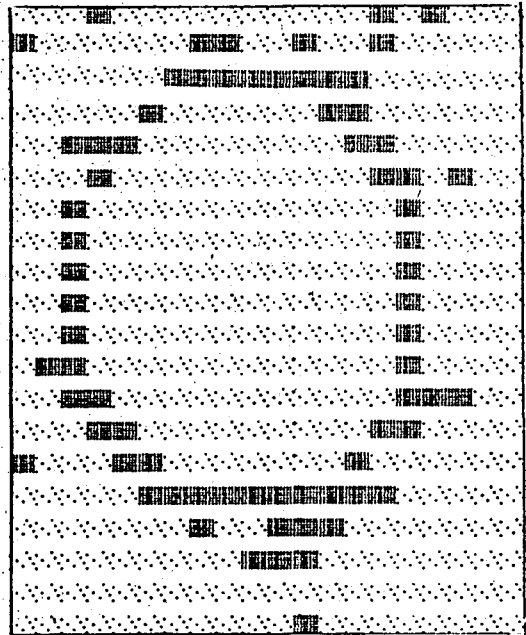
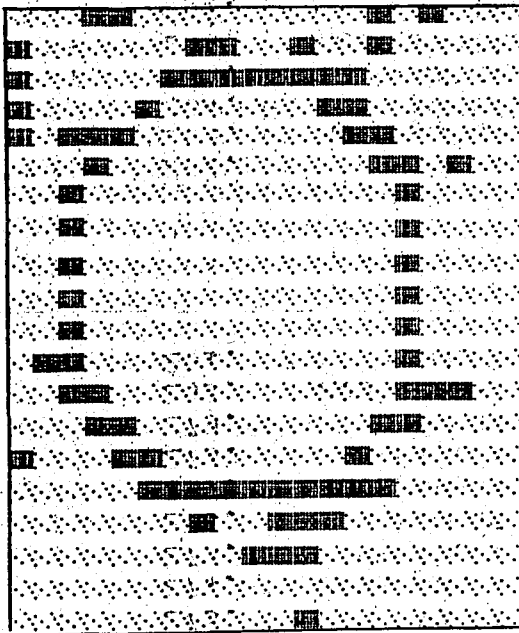
Fig 4.64 The behaviour of the Kirsch masks on curved edges with $C=20$
 a)Original image b)Fixed thr.image and output of postproc.algor.
 c)Adaptive thr.image and output of postproc.algor.



a

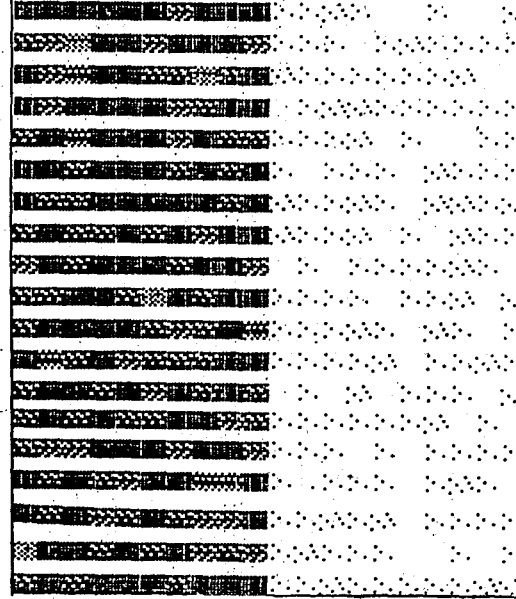


b,c

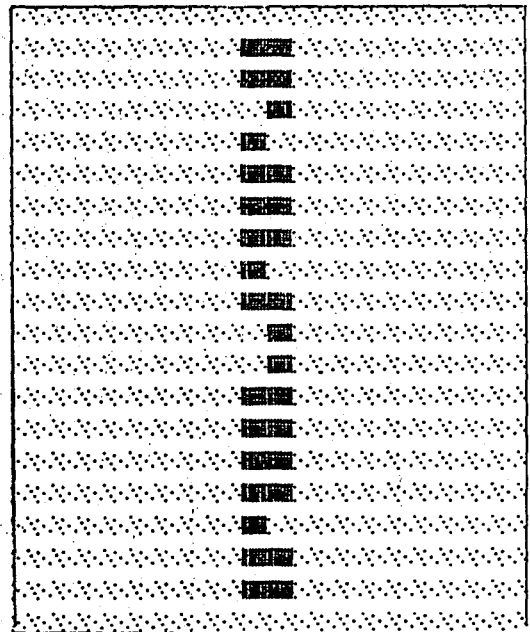
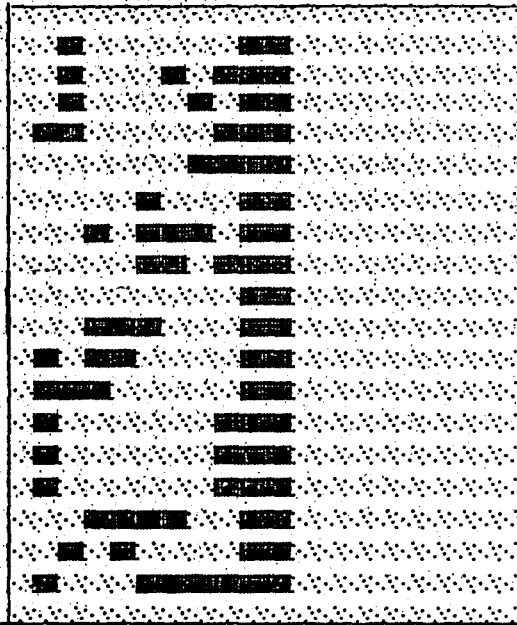


d,e

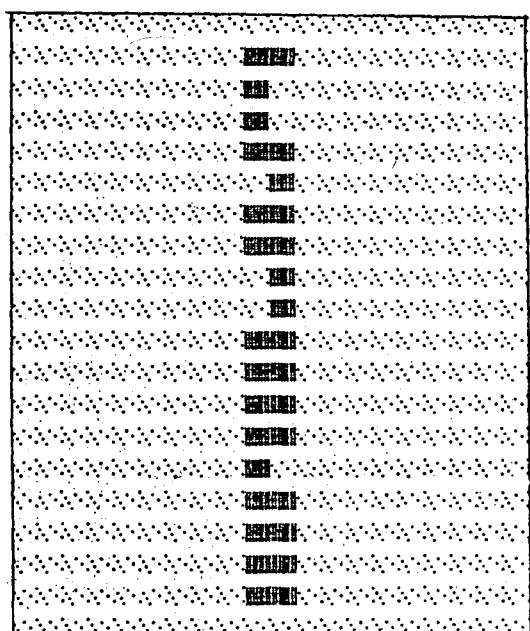
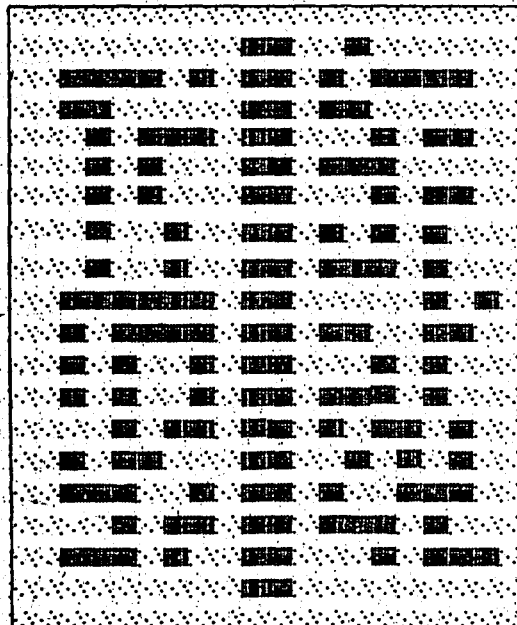
Fig 4:65 The behaviour of Rosenfeld's algorithm on curved edges with $C=20$. a)Original image b)Fixed thr.image $k=1$ c) $k=8$ d) $k=8*4*2*1$ e) $k=32*16*8*4*2*1$



a

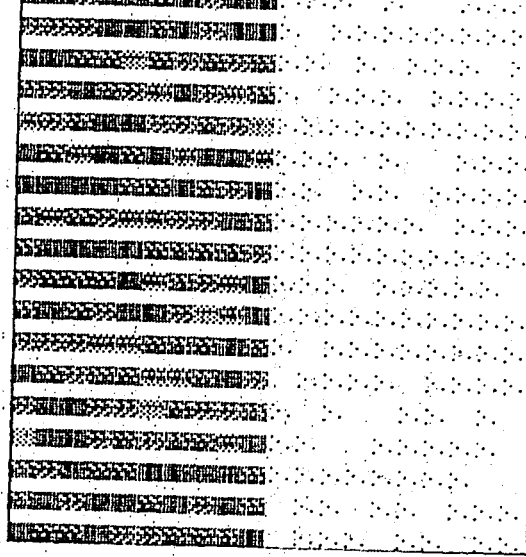


b

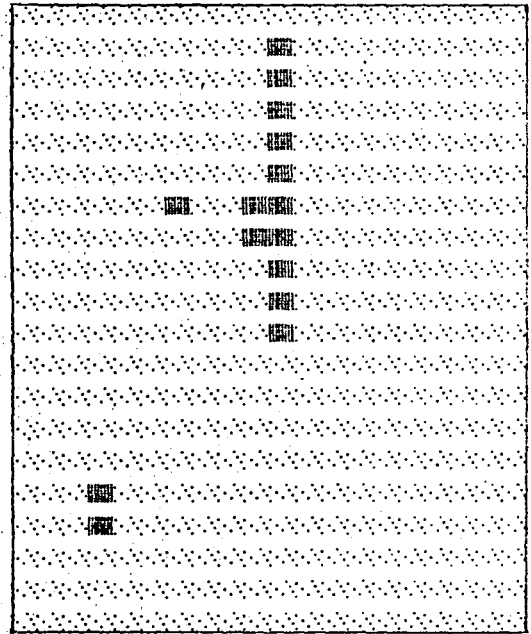
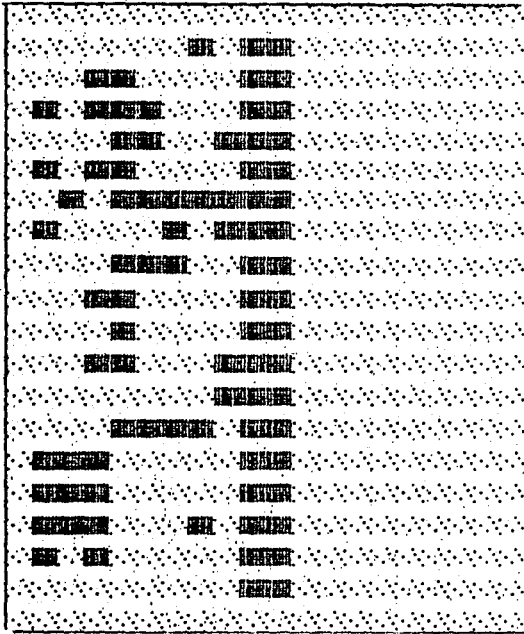


c

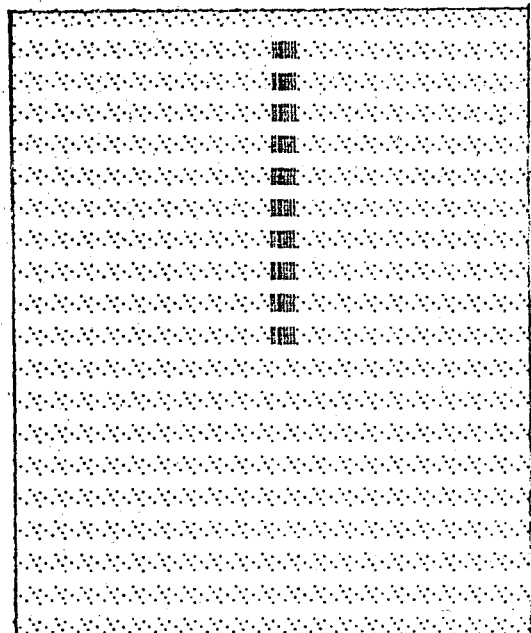
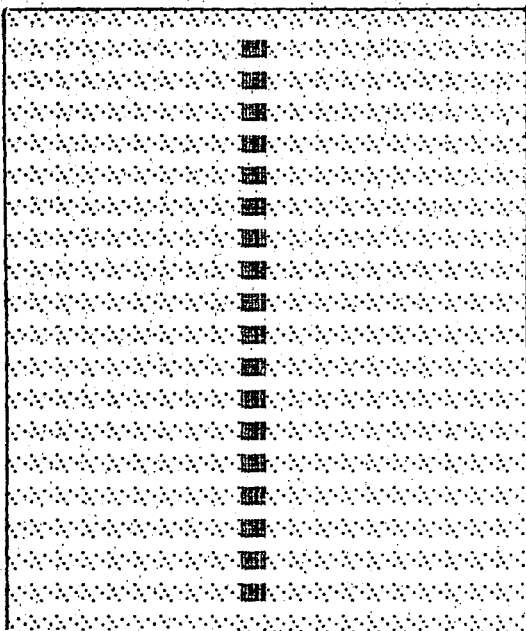
Fig 4.66 The ability of the Sobel operator to detect edges sharply with $C=30$ a)Original image b)Fixed thr.+postproc. c)Adap.thr.+postproc.



a

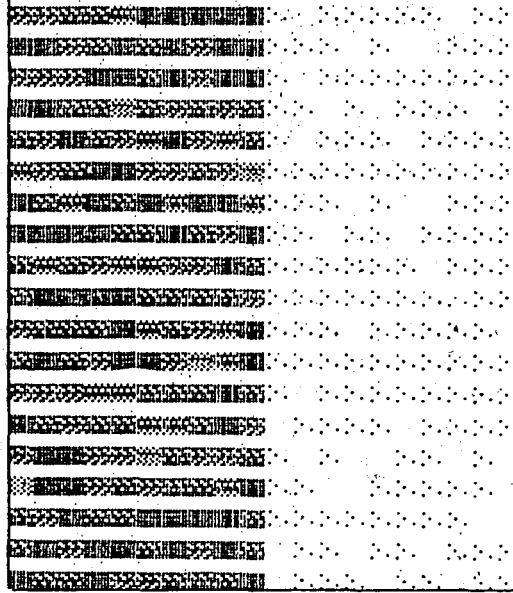


b

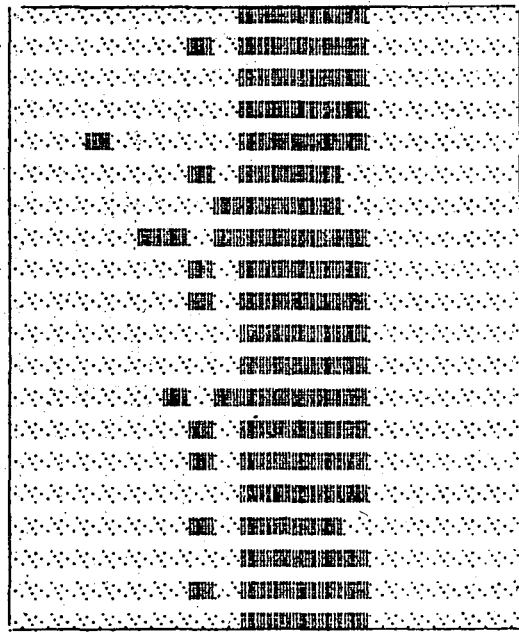
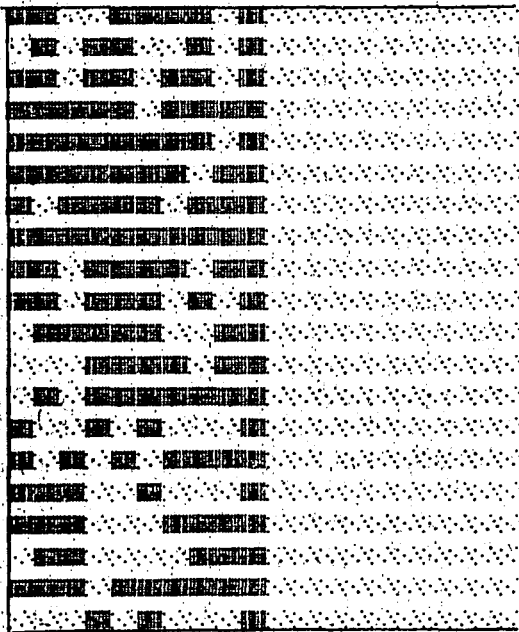


c

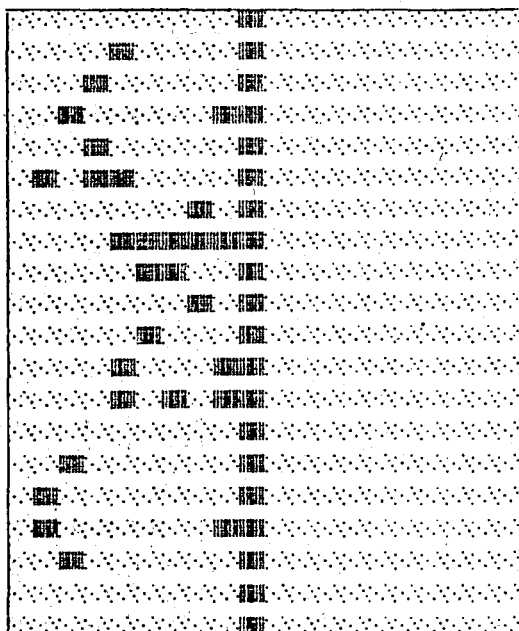
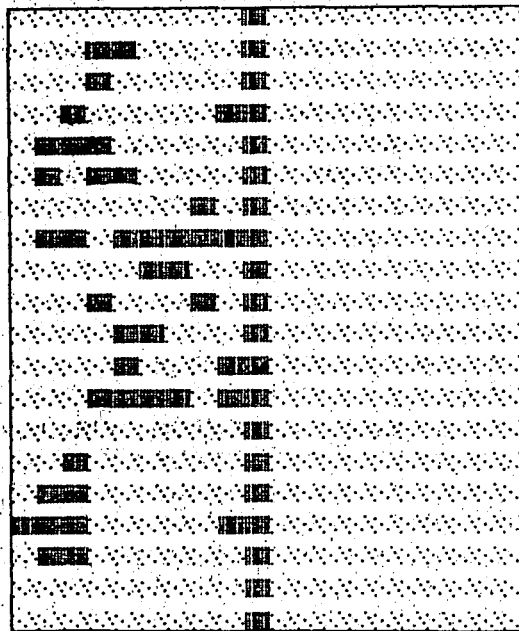
Fig 4.67 The ability of Kirsch masks to detect edges sharply with $C=30$
a) Original image b) Fixed thres.+postproc. c) Adaptive thres.+postproc.



a



b,c



d,e

Fig 4.68 The ability of Rosenfeld's algor. to detect edges sharply with C=30 a)Original image b)Thres.image k=1 c)k=8 d)k=8*4*2*1 e)k=32*...*1

4.6 CONCLUSION

In this chapter the most important edge detection methods were overviewed and some new ideas on decision levels and performance evaluations are introduced.

Edge detection methods can be divided into three levels: Preprocessing labeling and postprocessing. A key level is the one, where the decision to label a pixel as an edge point or not, has to be taken. Before this labeling stage, in order to make the decision as correct as possible, the pictures must be preprocessed.

Many different preprocessing and labeling algorithms have been developed. The most popular preprocessing schemes are included in the class of the local methods which attempt to approximate differentiation within a small window of size 2×2 or 3×3 . For labeling level many different ideas have been introduced. But the most commonly used parameter, especially for practical implementation reasons, is the magnitude of the edge vector at a given pixel. Comparison of various edge detection schemes are mostly dependent on their qualitative performances. Papers about the quantitative evaluation of edge detection schemes are very limited. A performance evaluation is difficult because of the large number of proposed methods, difficulties in determining the best parameters associated with each technique, and the lack of definitive performance criteria.

In the last section of this chapter, the connectivity test and a locally

adaptive thresholding technique which was introduced by Robinson has been adapted to the Kirsch masks. Another combination of two thresholding techniques has been applied to Sobel operator and a comparison between the most popular edge detection schemes was done and some new parameters for performance evaluation has been generated. Because of the memory limitations and lack of facilities, the comparison was done on some test images. Evaluation was also done in the presence of noise.

The results can be summarized as follows:

- (1) - The connectivity test is not suitable for Kirsch masks, especially with adaptive thresholding;
- (2) - Without connectivity test, Kirsch gives sharpest edge points with locally adaptive thresholding using LPF;
- (3) - Sobel operator is more sensitive than the other methods to edges which have very little contrasts;
- (4) - Rosenfeld's multiple difference equation gives very sufficient results at higher multiples, but it takes the biggest operation time, while Sobel with fixed threshold takes the least .
- (5) - Postprocessing algorithms can be applied to all edge detection schemes in order to eliminate the 1's outside the edge region and to get sharper edges;
- (6) - Two-level thresholding(e.g., first adaptive then fixed thresholding) can be applied to edge detection schemes to get better results;
- (7) - The Parameter P is very useful to understand the behaviour of the scheme with respect to the changes in the contrast value with the existance of noise. In addition, a large value for P does not mean that good results are obtained, because this large value is obtained by not only decrease in

noise effect but also increase in % error for pure image when the contrast increases;

(8) - For large values of noise variance, three preprocessing algorithm almost give the same response;

(9) - Mean square distance of errors does not give any reasonable responses and can not be used for the evaluation of the edge detection schemes;

(10) - Different schemes respond to each criterion in a different manner. Sometimes give better, sometime give worse results, means that a tester must choose the most convinient scheme according to his requirements and criteria he uses.

At the end of the chapter a set of package programs which are open to further expansion consists of edge detection techniques , various thresholding methods and complete analysis has been also introduced.

V CONCLUSION AND SUMMARY

In this study the most important aspects of image processing techniques and applications were reviewed. Although it was a relatively long study, all the important aspects of image processing could not be covered. The first part of the study was only a tutorial review of mathematical models and applications of image processing. In the second part of the study, various picture coding techniques were examined and their evaluation were done by computer simulations. In the last part, various edge detection schemes using different thresholding techniques and postprocessing algorithms were evaluated on some test images. Some performance criteria were developed in order to make a comparison among these schemes. A computer program was developed to simulate the picture coding techniques. In the last part, edge detection schemes were reviewed, some performance criteria were derived and comparison of the schemes was done using these criteria. Again a package program was developed and simulation results were obtained.

This study has three different aspects. The first one is that, it is a review of all image processing problems and methods to solve these problems and it might be helpful for further studies on the same topic.

The second one is that, a microcomputer is used to simulate picture compression, coding and edge detection techniques. This computer applications have given us important results on two picture processing techniques, but in addition to that, during this study, the advantages and disadvantages of working with a microcomputer for similar topics have been

found. The advantages are,

- 1 - Ability to work interactively
- 2 - Easy use and programming

But the term image implies a large number of data, besides picture processing techniques are based on complex algorithms, like matrix manipulations. In this study the memory capacity and the operation time of the computer set limits on the size of the model. For test images only 64 samples could be taken at the same time. In the literature, the minimum number of samples used for images is 256, but even it could be worked only with very small number of samples, very reasonable results which are comparable with literature were obtained. The complex structure of the algorithms has resulted in rather great operation times. That is why only minimum number of simulation outputs could be taken for evaluation purposes.

The third aspect is that, some performance criteria have been derived in order to evaluate and compare the results of the simulations. In the literature very few quantitative comparisons exist and evaluations are mostly done qualitatively. Because edge detection schemes have very different algorithms, it is difficult to compare them. That is why in order to evaluate and compare the edge detection schemes examined in this study, new quantitative performance criteria have been derived. If the results of the simulations of different edge detection schemes are observed, it can be seen that, for different criteria, different schemes can give better results. But on the average it can be said that Kirsch's directional masks with adaptive threshold give the best result among all other schemes which are examined in this study for almost all performance criteria.

A comprehensive bibliography is included at the end of the study for a reader interested in further details of theoretical and experimental results discussed here.

A P P E N D I X A

PROGRAM LISTINGS OF
PICTURE COMPRESSION
TECHNIQUES


```

10 REM    ** QUANTIZATION MAIN PROG. **
20 TEXT : HOME
25 HTAB 10: PRINT "-----"
30 D$ = CHR$ (4)
40 HTAB 10: PRINT "QUANTIZATION PROGRAM"
50 HTAB 10: PRINT "-----"
270 HOME
280 VTAB 6: HTAB 5: INVERSE : PRINT "WAVEFORMS": NORMAL
290 VTAB 10: HTAB 5: PRINT "1-SINE WAVE"
300 HTAB 5: PRINT "2-SQUARE WAVE"
310 HTAB 5: PRINT "3-RANDOM SINE WAVE"
320 HTAB 5: PRINT "4-AUT.REG.WAVEF.WITH GAUSS DEN."
330 VTAB 20: HTAB 5: INPUT "CHOOSE ONE OF THEM..=";A
400 IF A > 4 THEN PRINT CHR$ (7): GOTO 330
770 ON A GOTO 2000,3000,4000,5000
2000 PRINT D$;"RUN SIMPROG,S6,D1"
3000 PRINT D$;"RUN KARPROG,S6,D1"
4000 PRINT D$;"RUN RASPROG,S6,D1"
5000 PRINT D$;"RUN GAUPROG,S6,D1"

```

```

2000 REM ** SIMPROG **
2010 D$ = CHR$(4)
2014 HOME : VTAB 10: INPUT "TOTAL # OF SAMPLES ?=";N$
2015 N = VAL (N$)
2016 PRINT D$;"OPEN ORNEK,D2"
2018 PRINT D$;"DELETE ORNEK"
2019 PRINT D$;"OPEN ORNEK,D2"
2020 PRINT D$;"WRITE ORNEK"
2021 PRINT N$
2022 PRINT D$;"CLOSE ORNEK"
2024 DIM Y(N)
2025 DIM ARALIK(N)
2030 PI = 3.14159265
2035 PRINT D$;"PR#1": PRINT "TOT.# OF SAMPLES=";N: PRINT D$;"PR#0"
2040 PRINT : INPUT "AMPLITUDE ?=";F$
2050 A = VAL (F$)
2060 PRINT : INPUT "MULT.VALUE OF X-COORD. =?";G$
2070 CARP = VAL (G$)
2072 PRINT : INPUT "# OF PERIODS ?=";S$
2073 K = VAL (S$)
2075 PRINT : PRINT : PRINT
2080 PRINT "X(I)";: HTAB 20: PRINT "Y(I)"
2090 PRINT "-----";: HTAB 20: PRINT "-----"
2110 ARG = 2 * PI * K / N
2150 FOR I = 1 TO N
2160 ARALIK(I) = ARG * (I - 1)
2170 Y(I) = A * SIN (ARALIK(I))
2180 PRINT ARALIK(I);: HTAB 20: PRINT Y(I)
2270 NEXT I
2280 PRINT D$
2290 PRINT D$;"OPEN GIRDİ,D2"
2300 PRINT D$;"DELETE GIRDİ"
2310 PRINT D$;"OPEN GIRDİ,L11,D2"
2320 FOR I = 1 TO N
2322 INP$ = STR$(Y(I))
2324 IF LEN (INP$) = 10 THEN GOTO 2330
2325 IF (LEN (INP$) > 10 AND VAL (INP$) < .01 AND VAL (INP$) > 0) THEN INP$ = ".001000000": GOTO 2330
2326 IF LEN (INP$) > 10 THEN INP$ = LEFT$(INP$,10): GOTO 2330
2327 FOR L = LEN (INP$) TO 10:INP$ = INP$ + " ": NEXT L
2330 PRINT D$;"WRITE GIRDİ,R";I
2332 PRINT INP$
2334 NEXT I
2336 PRINT D$;"CLOSE GIRDİ"
2340 PRINT D$;"OPEN ARALIK,D2"
2342 PRINT D$;"DELETE ARALIK"
2344 PRINT D$;"OPEN ARALIK,L11,D2"
2346 FOR I = 1 TO N
2348 INP$ = STR$(ARALIK(I))
2350 IF LEN (INP$) = 10 THEN GOTO 2370
2352 IF (LEN (INP$) > 10 AND VAL (INP$) < .01 AND VAL (INP$) > 0) THEN INP$ = ".001000000": GOTO 2370

```

```
2354 IF LEN (INP$) > 10 THEN INP$ = LEFT$ (INP$,10): GOTO 2370
2356 FOR L = LEN (INP$) TO 10:INP$ = INP$ + " ": NEXT L
2370 PRINT D$;"WRITE ARALIK,R";I
2372 PRINT INP$
2376 NEXT I
2380 PRINT D$;"CLOSE ARALIK"
2382 PRINT D$;"OPEN IST,D2"
2384 PRINT D$;"DELETE IST"
2386 PRINT D$;"OPEN IST,L3,D2"
2388 FOR I = 1 TO N
2390 PRINT D$;"WRITE IST,R";I
2392 IF Y(I) < 0 THEN VAR$ = "-1": GOTO 2396
2394 VAR$ = "+1"
2396 PRINT VAR$: NEXT I
2398 PRINT D$;"CLOSE IST"
2400 HOME : VTAB 22: HGR
2420 X = FRE (0)
2430 HPLOT 0,100 TO 270,100
2440 HPLOT 0,0 TO 0,190
2450 FOR I = 1 TO N
2460 XI = ARALIK(I) : CARP
2470 YI = Y(I)
2480 HPLOT TO 0 + XI,100 - YI
2490 NEXT I
2500 PRINT "SINE WAVE"
2505 PRINT D$
2510 PRINT D$;"PR#1": PRINT CHR$ (9);"6": PRINT D$;"PR#0"
2520 TEXT
2530 HOME
2560 PRINT D$
5000 PRINT D$;"RUN QUANPROG,D1"
```

```

10 HOME
15 D$ = CHR$(4)
20 REM ** AUT.REG.PROCESS S/R **
25 HOME : VTAB 10: INPUT "TOTAL # OF SAMPLES ?=";N$
26 N = VAL (N$):T = N - 64:T$ = STR$(T)
27 PRINT D$;"OPEN ORNEK,D2"
28 PRINT D$;"DELETE ORNEK"
29 PRINT D$;"OPEN ORNEK,D2"
30 PRINT D$;"WRITE ORNEK"
32 PRINT T$
35 PRINT D$;"CLOSE ORNEK"
38 DIM Y(N)
40 DIM ARALIK(N)
45 DIM WN(N)
60 PRINT
80 INPUT "A1=? (0<A1<1)= .";K$
85 PRINT : INPUT "A2=? (0<A2<1)= ";S$
90 A1 = VAL (K$):A2 = VAL (S$)
95 PRINT D$;"PR#1": PRINT : PRINT "A1="A1: PRINT : PRINT "A2="A2: PRINT : PRINT "TOT.# OF SAMPLES=";N - 64
96 PRINT D$;"PR#0"
100 PRINT : INPUT "AMPLITUDE ?=";Y$
105 PRINT : INPUT "MUL.VALUE OF X-COORD. =?";G$
107 CARP = VAL (G$)
110 D = VAL (Y$)
130 SR = SQR (.12)
140 FOR I = 1 TO N
150 RAN = 0
160 FOR L = 1 TO 12
170 RAN = RAN + ( RND (I) - .5)
180 NEXT L
190 WN(I) = (D / SR) * RAN
200 NEXT I
210 Y(1) = WN(1) - 5
215 Y(2) = WN(2) - 5.
220 FOR I = 3 TO N
230 Y(I) = Y(I - 1) * A1 + Y(I - 2) * A2 + WN(I)
240 NEXT
250 PRINT : PRINT : PRINT "X(I)";: HTAB 20: PRINT "Y(I)"
260 PRINT "----";: HTAB 20: PRINT "----"
270 FOR I = 1 TO N
280 PRINT I;: HTAB 20: PRINT Y(I): NEXT I
290 PRINT D$;"OPEN GIRDI,D2"
300 PRINT D$;"DELETE GIRDI"
310 PRINT D$;"OPEN GIRDI,L11,D2"
312 FOR I = 65 TO N
314 INP$ = STR$(Y(I))
316 IF LEN (INP$) = 10 THEN GOTO 330
318 IF ( LEN (INP$) > 10 AND VAL (INP$) < .01 AND VAL (INP$) > 0) THEN INP$ = ".001000000": GOTO 330
320 IF LEN (INP$) > 10 THEN INP$ = LEFT$(INP$,10): GOTO 330
322 FOR L = LEN (INP$) TO 10:INP$ = INP$ + " ": NEXT L

```

```
330 PRINT D$;"WRITE GIRDI,R";I - 64
360 PRINT INP$
370 NEXT I
380 PRINT D$;"CLOSE GIRDI"
400 HOME : VTAB 22
410 HGR
420 HPLOT 0,100 TO 270,100
430 HPLOT 0,0 TO 0,190
440 FOR I = 65 TO N
450 XI = (I - 64) * CARP:YI = Y(I)
460 HPLOT TO 0 + XI,100 - YI
470 NEXT
480 PRINT "AUTO REGG.WAVEFORM"
490 PRINT D$;"PR#1": PRINT CHR$(9);"G": PRINT D$;"PR#0"
510 FOR I = 1 TO N - 64
520 ARALIK(I) = I: NEXT
530 PRINT D$;"OPEN ARALIK,D2"
540 PRINT D$;"DELETE ARALIK"
550 PRINT D$;"OPEN ARALIK,L11,D2"
560 FOR I = 1 TO N - 64
570 INP$ = STR$(ARALIK(I))
610 FOR L = LEN(INP$) TO 10:INP$ = INP$ + " ": NEXT L
700 PRINT D$;"WRITE ARALIK,R";I
740 PRINT INP$
750 NEXT I
760 PRINT D$;"CLOSE ARALIK"
770 PRINT D$;"OPEN IST,D2"
780 PRINT D$;"DELETE IST"
790 PRINT D$;"OPEN IST,L3,D2"
800 FOR I = 65 TO N
810 PRINT D$;"WRITE IST,R";I - 64
820 IF Y(I) < 0 THEN VAR$ = "-1": GOTO 840
830 VAR$ = "+1"
840 PRINT VAR$: NEXT I
850 PRINT D$;"CLOSE IST"
1000 TEXT : HOME
6000 PRINT D$;"RUN QUANPROG,D1"
```

```

10 REM    ** QUANPROG **
20 TEXT : HOME
22 DIM X(32)
23 DIM QX(32)
30 D$ = CHR$(4)
90 PRINT : INPUT "# OF QUANTIZER LEVELS ?=";Q$
100 PRINT : INPUT "# OF SUBBLOCK SAMPLES ?=";Q$
101 PRINT : INPUT "MULT.VALUE OF X-COORD. ?=";G$
102 CARP = VAL(G$)
105 Q = VAL(Q$);N = VAL(Q$)
106 M$ = Q$:M$ = Q$
110 VTAB 14: HTAB 5: PRINT "1-UNIF.QUANTIZATION"
120 HTAB 5: PRINT "2-NONUNIF.OPT. QUANTIZATION"
130 HTAB 5: PRINT "3-DIFF.QUANT.WITH SUMMER"
140 HTAB 5: PRINT "4-LOGARITHMIC QUANTIZATION"
150 HTAB 5: PRINT "5-FEEDFORWARD ADAP.QUANT."
160 HTAB 5: PRINT "6-DIFF.PULSE CODE MODULATION (DPCM)"
170 PRINT : HTAB 5: INPUT "CHOOSE ONE OF THEM...";U$
190 U = VAL(U$)
200 IF U = 1 THEN M$ = "UNIFORM": GOTO 420
210 IF U = 2 THEN M$ = "NONUNIF OPT.": GOTO 420
220 IF U = 3 THEN M$ = "DIFF. (SUMMER)": GOTO 420
230 IF U = 4 THEN M$ = "LOGARITHMIC": GOTO 420
240 IF U = 5 THEN M$ = "FFADAPTIVE": GOTO 420
250 IF U = 6 THEN M$ = "DPCM"
260 IF U > 6 THEN PRINT CHR$(7): GOTO 110
420 IF U < > 1 THEN GOTO 520
430 HOME : HTAB 5: VTAB 5: PRINT "1-UNIFORM-MIDRISER VALUES"
440 HTAB 5: PRINT "2-UNIFORM-MIDTREAD VALUES"
450 HTAB 5: VTAB 10: PRINT "CHOOSE ONE OF THEM";: HTAB 24: INPUT X$
460 VTAB 10: HTAB 24: PRINT X$:X = VAL(X$)
470 IF X = 1 THEN Q$ = "UNORN" + Q$:M$ = "UNQUAN" + M$: GOTO 490
480 Q$ = "UNTREDORN" + Q$:M$ = "UNTREDQUAN" + M$
490 IF X = 1 THEN R$ = "UNIFORM-MIDRISER": GOTO 510
500 IF X = 2 THEN R$ = "UNIFORM-MIDTREAD"
510 GOTO 650
520 HOME : VTAB 8: HTAB 5: PRINT "1-GAMMA DENSITIES"
530 HTAB 5: PRINT "2-LAPLACE VALUES"
540 HTAB 5: PRINT "3-GAUSS VALUES"
550 IF U = 2 THEN GOTO 580
560 HTAB 5: PRINT "4-UNIFORM-MIDRISER VALUES"
570 HTAB 5: PRINT "5-UNIFORM-MIDTREAD VALUES"
580 VTAB 14: HTAB 5: PRINT "CHOOSE ONE OF THEM";: HTAB 24: INPUT Y$
590 VTAB 14: HTAB 24: PRINT Y$:Y = VAL(Y$)
600 IF Y > 5 THEN PRINT CHR$(7): GOTO 520
610 IF Y > = 4 THEN X = Y - 3: GOTO 470
620 IF Y = 1 THEN Q$ = "ORNEK" + Q$:M$ = "QUANT" + M$: GOTO 650
630 IF Y = 2 THEN Q$ = "LAPORN" + Q$:M$ = "LABUAN" + M$: GOTO 650
640 IF Y = 3 THEN Q$ = "GAUORN" + Q$:M$ = "GAQUAN" + M$: GOTO 650
650 HOME
660 IF Y = 1 THEN R$ = "GAMMA": GOTO 690
670 IF Y = 2 THEN R$ = "LAPLACE": GOTO 690
680 IF Y = 3 THEN R$ = "GAUSS"
690 PRINT D$;"PR#1": PRINT CHR$(19);"1"

```

```

200 PRINT TAB( 9)"QUANTIZATION OF"
210 PRINT TAB( 5)0$;: PRINT TAB( 9)"SAMPLES " + V$ + " SIGNAL WITH"
230 PRINT TAB( 5)M$;: PRINT TAB( 9)"LEVELS " + W$ + " QUANT.USING "
240 PRINT TAB( 9)R$ + " DENSITIES"
250 PRINT D$;"PR#0"
2050 HOME
2060 PRINT D$;"OPEN";0$;" ,D2"
2070 PRINT D$;"READ";0$
2080 INPUT X1$,X2$,X3$,X4$,X5$,X6$,X7$
2090 INPUT X8$,X9$,XA$,XB$,XC$,XD$,XE$
2100 INPUT XF$,XG$,XH$,XI$,XJ$,XK$,XL$,XM$,XN$
2110 INPUT XO$,XP$,XQ$,XR$,XS$,XT$,XU$,XV$,XW$
2120 PRINT D$;"CLOSE";0$
2130 PRINT D$;"OPEN";N$;" ,D2"
2140 PRINT D$;"READ";N$
2150 INPUT Q1$,Q2$,Q3$,Q4$,Q5$,Q6$
2160 INPUT Q7$,Q8$,Q9$,QA$,QB$,QC$
2170 INPUT QD$,QE$,QF$,QG$,QH$,QI$,QJ$,QK$,QL$,QM$
2180 INPUT QN$,QO$,QP$,QQ$,QR$,QS$,QT$,QU$,QV$,QW$
2190 PRINT D$;"CLOSE";N$
2200 HOME
2220 PRINT : PRINT : PRINT "X1$ VALUES";: HTAB 20: PRINT "Q1$ VALUES": PRINT "-----";: HTAB 20: PRINT "-----"
2230 PRINT X1$;: HTAB 20: PRINT Q1$: PRINT X2$;: HTAB 20: PRINT Q2$: PRINT X3$;: HTAB 20: PRINT Q3$: PRINT X4$;: HTAB 20: PRINT Q4$: PRINT X5$
;: HTAB 20: PRINT Q5$
2240 PRINT X6$;: HTAB 20: PRINT Q6$: PRINT X7$;: HTAB 20: PRINT Q7$: PRINT X8$;: HTAB 20: PRINT Q8$
2250 PRINT X9$;: HTAB 20: PRINT Q9$
2260 PRINT XA$;: HTAB 20: PRINT QA$: PRINT XB$;: HTAB 20: PRINT QB$: PRINT XC$;: HTAB 20: PRINT QC$: PRINT XD$;: HTAB 20: PRINT QD$: PRINT XE$
;: HTAB 20: PRINT QE$
2270 PRINT XF$;: HTAB 20: PRINT QF$: PRINT XG$;: HTAB 20: PRINT QG$
2280 PRINT XH$;: HTAB 20: PRINT QH$: PRINT XI$;: HTAB 20: PRINT QI$
2290 PRINT XJ$;: HTAB 20: PRINT QJ$: PRINT XK$;: HTAB 20: PRINT QK$
2300 PRINT XL$;: HTAB 20: PRINT QL$: PRINT XM$;: HTAB 20: PRINT QM$: PRINT XN$;: HTAB 20: PRINT QN$
2310 PRINT XO$;: HTAB 20: PRINT QO$: PRINT XP$;: HTAB 20: PRINT QP$: PRINT XQ$;: HTAB 20: PRINT QQ$
2320 PRINT XR$;: HTAB 20: PRINT QR$: PRINT XS$;: HTAB 20: PRINT QS$: PRINT XT$;: HTAB 20: PRINT QT$
2330 PRINT XU$;: HTAB 20: PRINT QU$: PRINT XV$;: HTAB 20: PRINT QV$: PRINT XW$;: HTAB 20: PRINT QW$
2340 X(1) = VAL (X1$):X(2) = VAL (X2$)
2350 X(3) = VAL (X3$):X(4) = VAL (X4$)
2360 X(5) = VAL (X5$):X(6) = VAL (X6$)
2370 X(7) = VAL (X7$):X(8) = VAL (X8$)
2380 IF N = 4 THEN GOTO 2520
2390 X(9) = VAL (X9$):X(10) = VAL (XA$)
2400 X(11) = VAL (XB$):X(12) = VAL (XC$)
2410 X(13) = VAL (XD$):X(14) = VAL (XE$)
2420 X(15) = VAL (XE$):X(16) = VAL (XG$)
2430 IF N = 8 THEN GOTO 2520
2440 X(17) = VAL (XH$):X(18) = VAL (XI$)
2450 X(19) = VAL (XJ$):X(20) = VAL (XK$)
2460 X(21) = VAL (XL$):X(22) = VAL (XM$)
2470 X(23) = VAL (XM$):X(24) = VAL (XO$)
2480 X(25) = VAL (XP$):X(26) = VAL (XQ$)
2490 X(27) = VAL (XR$):X(28) = VAL (XS$)
2500 X(29) = VAL (XT$):X(30) = VAL (XU$)
2510 X(31) = VAL (XV$):X(32) = VAL (XW$)
2520 QX(1) = VAL (Q1$):QX(2) = VAL (Q2$)
2530 QX(3) = VAL (Q3$):QX(4) = VAL (Q4$)
2540 QX(5) = VAL (Q5$):QX(6) = VAL (Q6$)

```

```

2550 QX(7) = VAL (Q7$):QX(8) = VAL (Q8$)
2560 IF N = 4 THEN GOTO 2700
2570 QX(9) = VAL (Q9$):QX(10) = VAL (Q10$)
2580 QX(11) = VAL (Q11$):QX(12) = VAL (Q12$)
2590 QX(13) = VAL (Q13$):QX(14) = VAL (Q14$)
2600 QX(15) = VAL (Q15$):QX(16) = VAL (Q16$)
2610 IF N = 8 THEN GOTO 2700
2620 QX(17) = VAL (Q17$):QX(18) = VAL (Q18$)
2630 QX(19) = VAL (Q19$):QX(20) = VAL (Q20$)
2640 QX(21) = VAL (Q21$):QX(22) = VAL (Q22$)
2650 QX(23) = VAL (Q23$):QX(24) = VAL (Q24$)
2660 QX(25) = VAL (Q25$):QX(26) = VAL (Q26$)
2670 QX(27) = VAL (Q27$):QX(28) = VAL (Q28$)
2680 QX(29) = VAL (Q29$):QX(30) = VAL (Q30$)
2690 QX(31) = VAL (Q31$):QX(32) = VAL (Q32$)
2700 HOME
2710 VTAB 15: PRINT "PLEASE WAIT..."
2720 IF U = 3 THEN GOTO 2760
2725 IF U = 5 THEN GOTO 3000
2730 IF U = 6 THEN GOTO 2780
2740 PRINT D$;"BLOAD CHAIN,A520,D1"
2750 CALL 520"Q.SNRPR06"
2760 PRINT D$;"BLOAD CHAIN,A520,D1"
2770 CALL 520"DIFADAP0U"
2780 PRINT D$;"BLOAD CHAIN,A520,D1"
2790 CALL 520"DPCH"
3000 PRINT D$;"BLOAD CHAIN,A520,D1"
3010 CALL 520"FFADAP0U"

```



```

10 D$ = CHR$(4)
20 REM ** QUANTIZAT.,ERR & SNR CALCUL.**
30 DIM Y(N)
35 DIM QY(N)
37 DIM ARALIK(N)
40 DIM ERR(N)
43 C = 1:CC = N
45 PRINT D$;"OPEN DRNEK,D2"
50 PRINT D$;"READ DRNEK"
55 INPUT NN$
60 PRINT D$;"CLOSE DRNEK"
62 NN = VAL (NN$):HH = NN / M
65 PRINT D$;"OPEN GIRDI,L11,D2"
67 FOR I = C TO CC:P = I - C + 1
70 PRINT D$;"READ GIRDI,R";I
72 INPUT INP$:Y(P) = VAL (INP$)
73 NEXT I
74 PRINT D$;"CLOSE GIRDI"
75 PRINT D$;"OPEN ARALIK,L11,D2"
77 FOR I = C TO CC:P = I - C + 1
80 PRINT D$;"READ ARALIK,R";I
82 INPUT INP$:ARALIK(P) = VAL (INP$)
83 NEXT I
84 PRINT D$;"CLOSE ARALIK"
100 IF U = 4 THEN GOTO 3000
168 REM ** SIGMA HESAPLANIYOR **
170 SIG = 0
176 FOR I = 1 TO N
178 SIG = SIG + Y(I) * Y(I)
180 NEXT I
182 SIG = SIG / N:SUM = SIG
184 SIG = SQR (SIG)
186 PRINT D$;"PR# 1"
187 PRINT : PRINT "SIG=";SIG
188 PRINT D$;"PR#0"
189 PRINT : INPUT "DO YOU WANT ADAPT.VIA INP.VARIANCE=(Y/N)";T$
190 IF T$ = "Y" THEN GOTO 193
191 IF T$ < > "N" THEN PRINT CHR$(7): GOTO 189
192 SIG = 1
193 FOR I = 1 TO 32
196 X(I) = X(I) * SIG
198 QX(I) = QX(I) * SIG
200 NEXT I
210 FOR M = 1 TO N
220 FOR I = 1 TO 32
230 IF ABS (Y(M)) > X(I) THEN GOTO 260
240 IF (U = 4 AND Y(M) < 0) THEN QY(M) = - QX(I): GOTO 270
250 QY(M) = QX(I): GOTO 270
260 NEXT I
270 NEXT M
280 IF U = 4 THEN GOTO 3180
290 HOME
300 PRINT D$
310 PRINT D$;"OPEN IST,L3,D2"
320 FOR I = C TO CC:P = I - C + 1

```

```

330 PRINT D$;"READ IST,R";I
340 INPUT IST$:IMG = VAL (IST$)
350 QY(P) = QY(P) & IMG
360 IF U = 4 THEN Y(P) = Y(P) & IMG
370 NEXT I
380 PRINT D$;"CLOSE IST"
390 PRINT D$;"PR#1"
400 PRINT : PRINT "QUANT.VALUES";: HTAB 20: PRINT "ORG.VALUES": PRINT
410 FOR I = 1 TO M
420 PRINT QY(I);: HTAB 20: PRINT Y(I): NEXT
430 PRINT D$;"PR#0"
440 PRINT
450 HOME : VTAB 22
460 HGR : HPLOT 0,100 TO 270,100: HPLLOT 0,0 TO 0,190
470 FOR L = 1 TO M
480 PRINT D$
490 YI = QY(L).
500 XI = ARALIK(L) & CARP
510 HPLOT TO 0 + XI,100 - YI
520 NEXT L
530 PRINT "QUANTIZED SIGNAL"
550 PRINT D$;"PR#1": PRINT CHR$ (9);"G"
560 TEXT : HOME : PRINT "ERRORS!"
570 PRINT "-----"
600 FOR I = 1 TO M
610 ERR(I) = QY(I) - Y(I)
620 PRINT ERR(I)
630 NEXT
640 PRINT D$;"PR#0"
680 PRINT D$;"OPEN HATA,L11,D2"
690 FOR I = C TO CC:P = I - C + 1
700 ERR$ = STR$ (ERR(P))
710 IF LEN (ERR$) = 10 THEN GOTO 750
715 IF ( LEN (ERR$) > 10 AND VAL (ERR$) < .01 AND VAL (ERR$) > 0) THEN ERR$ = ".001000000": GOTO 750
720 IF LEN (ERR$) > 10 THEN ERR$ = LEFT$ (ERR$,10): GOTO 750
730 FOR L = LEN (ERR$) TO 10
740 ERR$ = ERR$ + " ": NEXT L
750 PRINT D$;"WRITE HATA,R";I
760 PRINT ERR$
770 NEXT I
780 PRINT D$;"CLOSE HATA"
900 HOME : VTAB 22: HGR : HPLOT 0,100 TO 270,100: HPLLOT 0,0 TO 0,190
910 FOR I = 1 TO M
920 YI = ERR(I)
930 XI = ARALIK(I) & CARP
940 HPLOT TO 0 + XI,100 - YI: NEXT I
950 PRINT "QUANTIZATION ERROR"
970 PRINT D$;"PR#1": PRINT CHR$ (9);"B": PRINT D$;"PR#0"
980 HOME : TEXT
990 TPL = 0
1000 FOR I = 1 TO M
1010 TPL = TPL + ERR(I) & ERR(I)

```

```

1020 NEXT :TPL = TPL / N
1030 IF U = 4 THEN GOTO 3100
1040 SNR = SUM / TPL
1050 PRINT "SUM=";SUM: PRINT "TPL=";TPL
1060 SNR = 10 * LOG (SNR) / 2.3: PRINT "SNR=";SNR
1070 HOME : VTAB 5: PRINT D$;"PR#1": PRINT CHR$ (9);"I": PRINT "SNR=";SNR: PRINT D$;"PR#0"
1080 IST$ = ""
1090 GOTO 10000
3000 REM ** LOGARITMIK QUANTIZATION.**
3010 HOME : PRINT "LOG VALUES OF Y(I)"
3020 PRINT "-----"
3030 FOR I = 1 TO N
3040 Y(I) = ABS (Y(I))
3050 IF Y(I) = 0 THEN Y(I) = 0.00001
3060 Y(I) = LOG (Y(I)) / 2.3
3070 PRINT Y(I)
3080 NEXT
3090 GOTO 210
3100 REM
3110 SIG = 0
3120 FOR I = 1 TO N
3130 SIG = SIG + Y(I) * Y(I): NEXT
3140 SUM = SIG / N: GOTO 1040
3180 FOR I = 1 TO N
3190 Y(I) = 2.71838 ^ (Y(I) * 2.3)
3200 QY(I) = 2.71838 ^ (QY(I) * 2.3)
3210 NEXT
3220 GOTO 290
10000 REM SURECIN DEVANI DEVREYE GIRIYOR
10020 IF CC = NN THEN GOTO 15010
10030 C = C + N:CC = CC + N
10032 FOR I = 1 TO N
10035 X(I) = X(I) / SIG:QX(I) = QY(I) / SIG: NEXT
10045 GOTO 65
15010 HOME : VTAB 15: INPUT "DO YOU WANT AUTOCORR.COEFF. ? (Y/N)";S$
15020 IF S$ = "N" THEN GOTO 15050
15030 IF S$ < -> "Y" THEN GOTO 15010
15040 PRINT D$;"RUN OZILINTI,S6,D1"
15050 PRINT D$;"RUN FFTPROG,S6,D1"

```

```

5 D$ = CHR$(4)
10 REM ** DIFF. QUAN. S/R **
15 DIM Y(N)
20 DIM ERR(N)
25 DIM QY(N)
30 DIM USE(N + 1)
32 DIM ARALIK(N)
35 C = 1:CC = N
37 PRINT D$;"OPEN ORNEK,D2"
40 PRINT D$;"READ ORNEK"
42 INPUT NN$
44 PRINT D$;"CLOSE ORNEK"
45 NN = VAL (NN$):HH = NN / N
47 PRINT D$;"OPEN GIRD1,L11,D2"
49 FOR I = C TO CC:P = I - C + 1
50 PRINT D$;"READ GIRD1,R";I
52 INPUT INP$:Y(P) = VAL (INP$)
53 NEXT I
54 PRINT D$;"CLOSE GIRD1"
56 PRINT D$;"OPEN ARALIK,L11,D2"
58 FOR I = C TO CC:P = I - C + 1
60 PRINT D$;"READ ARALIK,R";I
62 INPUT INP$:ARALIK(P) = VAL (INP$)
63 NEXT I
64 PRINT D$;"CLOSE ARALIK"
65 REM ** SIGMA IS BEING CALCULATED **
67 SIG = 0
68 FOR I = 1 TO N
70 SIG = SIG + Y(I) * Y(I)
80 NEXT
90 SIG = SIG / N:SUM = SIG
100 SIG = SQR (SIG)
110 PRINT D$;"PR# 1"
120 PRINT : PRINT "SIG=";SIG
130 PRINT D$;"PR#0"
140 RO = SUM:R1 = 0
150 FOR I = 1 TO N - 1
160 R1 = Y(I) * Y(I + 1) + R1
170 NEXT I
180 R1 = R1 / N
190 HOME
200 PRINT D$;" PR#1": VTAB 10: PRINT "RO=";RO: PRINT "R1=";R1
210 ALF = R1 / RO: PRINT "ALF=";ALF
230 FOR I = 1 TO N - 1
240 ERR(I) = Y(I + 1) - ALF * Y(I)
260 NEXT
270 RS = 0: FOR I = 1 TO N
280 RS = RS + ERR(I) * ERR(I): NEXT
290 RS = RS / (N - 1):RS = SQR (RS): PRINT : PRINT "RS=";RS: PRINT
330 PRINT D$;"PR#0"
2000 REM ** DIFF.QUAN. **

```

```

2010 USE(I) = 0
2020 FOR M = 1 TO N
2030 FOR I = 1 TO 32
2040 ERR(M) = Y(M) - USE(M):ZNM = 1
2050 IF ERR(M) < 0 THEN ZNM = - 1
2060 IF ABS (ERR(M)) > X(I) THEN GOTO 2200
2070 QY(M) = QX(I) * ZNM
2080 USE(M + 1) = USE(M) + QY(M): GOTO 2210
2200 NEXT I
2210 NEXT M
2211 FOR I = 1 TO N:QY(I) = ERR(I) - QY(I): NEXT
2213 SUM = 0:TPL = 0
2214 FOR I = 1 TO N
2215 SUM = SUM + ERR(I) * ERR(I):TPL = TPL + QY(I) * QY(I)
2216 NEXT
2217 SPR = 10 * LOG (SUM / TPL) / 2.3
2219 PR# 1
2220 PRINT : PRINT "Y(M)";: HTAB 15: PRINT "USE(M)";: HTAB 27: PRINT "QY(M)"
2230 FOR I = 1 TO N: PRINT Y(I);: HTAB 15: PRINT USE(I);: HTAB 27: PRINT QY(I)
2240 NEXT
2250 SUM = 0:TPL = 0
2260 FOR I = 1 TO N
2265 ERR(I) = Y(I) - USE(I + 1)
2270 SUM = SUM + Y(I) * Y(I)
2280 TPL = TPL + ERR(I) * ERR(I)
2290 NEXT
2300 SUM = SUM / N:TPL = TPL / N
2310 SNR = SUM / TPL:SNR = 10 * LOG (SNR) / 2.3
2330 PRINT "SNR-QUAN+PRED=";SNR: PRINT "SNR-QUAN    " =";SPR
2335 PRINT D$;"PR#0"
2340 HOME : VTAB 22
2350 HGR : HPLLOT 0,100 TO 270,100: HPLLOT 0,0 TO 0,190
2360 FOR L = 1 TO N
2370 YI = USE(L):XI = ARALIK(L) * CARP * HH
2390 HPLLOT TO 0 + XI,100 - YI
2400 NEXT L
2410 PRINT "QUANTIZED SIGNAL"
2420 INPUT S$
2430 PRINT D$;"PR#1": PRINT CHR$ (9);"G": PRINT D$;"PR#0": INPUT S$
2435 HOME : VTAB 22
2440 HGR : HPLLOT 0,100 TO 270,100: HPLLOT 0,0 TO 0,190
2450 FOR L = 1 TO N
2460 YI = ERR(L):XI = ARALIK(L) * CARP * HH
2470 HPLLOT TO 0 + XI,100 - YI
2480 NEXT L
2500 PRINT "ERROR(Y(M)-USE(M))"
2501 PRINT D$;"PR#1": PRINT CHR$ (9);"G": PRINT D$;"PR#0"
2502 TEXT : HOME
2504 PRINT D$;"PR#1": PRINT "ERR(M)"
2505 FOR I = 1 TO N
2506 PRINT ERR(I): NEXT
2509 PRINT D$;"PR#0"
3030 PRINT D$;"OPEN HATA,L11,D2"
3040 FOR I = C TO CC:P = I - C + 1
3050 ERR$ = STR$ (ERR(P))
3060 IF LEN (ERR$) = 10 THEN GOTO 3090
3065 IF ( LEN (ERR$) > 10 AND VAL (ERR$) < 0) THEN ERR$ = LEFT$ (ERR$,10): GOTO 3090
3067 IF ( LEN (ERR$) > 10 AND VAL (ERR$) < .01) THEN ERR$ = ".001000000": GOTO 3090

```

```

3070 FOR L = LEN (ERR$) TO 10
3080 ERR$ = ERR$ + " ": NEXT L
3090 PRINT D$;"WRITE HATA,R";I
3100 PRINT ERR$
3120 NEXT I
3130 PRINT D$;"CLOSE HATA"
4000 REM SURECIN DEVAMI DEVREYE GIRIYOR
4010 IF CC = NN THEN GOTO 10010
4020 C = C + N:CC = CC + N
4032 FOR I = 1 TO N
4035 X(I) = X(I) / RS:QX(I) = QX(I) / RS: NEXT
4050 GOTO 47
10010 VTAB 15: INPUT "DO YOU WANT AUT.CORRELATION COEFF.? (Y/N)";S$
10013 IF S$ = "N" THEN GOTO 10020
10014 IF S$ < > "Y" THEN GOTO 10010
10015 PRINT D$;"RUN OZILINTI,S6,D1"
10020 PRINT D$;"RUN FFTPROG,S6,D1"

```

```

10 D$ = CHR$(4)
20 REM ## FEEDFORWARD ADAP.QUANT. ##
30 DIM Y(N)
35 DIM QY(N)
37 DIM ARALIK(N)
40 DIM ERR(N)
43 C = 1:CC = N
45 PRINT D$;"OPEN ORNEK,D2"
50 PRINT D$;"READ ORNEK"
55 INPUT NN$
60 PRINT D$;"CLOSE ORNEK"
62 NN = VAL (NN$):HH = NN / N
65 PRINT D$;"OPEN GIRDI,L11,D2"
67 FOR I = C TO CC:P = I - C + 1
70 PRINT D$;"READ GIRDI,R";I
72 INPUT INP$:Y(P) = VAL (INP$)
73 NEXT I
74 PRINT D$;"CLOSE GIRDI"
75 PRINT D$;"OPEN ARALIK,L11,D2"
77 FOR I = C TO CC:P = I - C + 1
80 PRINT D$;"READ ARALIK,R";I
82 INPUT INP$:ARALIK(P) = VAL (INP$)
83 NEXT I
84 PRINT D$;"CLOSE ARALIK"
90 HOME : VTAB (0: INPUT "CHOOSE M VALUE AS WINDOW= ";MI
168 REM ## SIGMA HESAPLANIYOR ##
170 SIG = 0.
176 FOR I = 1 TO N
178 SIG = SIG + Y(I) * Y(I)
180 NEXT I
181 SUM = SIG / N
182 SIG = SQR (SUM)
184 FOR I = 1 TO N
186 SJG = 0
188 FOR M = I TO I + MI - 1
190 IF M > N THEN GOTO 199
192 SJG = SJG + Y(M) * Y(M)
193 NEXT M
196 SJG = SQR (SJG / MI)
198 IF M < N THEN GOTO 200
199 SJG = SIG
200 FOR M = 1 TO 32
202 X(M) = Y(N) * SJG:QX(M) = QX(M) * SJG
206 NEXT M
220 FOR M = 1 TO 32
230 IF ABS (Y(I)) > X(M) THEN GOTO 260
250 QY(I) = QX(M): GOTO 262
260 NEXT M
262 FOR M = 1 TO 32
264 X(M) = X(M) / SJG:QX(M) = QX(M) / SJG
266 NEXT M
270 NEXT I
290 HOME
300 PRINT D$
310 PRINT D$;"OPEN IST,L3,D2"
320 FOR I = C TO CC:P = I - C + 1
330 PRINT D$;"READ IST,R";I
340 INPUT IST$:IMG = VAL (IST$)

```

```

350 QY(P) = QY(P) * IMG
370 NEXT I
380 PRINT D$;"CLOSE IST"
390 PRINT D$;"PR#1"
395 PRINT : PRINT "WINDOW-WI=";WI
400 PRINT : PRINT "QUANT.VALUES";: HTAB 20: PRINT "ORG.VALUES": PRINT
410 FOR I = 1 TO N
420 PRINT QY(I);: HTAB 20: PRINT Y(I): NEXT
430 PRINT D$;"PR#0"
440 PRINT
450 HOME : VTAB 22
460 HGR : HPLOT 0,100 TO 270,100: HPLOT 0,0 TO 0,190
470 FOR L = 1 TO N
480 PRINT D$
490 YI = QY(L)
500 XI = ARALIK(L) * CARP
510 HPLOT TO 0 + XI,100 - YI
520 NEXT L
530 PRINT "QUANTIZED SIGNAL"
550 PRINT D$;"PR#1": PRINT CHR$(9);"B"
560 TEXT : HOME : PRINT "ERRORS!"
570 PRINT "-----"
600 FOR I = 1 TO N
610 ERR(I) = QY(I) - Y(I)
620 PRINT ERR(I)
630 NEXT
640 PRINT D$;"PR#0"
680 PRINT D$;"OPEN HATA,L11,D2"
690 FOR I = C TO CC:P = I - C + 1
700 ERR$ = STR$(ERR(P))
710 IF LEN (ERR$) = 10 THEN GOTO 750
715 IF ( LEN (ERR$) > 10 AND VAL (ERR$) < .01 AND VAL (ERR$) > 0) THEN ERR$ = ".001000000": GOTO 750
720 IF LEN (ERR$) > 10 THEN ERR$ = LEFT$(ERR$,10): GOTO 750
730 FOR L = LEN (ERR$) TO 10
740 ERR$ = ERR$ + " ": NEXT L
750 PRINT D$;"WRITE HATA,R";I
760 PRINT ERR$
770 NEXT I
780 PRINT D$;"CLOSE HATA"
900 HOME : VTAB 22: HGR : HPLOT 0,100 TO 270,100: HPLOT 0,0 TO 0,190
910 FOR I = 1 TO N
920 YI = ERR(I)
930 XI = ARALIK(I) * CARP
940 HPLOT TO 0 + XI,100 - YI: NEXT I
950 PRINT "QUANTIZATION ERROR"
970 PRINT D$;"PR#1": PRINT CHR$(9);"B": PRINT D$;"PR#0"
980 HOME : TEXT

```



```
990 TPL = 0
1000 FOR I = 1 TO M
1010 TPL = TPL + ERR(I) * ERR(I)
1015 NEXT I
1020 TPL = TPL / M
1040 SNR = SUM / TPL
1050 PRINT "SUM=";SUM: PRINT "TPL=";TPL
1060 SNR = 10 * LOG (SNR) / 2.3: PRINT "SNR=";SNR
1070 HOME : VTAB 5: PRINT D$;"PR#1": PRINT CHR$(9);"I": PRINT "SNR=";SNR: PRINT D$;"PR#0"
1080 IST$ = ""
1090 GOTO 10000
10000 REM SURECIN DEVAMI DEVREYE GIRIYOR
10020 IF CC = NN THEN GOTO 15010
10030 C = C + N:CC = CC + N
10032 FOR I = 1 TO N
10035 X(I) = X(I) / SJG:QX(I) = QX(I) / SJG: NEXT
10045 GOTO 65
15010 HOME : VTAB 15: INPUT "DO YOU WANT AUTOCORR.COEFF. ? (Y/N)";S$
15020 IF S$ = "N" THEN GOTO 15050
15030 IF S$ < > "Y" THEN GOTO 15010
15040 PRINT D$;"RUN OZILINTI,S6,D1"
15050 PRINT D$;"RUN FFTPROG,S6,D1"
```

```

5 REM ** DPCM **
10 D$ = CHR$(4)
15 TEXT : HOME : PRINT "DIFF.PULSE CODE MOD."
16 PRINT D$;"OPEN ORNEK,D2"
17 PRINT D$;"READ ORNEK"
18 INPUT NN$
19 PRINT D$;"CLOSE ORNEK"
20 NN = VAL (NN$):HH = NN / M
22 VTAB 10: PRINT "TOT.# OF SAMPLES =" ; NN$
24 PRINT : PRINT "# OF SUBBLOCK SAMPLES =" ; 0$
25 PRINT : INPUT "# OF COEFFICIENTS =" ; R$
30 R = VAL (R$)
35 PRINT D$;"PR#1": PRINT : PRINT "# OF SUBBLOCK SAMPLES=" ; 0$ : PRINT "TOT.# OF SAMPLES=" ; NN$ : PRINT "# OF COEFF.=" ; R$ : PRINT D$;"PR#0"
40 DIM Y(N)
45 DIM PRE(N)
50 DIM QY(N)
55 DIM ERR(N)
60 DIM PD(R)
63 DIM R(R)
65 DIM C(R,R)
67 DIM B(R,R)
68 C = 1:CC = N
70 PRINT D$;"OPEN GIRDI,L11,D2"
72 FOR I = C TO CC:P = I - C + 1
74 PRINT D$;"READ GIRDI,R";I
76 INPUT INP$:Y(P) = VAL (INP$)
78 NEXT I
80 PRINT D$;"CLOSE GIRDI"
105 REM -- ONGORUCU KATSAYILARI HESAPLANIYOR --
106 VAR = 0
107 FOR I = 1 TO N
109 VAR = VAR + Y(I) * Y(I): NEXT
110 VAR = VAR / N
120 FOR K = 1 TO R
130 R(K) = 0
140 FOR I = 1 TO N - K
150 R(K) = R(K) + Y(I) * Y(I + K)
160 NEXT I
170 R(K) = R(K) / (N - K)
180 NEXT K
190 HOME : PRINT D$;"PR#1"
200 PRINT : PRINT "AUTOCORRELATION COEFF."
205 PRINT "-----"
210 PRINT : PRINT "R(0)=" ; VAR
220 FOR I = 1 TO R
230 PRINT "R(" ; I ")=" ; R(I): NEXT
250 FOR I = 1 TO R:R(I) = R(I) / VAR: NEXT
260 FOR I = 1 TO R - 1
270 FOR L = 1 TO R - I
280 C(L,L) = 1
290 IF I + L > R THEN GOTO 320
300 C(L + I,L) = R(I):C(L,L + I) = R(I)
310 NEXT L
320 NEXT I
340 PRINT : PRINT "AUTOCORRELATION MATRIX"
345 PRINT "-----": PRINT

```

```

350 FOR I = 1 TO R
360 FOR L = 1 TO R
365 PRINT "C(";I;";";L;")=";C(I,L)
370 NEXT L
380 NEXT I
390 PRINT D$;"PR#0"
400 GOSUB 1000: REM MATX.INVERSION S/R
410 REM $$ REM MATX.MULTP $$
415 PRINT D$;"PR#1"
420 PRINT : PRINT "PREDICTOR COEFF."
430 PRINT "-----"
440 FOR I = 1 TO R
450 PD(I) = 0
460 FOR K = 1 TO R
470 PD(I) = PD(I) + B(I,K) * R(K)
480 NEXT K
500 PRINT "PD(";I;")=";PD(I)
502 NEXT I
504 PRINT D$;"PR#0"
505 REM --NICELEYICI FFADAP OLUYOR--
506 FOR I = 1 TO N - R
507 ERR(I) = Y(I + R)
508 FOR J = 1 TO R
509 ERR(I) = ERR(I) - R(J) * Y(I + R - J): NEXT J
510 NEXT I
512 RS = 0
513 FOR I = 1 TO N
514 RS = RS + ERR(I) * ERR(I)
515 NEXT I
516 RS = RS / (N - 1):RS = SQR (RS): PRINT : PRINT "RS=";RS
518 FOR I = 1 TO 32
520 X(I) = X(I) * RS:QX(I) = QX(I) * RS: NEXT I
522 REM --DPCM UYGULANIYOR--
525 HOME :PRE(1) = 0:ERR(1) = Y(1)
530 FOR I = 1 TO R
535 FOR M = 1 TO 32
540 ZMN = 1
542 IF ERR(1) < 0 THEN ZMN = - 1
544 IF ABS (ERR(1)) > X(M) THEN GOTO 549
546 QY(1) = QX(M) * ZMN: GOTO 550
549 NEXT M
550 FOR K = 1 TO I
560 PRE(I + 1) = PD(K) * PRE(I + 1 - K) + PRE(I + 1): NEXT K
580 PRE(I + 1) = PRE(I + 1) + QY(I)
585 ERR(I + 1) = Y(I + 1) - PRE(I + 1)
590 NEXT I
600 FOR I = R + 1 TO N - 1
603 FOR M = 1 TO 32
607 ZMN = 1
608 IF ERR(1) < 0 THEN ZMN = - 1
610 IF ABS (ERR(1)) > X(M) THEN GOTO 620
615 QY(1) = QX(M) * ZMN: GOTO 630
620 NEXT M
630 FOR K = 1 TO R
640 PRE(I + 1) = PD(K) * PRE(I + 1 - K) + PRE(I + 1): NEXT K
650 PRE(I + 1) = PRE(I + 1) + QY(I)
655 ERR(I + 1) = Y(I + 1) - PRE(I + 1)
660 NEXT I

```

```

662 HOME : VTAB 22: HGR
663 HPLLOT 0,100 TO 270,100: HPLLOT 0,0 TO 0,190
664 FOR I = 1 TO N
665 YI = PRE(I):XI = I * CARP * HH
667 HPLLOT TO 0 + XI,100 - YI: NEXT
668 PRINT "PRED.& QUANTIZED SIGNAL"
670 PRINT D$;"PR#1": PRINT CHR$(9);"6": PRINT D$;"PR#0"
672 TEXT : HOME
673 FOR I = 1 TO N
674 ERR(I) = Y(I) - PRE(I): NEXT
675 PRINT D$;"PR#1"
680 PRINT "ORG.SIGNAL";: HTAB 13: PRINT "PRED.SIGN.";: HTAB 26: PRINT "QY(I)"
690 PRINT "-----";: HTAB 13: PRINT "-----";: HTAB 26: PRINT "-----"
700 FOR I = 1 TO N
710 PRINT Y(I);: HTAB 13: PRINT PRE(I);: HTAB 26: PRINT QY(I): NEXT I
720 PRINT : PRINT "QUAN.ERR.";: HTAB 20: PRINT "PRED.ERR."
730 PRINT "-----";: HTAB 20: PRINT "-----"
740 FOR I = 1 TO N - 1
750 PRINT ERR(I) - QY(I);: HTAB 20: PRINT ERR(I): NEXT I
755 SQNR = 0:SUM = 0:TPL = 0
760 FOR I = 1 TO N - 1
765 SQNR = (ERR(I) - QY(I)) * (ERR(I) - QY(I)) + SQNR
775 SUM = Y(I) * Y(I) + SUM
777 TPL = ERR(I) * ERR(I) + TPL
780 NEXT I
800 SNR = SUM / SQNR:SNR = 10 * LOG (SNR) / 2.3
805 SQNR = TPL / SQNR
810 SPNR = SUM / TPL
815 SQNR = 10 * LOG (SQNR) / 2.3:SPNR = 10 * LOG (SPNR) / 2.3
817 PRINT : PRINT "SNR" =";SQNR
818 PRINT : PRINT "PREDICTOR GAIN" =";SPNR
819 PRINT : PRINT "SIGNAL TO QUANTIZATION NOISE RATIO=";SNR
820 PRINT D$;"PR#0"
823 PRINT D$;"OPEN HATA,,L11,D2"
825 FOR I = C TO CC - 1:P = I - C + 1
827 ERR$ = STR$(ERR(P) - QY(P))
830 IF LEN (ERR$) = 10 THEN GOTO 847
832 IF ( LEN (ERR$) > 10 AND VAL (ERR$) < .01 AND VAL (ERR$) > 0) THEN ERR$ = ".001000000": GOTO 847
834 IF LEN (ERR$) > 10 THEN ERR$ = LEFT$(ERR$,10): GOTO 847
836 FOR L = LEN (ERR$) TO 10
838 ERR$ = ERR$ + " ": NEXT L
847 PRINT D$;"WRITE HATA,R";I
849 PRINT ERR$
850 NEXT I
851 PRINT D$;"WRITE HATA,R";CC
852 PRINT STR$(0)
853 PRINT D$;"CLOSE HATA"
855 REM SURECIN DEVAMI DEVREYE GIRIYOR
858 IF CC = NN THEN GOTO 900
860 C = C + N:CC = CC + N
865 FOR I = 1 TO N
870 X(I) = X(I) / RS:QX(I) = QX(I) / RS
875 NEXT I
880 GOTO 70
900 HOME : VTAB 15: INPUT "DO YOU WANT AUTOCORR.COEFF.?(Y/N)";S$
910 IF S$ = "N" THEN PRINT D$;"RUN FFTPROG,D1"
920 IF S$ < > "Y" THEN GOTO 900
930 PRINT D$;"RUN OZILINTI,D1"
1000 REM ** MATX INVERSION S/R **
1070 PRINT "MATRIX ELEMENTS : "
1080 FOR J = 1 TO R

```

```

1070 PRINT "MATRIX ELEMENTS : "
1080 FOR J = 1 TO R
1130 B(J,J) = 1
1140 NEXT J
1150 FOR J = 1 TO R
1160 FOR I = J TO R
1170 IF C(I,J) < > 0 THEN 1210
1180 NEXT I
1190 PRINT : PRINT "SINGULAR MATRIX": PRINT
1200 END
1210 FOR K = 1 TO R
1220 S = C(J,K)
1230 C(J,K) = C(I,K)
1240 C(I,K) = S
1250 S = B(J,K)
1260 B(J,K) = B(I,K)
1270 B(I,K) = S
1280 NEXT K
1290 T = 1 / C(J,J)
1300 FOR K = 1 TO R
1310 C(J,K) = T * C(J,K)
1320 B(J,K) = T * B(J,K)
1330 NEXT K
1340 FOR L = 1 TO R
1350 IF L = J THEN 1410
1360 T = - C(L,J)
1370 FOR K = 1 TO R
1380 C(L,K) = C(L,K) + T * C(J,K)
1390 B(L,K) = B(L,K) + T * B(J,K)
1400 NEXT K
1410 NEXT L
1420 NEXT J
1425 PRINT D$;"PR#1"
1430 PRINT
1440 PRINT "THE INVERSE MATRIX IS:"
1445 PRINT "-----": PRINT
1450 FOR I = 1 TO R
1460 FOR J = 1 TO R
1470 PRINT "B(";I;",";J;")= ";B(I,J)
1480 NEXT J
1510 NEXT I
1515 PRINT D$;"PR#0"
1520 RETURN

```

```

5 D$ = CHR$ (4)
10 REM $$ FFTPROG $$
20 REM $$RADIX-2 FFT$$
40 HOME
50 VTAB 2: HTAB 15: PRINT "FOURIER TRANSFORM"
51 HTAB 15: PRINT "-----"
53 VTAB 10: INPUT "ARE THERE MULT.OF SIGNALS ?(Y/N)=";G$
54 IF G$ = "N" THEN GOTO 57
55 IF G$ < > "Y" THEN PRINT CHR$ (7): GOTO 53
56 H$ = "1": PRINT : INPUT "# OF MULTIPLICATION=";G$:G = VAL (G$)
57 PRINT D$;"OPEN ORNEK,D2"
58 PRINT D$;"READ ORNEK"
60 INPUT CC$
62 PRINT D$;"CLOSE ORNEK"
71 CC = VAL (CC$)
72 HOME
74 VTAB 10: PRINT "TOTAL # OF SAMPLES =" ;CC$;"
75 DIM X(CC)
76 DIM Y(CC)
77 DIM MAG(CC)
80 PRINT : INPUT "L? (NN=2^L)";F$
90 L = VAL (F$)
100 PRINT : INPUT "FFTI=?(-1 OR 1)";S$
110 FFTI = VAL (S$)
113 HOME : VTAB 10: PRINT "1-SPECTRUM OF THE INPUT"
115 PRINT : PRINT "2-SPECTRUM OF THE ERROR"
117 PRINT : PRINT "3-SPEC.OF INPUT AUTOCORR.COEFF."
120 PRINT : PRINT "4-SPEC.OF ERROR AUTOCORR.COEFF."
121 PRINT : PRINT "5-SPECTRUM OF MULTIPLE SIGNALS"
122 PRINT : PRINT "6-EXIT"
123 PRINT : INPUT "CHOOSE ONE OF THEM...=";S$
125 S = VAL (S$): IF S = 1 THEN SPEK$ = "GIRD1":JJ$ = "INPUT":FFTS$ = "GIRDIFFT": GOTO 140
127 IF S = 2 THEN SPEK$ = "HATA":JJ$ = "ERROR":FFTS$ = "HATAFFT": GOTO 140
130 IF S = 3 THEN SPEK$ = "AUTOB":JJ$ = "AUT.OF INP":FFTS$ = "AUTOBFFT": GOTO 140
135 IF S = 4 THEN SPEK$ = "AUTOH":JJ$ = "AUT.OF ERR.":FFTS$ = "AUTOHFFT": GOTO 140
136 IF S = 5 THEN PRINT D$;"RUN MAIN,D1"
137 GOTO 2000
140 IF (S = 1 OR S = 2) THEN NP = CC: GOTO 155
145 IF (S = 3 OR S = 4 AND CC < 32) THEN NP = 16: GOTO 155
150 IF (S = 3 OR S = 4 AND CC > 32) THEN NP = 32:L = 5
155 HOME : VTAB 10: HTAB 5: PRINT "SPECTRUM OF ";JJ$
160 HTAB 5: VTAB 12: PRINT "PLEASE WAIT..."
163 PRINT D$
165 PRINT D$;"OPEN";SPEK$;" ,L11,D2"
166 IF (S = 1 OR S = 2) THEN Z = 1: GOTO 170
167 PRINT D$;"READ";SPEK$;" ,R0"
168 INPUT VAR$:X(1) = VAL (VAR$):Y(1) = 0:Z = 2
170 FOR I = Z TO NP
175 PRINT D$;"READ";SPEK$;" ,R";I
178 INPUT ERR$:ERR = VAL (ERR$):X(I) = ERR
200 Y(I) = 0
210 NEXT I
215 PRINT D$;"CLOSE";SPEK$
217 PRINT JJ$;" VALUES"
218 FOR I = 1 TO NP: PRINT X(I): NEXT
220 LMX = NP
230 SCL = 6.283185 / NP

```

```

240 FOR L0 = 1 TO L
250 LIX = LMX
260 LMX = LMX / 2
270 ARG = 0
280 FOR L2M = 1 TO LMX
290 C = COS (ARG)
300 S = SIN (ARG)
310 ARG = ARG + SCL
320 FOR L1 = LIX TO NP STEP LIX
330 J1 = L1 - LIX + L2M
340 J2 = J1 + LMX
350 T1 = X(J1) - X(J2)
360 T2 = Y(J1) - Y(J2)
370 X(J1) = X(J1) + X(J2)
380 Y(J1) = Y(J1) + Y(J2)
390 X(J2) = C * T1 + S * T2
400 Y(J2) = C * T2 - S * T1
410 NEXT L1, L2M
420 SCL = 2 * SCL
430 NEXT L0
440 REM --BIT REVERSAL--
450 J = 1
460 NV2 = NP / 2
470 N1PM = NP - 1
480 FOR I = 1 TO N1PM
490 IF I >= J THEN GOTO 560
500 T1 = X(J)
510 T2 = Y(J)
520 X(J) = X(I)
530 Y(J) = Y(I)
540 X(I) = T1
550 Y(I) = T2
560 K = NV2
570 IF K >= J THEN GOTO 610
580 J = J - K
590 K = K / 2
600 GOTO 570
610 J = J + K
620 NEXT I
630 PRINT : PRINT : PRINT
640 HTAB 1: PRINT "COEFFICIENTS OF FOURIER TRANSFORMATION"
650 HTAB 1: PRINT "-----"
660 HTAB 1: PRINT "X(I)";; HTAB 20: PRINT "Y(I)"
670 HTAB 1: PRINT "----";; HTAB 20: PRINT "----"
680 FOR I = 1 TO N1PM
690 PRINT X(I);; HTAB 20: PRINT Y(I)
700 NEXT I
720 GOSUB 810: REM --MAGNITUDE S/R--
730 IF FFTI = 1 THEN GOTO 790
740 FOR I = 1 TO NP
750 X(I) = X(I) / NP

```

```

760 Y(I) = - Y(I) / NP
770 HTAB 1: PRINT X(I);: HTAB 20: PRINT Y(I)
780 NEXT I
790 A = 1
800 END
810 REM ** MAGNITUDE S/R **
830 IF L < 5 THEN Y = 1
840 Y = 5
850 FOR I = 1 TO N1PM
860 SX = X(I) * X(I)
870 SY = Y(I) * Y(I)
880 SUM = SX + SY
890 MAG(I) = SQR (SUM)
900 IF MAG(I) = 0 THEN MAG(I) = 0.0000001
910 MAG(I) = 10 * LOG (MAG(I))
920 NEXT I: PRINT D$;"PR#1"
930 PRINT : PRINT : PRINT
940 PRINT "MAGNITUDE VALUES (IN DB)"
950 PRINT "-----"
960 PRINT JJ$
970 PRINT "-----"
980 FOR I = 1 TO N1PM
990 PRINT MAG(I): NEXT I
995 PRINT D$;"PR#0"
1000 IF H$ = "1" THEN GOTO 1005
1002 C = 1:N = NP: GOTO 1006
1005 C = (6 - 1) * NP:N = NP * 6
1006 PRINT D$
1007 PRINT D$;"OPEN";FFT$;","L11,D2"
1008 FOR I = C TO N - 1:P = I - C + 1
1012 INP$ = STR$ (MAG(P))
1014 IF LEN (INP$) = 10 THEN GOTO 1023
1016 IF ( LEN (INP$) > 10 AND VAL (INP$) < .01 AND VAL (INP$) > 0) THEN INP$ = ".001000000": GOTO 1023
1018 IF LEN (INP$) > 10 THEN INP$ = LEFT$ (INP$,10): GOTO 1023
1020 FOR XX = LEN (INP$) TO 10
1022 INP$ = INP$ + " ": NEXT XX
1023 PRINT D$
1025 PRINT D$;"WRITE";FFT$;","R";I
1027 PRINT INP$
1028 NEXT I
1029 PRINT D$
1030 PRINT D$;"CLOSE";FFT$
1035 IF N1PM < 64 THEN CARP = 4: GOTO 1040
1036 IF N1PM < 128 THEN CARP = 2: GOTO 1040
1038 IF N1PM < 256 THEN CARP = 1: GOTO 1040
1039 IF N1PM > 256 THEN CARP = .5
1040 HOME : VTAB 23: HGR
1050 HPLLOT 0,100 TO 270,100
1060 HPLLOT 0,0 TO 0,190
1070 FOR I = 1 TO N1PM
1080 XI = I * CARP
1090 YI = MAG(I)
1100 HPLLOT TO 0 + XI,100 - YI
1110 NEXT I
1130 PRINT "FOURIER TRANSFORM OF ";JJ$
1150 PRINT D$;"PR#1": PRINT CHR$ (9);"B": PRINT D$;"PR#0"
1160 TEXT
1170 GOTO 113
2000 IF H$ = "1" THEN GOTO 5000

```



```

2005 PRINT D$;"OPEN GIRD1,L11,D2"
2010 PRINT D$;"DELETE GIRD1"
2020 PRINT D$;"OPEN HATA,L11,D2"
2030 PRINT D$;"DELETE HATA"
2040 PRINT D$;"OPEN AUTO6,L11,D2"
2050 PRINT D$;"DELETE AUTO6"
2060 PRINT D$;"OPEN AUTOH,L11,D2"
2070 PRINT D$;"DELETE AUTOH"
2080 PRINT D$;"OPEN ARALIK,L11,D2"
2090 PRINT D$;"DELETE ARALIK"
2100 PRINT D$;"OPEN IST,L11,D2"
2110 PRINT D$;"DELETE IST"
2120 PRINT D$;"OPEN FFT,L11,D2"
2130 PRINT D$;"DELETE FFT"
3000 HOME : END
5000 REM KATLAMALI SPEKTRUM CIZIMI
5002 FOR I = 1 TO NP
5005 MAG(I) = 0: NEXT
5006 HOME : VTAB 10: PRINT "1-MULT.SPECTRUM OF THE INPUT"
5007 PRINT : PRINT "2-MULT.SPECTRUM OF THE ERROR"
5008 PRINT : PRINT "3-MULT.SPEC.OF INPUT AUTOCORR.COEFF."
5009 PRINT : PRINT "4-MULT.SPEC.OF ERROR AUTOCORR.COEFF."
5010 PRINT : PRINT "5-EXIT"
5011 PRINT : INPUT "CHOOSE ONE OF THEM...=";S$
5014 S = VAL (S$): IF S = 1 THEN FFT$ = "GIRDIFFT": GOTO 5038
5015 IF S = 2 THEN FFT$ = "HATAFFT": GOTO 5038
5016 IF S = 3 THEN FFT$ = "AUTO6FFT": GOTO 5038
5017 IF S = 4 THEN FFT$ = "AUTOHFFT": GOTO 5038
5020 GOTO 2005
5038 PRINT D$;"OPEN";FFT$; ",L11,D2"
5039 FOR L = 1 TO N1PH
5040 FOR I = L TO N STEP N1PH
5050 PRINT D$;"READ";FFT$; ",R";I
5060 INPUT INP$
5070 MAG(L) = MAG(L) + VAL (INP$)
5080 NEXT I
5085 MAG(L) = MAG(L) / 6
5090 NEXT L
5100 PRINT D$;"CLOSE";FFT$
5150 HOME : VTAB 21
5160 HGR
5170 HPL0T 0,100 TD 270,100
5180 HPL0T 0,0 TO 0,190
5190 FOR I = 1 TO N1PH
5200 XI = I * CARP
5210 YI = MAG(I)
5220 HPL0T TD 0 + XI,100 - YI
5230 NEXT I
5240 PRINT "MULTIPLE FOURIER TRANSFORM OF ";JJ$
5270 PRINT D$;"PR#1": PRINT CHR$(9);"G": PRINT D$;"PR#0"
5280 TEXT
5290 GOTO 2005

```

A P P E N D I X B

PROGRAM LISTINGS OF
EDGE DETECTION
TECHNIQUES

```

10 REM *** MAIN MENU ***
20 D$ = CHR$(4)
30 HOME : VTAB 8: PRINT "1-VERTICAL EDGE"
40 PRINT "2-DIAGONAL EDGE"
50 PRINT "3-CIRCULAR SHAPE"
60 VTAB 13: INPUT "CHOOSE ONE OF THEM...";A
70 IF (A < 1 OR A > 3) THEN PRINT CHR$(7): GOTO 60
80 HOME : VTAB 8: PRINT "1-KIRSCH MASKS"
90 PRINT "2-SOBEL OPERATOR"
100 PRINT "3-NONLIN.EDGE DETEC,ROSENFIELD"
110 VTAB 12: INPUT "CHOOSE ONE OF THEM...";B
120 IF (B < 1 OR B > 3) THEN PRINT CHR$(7): GOTO 110
130 IF B = 3 THEN GOTO 180
140 HOME : VTAB 10: PRINT "1- FIXED THRESHOLD"
150 PRINT "2- LOCALLY ADAP.THRH."
160 VTAB 13: INPUT "CHOOSE ONE OF THEM...";QQ
170 IF (QQ < 1 OR QQ > 2) THEN PRINT CHR$(7): GOTO 160
180 ON A GOTO 190,210,230
190 PRINT D$;"BLOAD CHAIN,A520"
200 CALL 520"VERTICAL"
210 PRINT D$;"BLOAD CHAIN,A520"
220 CALL 520"DIAGONAL"
230 PRINT D$;"BLOAD CHAIN,A520"
240 CALL 520"CIRCULAR"

```

```

10 REM :$ VERTICAL EDGE :$
20 HOME : VTAB 10
30 INPUT "CHOOSE THE DIMENSION OF THE PICTURE:";BY
40 DIM A(BY,BY)
50 HOME : VTAB 10: PRINT "PLEASE CHOOSE THE CONTRAST VALUES:" : PRINT
60 INPUT "D1=";D1
70 INPUT "D2=";D2
80 IF A = 3 THEN GOTO 120
90 INPUT "D3=";D3
100 IF A = 2 THEN GOTO 120
110 INPUT "D4=";D4
120 REM DESIGNING A VERTICAL EDGE
130 REM NOISE DECISION
140 PRINT : INPUT "DO YOU WANT TO ADD NOISE? ";CEV$
150 IF CEV$ = "N" THEN GOTO 200
160 IF CEV$ < > "Y" THEN PRINT CHR$(7): GOTO 140
170 PRINT : INPUT "CHOOSE THE SIGMA VALUE=";SIG
180 PRINT D$;"PR#1": PRINT "NOISE ADDITION WITH SIGMA=";SIG
190 PRINT D$;"PR#0"
200 FOR I = 1 TO BY / 2 - 2
210 FOR J = 1 TO BY
220 VAR = 0
230 FOR K = 1 TO 12
240 RN = RND (K + J)
250 VAR = RN + VAR
260 NEXT K
270 VAR = 1 / SQRT (12) * VAR
280 A(J,I) = INT (VAR * D1)
290 NEXT J
300 NEXT I
310 FOR I = BY / 2 - 1 TO BY / 2
320 FOR J = 1 TO BY
330 VAR = 0
340 FOR K = 1 TO 12
350 RN = RND (K + J)
360 VAR = RN + VAR
370 NEXT K
380 VAR = 1 / SQRT (12) * VAR
390 A(J,I) = INT (VAR * D2)
400 NEXT J
410 NEXT I
420 FOR I = BY / 2 + 1 TO BY / 2 + 2
430 FOR J = 1 TO BY
440 VAR = 0
450 FOR K = 1 TO 12
460 RN = RND (K + J)
470 VAR = RN + VAR
480 NEXT K
490 VAR = 1 / SQRT (12) * VAR
500 A(J,I) = INT (VAR * D3)
510 NEXT J
520 NEXT I
530 FOR I = BY / 2 + 3 TO BY
540 FOR J = 1 TO BY
550 VAR = 0
560 FOR K = 1 TO 12
570 RN = RND (K + J)
580 VAR = RN + VAR
590 NEXT K
600 VAR = 1 / SQRT (12) * VAR
610 A(J,I) = INT (VAR * D4)

```

```
620 NEXT J
630 NEXT I
640 IF CEV% = "N" THEN GOTO 690
650 FOR K = 1 TO BY
660 FOR L = 1 TO BY
670 A(K,L) = INT ( RND ( K + L ) * SIG + A(K,L) )
680 NEXT L,K
690 ON B GOTO 700,750,800
700 IF QR = 2 THEN GOTO 730
710 PRINT D$;"BLOAD CHAIN,A520"
720 CALL 520"VERT.KIRFIX"
730 PRINT D$;"BLOAD CHAIN,A520"
740 CALL 520"VERT.KIRADAP"
750 IF QR = 2 THEN GOTO 780
760 PRINT D$;"BLOAD CHAIN,A520"
770 CALL 520"VERT.SOBFIX"
780 PRINT D$;"BLOAD CHAIN,A520"
790 CALL 520"VERT.SOBADAP"
800 PRINT D$;"BLOAD CHAIN,A520"
810 CALL 520"VERT.R0S"
```

```

10 REM ** DIAGONAL EDGE **
20 HOME : VTAB 10
30 INPUT "CHOOSE THE DIMENSION OF THE PICTURE:";BY
40 DIM A(BY,BY)
50 HOME : VTAB 10: PRINT "PLEASE CHOOSE THE CONTRAST VALUES:" : PRINT
60 INPUT "D1=";D1
70 INPUT "D2=";D2
80 IF A = 3 THEN GOTO 120
90 INPUT "D3=";D3
100 IF A = 2 THEN GOTO 120
110 INPUT "D4=";D4
120 REM DESIGNING A DIAGONAL EDGE
130 REM NOISE DECISION
140 PRINT : INPUT "DO YOU WANT TO ADD NOISE? ";CEV$
150 IF CEV$ = "N" THEN GOTO 200
160 IF CEV$ < > "Y" THEN PRINT CHR$(7): GOTO 140
170 PRINT : INPUT "CHOOSE THE SIGMA VALUE=";SIG
180 PRINT D$;"PR#1": PRINT "NOISE ADDITION WITH SIGMA=";SIG
190 PRINT D$;"PR#0"
200 FOR I = 1 TO BY
210 FOR J = 1 TO BY
220 VAR = 0
230 FOR K = 1 TO 12
240 RN = RND (K + J)
250 VAR = RN + VAR
260 NEXT K
270 VAR = (1 / SQR (12)) * VAR
280 IF I > J THEN A(I,J) = INT (VAR * D1): GOTO 310
290 IF I = J THEN A(I,J) = INT (VAR * D2): GOTO 310
300 A(I,J) = INT (VAR * D3)
310 NEXT J,I
320 IF CEV$ = "N" THEN GOTO 370
330 FOR K = 1 TO BY
340 FOR L = 1 TO BY
350 A(K,L) = INT ( RND (K + L) * SIG + A(K,L))
360 NEXT L,K
370 ON B GOTO 380,430,480
380 IF QQ = 2 THEN GOTO 410
390 PRINT D$;"BLOAD CHAIN,A520"
400 CALL 520"VERT.KIRFIX"
410 PRINT D$;"BLOAD CHAIN,A520"
420 CALL 520"VERT.KIRADAP"
430 IF QQ = 2 THEN GOTO 460
440 PRINT D$;"BLOAD CHAIN,A520"
450 CALL 520"VERT.SOBFIX"
460 PRINT D$;"BLOAD CHAIN,A520"
470 CALL 520"VERT.SOBADAP"
480 PRINT D$;"BLOAD CHAIN,A520"
490 CALL 520"VERT.ROS"

```

```

10 REM ** CIRCULAR SHAPE **
20 HOME : VTAB 10
30 INPUT "CHOOSE THE DIMENSION OF THE PICTURE:";BY
40 DIM A(BY,BY)
50 HOME : VTAB 10: PRINT "PLEASE CHOOSE THE CONTRAST VALUES:" : PRINT
60 INPUT "D1=";D1
70 INPUT "D2=";D2
80 PRINT : INPUT "CHOOSE THE RADIOUS OF THE CIRCLE=";RA
90 IF RA > BY / 2 THEN PRINT CHR$(7): GOTO 80
120 REM DESIGNING A CIRCULAR SHAPE
130 REM NOISE DECISION
140 PRINT : INPUT "DO YOU WANT TO ADD NOISE? ";CEV$
150 IF CEV$ = "N" THEN GOTO 200
160 IF CEV$ < "Y" THEN PRINT CHR$(7): GOTO 140
170 PRINT : INPUT "CHOOSE THE SIGMA VALUE=";SIG
180 PRINT D$;"PR#1": PRINT "NOISE ADDITION WITH SIGMA=";SIG
190 PRINT D$;"PR#0"
200 XC = BY / 2:YC = BY / 2
210 FOR K = 1 TO BY
220 FOR L = 1 TO BY
230 VAR = 0
240 FOR I = 1 TO 12
250 RN = RND (K + J)
260 VAR = VAR + RN
270 NEXT I
280 VAR = 1 / SQR (12) * VAR
290 IF ((K - XC) ^ 2 + (L - YC) ^ 2) < RA ^ 2 THEN A(K,L) = INT (VAR * D2): GOTO 310
300 A(K,L) = INT (VAR * D1)
310 NEXT L,K
320 IF CEV$ = "N" THEN GOTO 370
330 FOR K = 1 TO BY
340 FOR L = 1 TO BY
350 A(K,L) = INT (RND (K + L) * SIG + A(K,L))
360 NEXT L,K
370 ON B GOTO 380,430,480
380 IF QQ = 2 THEN GOTO 410
390 PRINT D$;"BLOAD CHAIN,A520"
400 CALL 520"VERT.KIRFIX"
410 PRINT D$;"BLOAD CHAIN,A520"
420 CALL 520"VERT.KIRADAP"
430 IF QQ = 2 THEN GOTO 460
440 PRINT D$;"BLOAD CHAIN,A520"
450 CALL 520"VERT.SOBFIX"
460 PRINT D$;"BLOAD CHAIN,A520"
470 CALL 520"VERT.SOBADAP"
480 PRINT D$;"BLOAD CHAIN,A520"
490 CALL 520"VERT.ROS"

```

```

10 REM ** VERTICAL EDGE-SOBEL OP. WITH ADAP. THR. **
20 DIM AA(3,3),SN((BY * BY) / 2,3)
30 DIM GD(BY,BY),T(BY,BY),R(BY,BY)
40 PRINT D$;"PR#1"
50 PRINT : PRINT : PRINT "EDGE DETECTION BY USING SOBEL OPERATOR & ADAPTIVE THRESH."
60 PRINT "-----"
70 PRINT "CONTRASTS:D1=";D1
80 PRINT "      D2=";D2
85 IF A = 3 THEN GOTO 110
90 PRINT "      D3=";D3
95 IF A = 2 THEN GOTO 110
100 PRINT "      D4=";D4
110 PRINT : PRINT : PRINT "ORIGINAL IMAGE"
120 PRINT "-----"
130 PRINT : PRINT
140 PRINT CHR$(15): POKE 1657,130
150 FOR J = 1 TO BY
160 FOR I = 1 TO BY
170 X$ = STR$(A(J,I));Y$ = ""
180 FOR K = LEN(X$) TO 3:Y$ = Y$ + " ":NEXT K
190 PRINT A(J,I);Y$;
200 NEXT I
210 PRINT
220 NEXT J
230 GOSUB 7000: REM BIT IMAGE-PR.
2000 REM SOBEL OPERATION
2010 FOR K = 1 TO BY
2020 FOR L = 1 TO BY
2030 FOR I = 1 TO 3
2040 FOR J = 1 TO 3
2050 AA(I,J) = 0
2060 NEXT J
2070 NEXT I
2080 I = 1:Y = 1
2090 FOR I = K - 1 TO K + 1
2100 FOR J = L - 1 TO L + 1
2110 IF (I = 0 OR J = 0) THEN GOTO 2140
2120 IF (I = BY + 1 OR J = BY + 1) THEN GOTO 2140
2130 AA(I,J) = A(I,J)
2140 Y = Y + 1
2150 NEXT J
2160 I = I + 1:Y = 1
2170 NEXT I
2180 XX = (AA(I,3) + 2 * AA(2,3) + AA(3,3)) - (AA(1,1) + 2 * AA(2,1) + AA(3,1))
2190 YY = (AA(1,1) + 2 * AA(1,2) + AA(1,3)) - (AA(3,1) + 2 * AA(3,2) + AA(3,3))
2200 REM GD(K,L) = SQR (XX ^ 2 + YY ^ 2):GD(K,L) = INT (GD(K,L))
2210 GD(K,L) = ABS (XX) + ABS (YY):GD(K,L) = INT (GD(K,L))
2220 NEXT L
2230 NEXT K
2240 PRINT : PRINT : PRINT "GRADIENT IMAGE"
2250 PRINT "-----"
2260 PRINT : PRINT : PRINT CHR$(15)

```



```

2270 FOR J = 1 TO BY
2280 FOR I = 1 TO BY
2290 X$ = STR$ (GD(J,I)); Y$ = ""
2300 FOR K = LEN (X$) TO 4: Y$ = Y$ + " ": NEXT K
2310 PRINT GD(J,I); Y$;
2320 NEXT I
2330 PRINT
2340 NEXT J
2350 FOR I = 1 TO BY
2360 FOR J = 1 TO BY
2370 A(I,J) = GD(I,J)
2380 NEXT J
2390 NEXT I
2400 GOSUB 7000: REM BIT IM.PR.
4000 REM ** THRESHOLDING **
4010 FOR K = 1 TO BY
4020 FOR L = 1 TO BY
4030 Y = 1: CNV = 0
4040 FOR I = K - 1 TO K + 1
4050 FOR J = L - 1 TO L + 1
4060 IF (I = 0 OR J = 0) THEN GOTO 4100
4070 IF (I = BY + 1 OR J = BY + 1) THEN GOTO 4100
4080 CNV = CNV + GD(I,J)
4090 Y = Y + 1
4100 NEXT J
4110 NEXT I
4120 CNV = CNV / Y
4130 IF (K = 1 OR L = 1) THEN 4160
4140 IF (K = BY OR L = BY) THEN 4160
4150 IF GD(K,L) > CNV THEN T(K,L) = 1: GOTO 4170
4160 T(K,L) = 0
4170 NEXT L
4180 NEXT K
4190 PRINT : PRINT
4200 PRINT "ADAPTIVE THRESHOLDING"
4210 PRINT "-----"
4220 PRINT
4300 FOR K = 1 TO BY
4310 FOR L = 1 TO BY
4320 IF T(K,L) = 1 THEN GOTO 4380
4330 PRINT CHR$ (27); "K"; CHR$ (8); CHR$ (0);
4340 FOR Q = 1 TO 8
4350 PRINT CHR$ (8(12,Q));
4360 NEXT Q
4370 GOTO 4420
4380 PRINT CHR$ (27); "K"; CHR$ (8); CHR$ (0);
4390 FOR Q = 1 TO 8
4400 PRINT CHR$ (8(1,Q));
4410 NEXT Q

```

```

4420 NEXT L
4430 PRINT
4440 NEXT K
4445 GOSUB 12000: REM ANALYSIS
4450 REM --LIMITING #--
4460 L = 0
4470 FOR I = 1 TO BY
4480 FOR J = 1 TO BY
4490 IF T(I,J) < > 1 THEN GOTO 4520
4500 L = L + 1
4510 SN(L,1) = 6D(I,J):SN(L,2) = I:SN(L,3) = J
4520 NEXT J
4530 NEXT I
4540 PRINT : PRINT "NUMBER OF BLACK PNTS=";L
4550 PRINT D$;"PR#0"
4560 HOME : VTAB 10: INPUT "CHOOSE THE MAXIMUM # AS A THRESHOLD:";LIM
4570 PRINT D$;"PR#1"
4580 PRINT : PRINT "THRESHOLD AS A LIMIT.# IS= ";LIM
4590 PRINT "-----"
4600 FOR I = 1 TO L
4610 FOR J = I + 1 TO L
4620 IF SN(I,1) > SN(J,1) THEN GOTO 4680
4630 FOR K = 1 TO 3
4640 TEMP = SN(I,K)
4650 SN(I,K) = SN(J,K)
4660 SN(J,K) = TEMP
4670 NEXT K
4680 NEXT J
4690 NEXT I
4700 FOR I = 1 TO BY
4710 FOR J = 1 TO BY
4720 T(I,J) = 0
4730 NEXT J
4740 NEXT I
4750 FOR K = 1 TO LIM
4760 T(SN(K,2),SN(K,3)) = 1
4770 NEXT K
4850 PRINT
4860 FOR K = 1 TO BY
4870 FOR L = 1 TO BY
4880 IF T(K,L) = 1 THEN GOTO 4940
4890 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
4900 FOR Q = 1 TO 8
4910 PRINT CHR$(B(12,Q));
4920 NEXT Q
4930 GOTO 4980
4940 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
4950 FOR Q = 1 TO 8
4960 PRINT CHR$(B(1,Q));
4970 NEXT Q
4980 NEXT L
4990 PRINT
5000 NEXT K
5005 GOSUB 12000: REM ANALYSIS
5010 PRINT D$
5020 PRINT D$;"PR#0"
5030 END
7000 REM *** BITIMAGE PRINTING ***
7010 MAX = A(1,1)
7020 FOR K = 1 TO BY

```

```
7030 FOR L = 1 TO BY
7040 IF A(K,L) > MAX THEN MAX = A(K,L)
7050 NEXT L
7060 NEXT K
7070 MULT = 400 / MAX
7080 PRINT : PRINT "MULTIPLICATION VALUE=";MULT: PRINT
7090 FOR K = 1 TO BY
7100 FOR L = 1 TO BY
7110 R(K,L) = INT (A(K,L) * MULT)
7120 NEXT L
7130 NEXT K
7250 PRINT CHR$(18)
7260 FOR J = 1 TO BY
7270 FOR I = 1 TO BY
7280 IF R(J,I) > 280 THEN 8010
7290 IF R(J,I) > 240 THEN 8060
7300 IF R(J,I) > 210 THEN 8120
7310 IF R(J,I) > 190 THEN 8180
7320 IF R(J,I) > 175 THEN 8240
7330 IF R(J,I) > 160 THEN 8300
7340 IF R(J,I) > 135 THEN 8360
7350 IF R(J,I) > 110 THEN 8420
7360 IF R(J,I) > 80 THEN 8480
7370 IF R(J,I) > 50 THEN 8540
7380 IF R(J,I) > 40 THEN 8600
7390 IF R(J,I) > 20 THEN 8660
7400 IF R(J,I) > 10 THEN 8720
7410 GOTO 8780
7420 NEXT I
7430 PRINT
7440 NEXT J
7450 RETURN
8000 REM
8010 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8020 FOR Q = 1 TO 8
8030 PRINT CHR$(B(1,Q));
8040 NEXT Q
8050 GOTO 7420
8060 REM
8070 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8080 FOR Q = 1 TO 8
8090 PRINT CHR$(B(2,Q));
8100 NEXT Q
8110 GOTO 7420
8120 REM
8130 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8140 FOR Q = 1 TO 8
8150 PRINT CHR$(B(3,Q));
8160 NEXT Q
8170 GOTO 7420
8180 REM
8190 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8200 FOR Q = 1 TO 8
8210 PRINT CHR$(B(4,Q));
8220 NEXT Q
8230 GOTO 7420
8240 REM
8250 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
```

```
8260 FOR Q = 1 TO 8
8270 PRINT CHR$ (B(5,Q));
8280 NEXT Q
8290 GOTO 7420
8300 REM
8310 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8320 FOR Q = 1 TO 8
8330 PRINT CHR$ (B(6,Q));
8340 NEXT Q
8350 GOTO 7420
8360 REM
8370 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8380 FOR Q = 1 TO 8
8390 PRINT CHR$ (B(7,Q));
8400 NEXT Q
8410 GOTO 7420
8420 REM
8430 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8440 FOR Q = 1 TO 8
8450 PRINT CHR$ (B(8,Q));
8460 NEXT Q
8470 GOTO 7420
8480 REM
8490 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8500 FOR Q = 1 TO 8
8510 PRINT CHR$ (B(9,Q));
8520 NEXT Q
8530 GOTO 7420
8540 REM
8550 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8560 FOR Q = 1 TO 8
8570 PRINT CHR$ (B(10,Q));
8580 NEXT Q
8590 GOTO 7420
8600 REM
8610 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8620 FOR Q = 1 TO 8
8630 PRINT CHR$ (B(11,Q));
8640 NEXT Q
8650 GOTO 7420
8660 REM
8670 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8680 FOR Q = 1 TO 8
8690 PRINT CHR$ (B(12,Q));
8700 NEXT Q
8710 GOTO 7420
8720 REM
8730 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8740 FOR Q = 1 TO 8
8750 PRINT CHR$ (B(13,Q));
8760 NEXT Q
8770 GOTO 7420
8780 REM
8790 PRINT CHR$ (27);"K"; CHR$ (8); CHR$ (0);
8800 FOR Q = 1 TO 8
8810 PRINT CHR$ (B(14,Q));
8820 NEXT Q
8830 GOTO 7420
12000 REM ** I ERROR S/R **
12005 PRINT : PRINT
12010 BLA = 0:VAR = 0:ERR = 0
```

```

12015 FOR K = 1 TO BY
12020 FOR L = 1 TO BY
12025 IF T(K,L) = 0 THEN GOTO 12035
12030 BLA = BLA + 1
12035 NEXT L,K
12040 ON A GOTO 12045,12400,12600
12045 REM VERTICAL EDGE REGION
12050 FOR K = 2 TO BY - 1
12055 FOR L = BY / 2 TO BY / 2 + 1
12060 IF T(K,L) = 0 THEN GOTO 12070
12065 VAR = VAR + 1
12070 NEXT L,K
12075 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12080 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12085 PRINT :FARK = BLA - VAR
12090 FOR K = 2 TO BY - 1
12095 IF (T(K,BY / 2) = 0 AND T(K,BY / 2 + 1) = 0) THEN ERR = ERR + 1
12100 NEXT K
12105 PRINT "UNCOVERED ROWS IN EDGE REGION(MISSING EDGE PNT)=";ERR
12110 PRINT "# OF EDGE POINTS=";BY - 2
12115 PRINT "# OF MISSING EDGE PNT./# OF EDGE PNT=";ERR / (BY - 2)
12120 ERR = ERR + BLA - VAR
12125 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12130 VAR = 0
12135 FOR K = 1 TO BY
12140 FOR L = 1 TO BY / 2 - 1
12145 IF T(K,L) = 0 THEN GOTO 12155
12150 UZ = BY / 2 - 2 - L:UZ = UZ ^ 2:VAR = VAR + UZ
12155 NEXT L
12160 FOR L = BY / 2 + 2 TO BY
12165 IF T(K,L) = 0 THEN GOTO 12180
12170 UZ = L - BY / 2 + 1:UZ = UZ ^ 2:VAR = VAR + UZ
12175 NEXT L
12180 NEXT K
12185 IF FARK = 0 THEN GOTO 12200
12190 VAR = VAR / FARK
12195 PRINT "MEAN SQUARE DISTANCE OF ERROR=";VAR
12200 PRINT
12205 RETURN
12400 REM DIAGONAL EDGE REGION
12405 FOR K = 2 TO BY - 1
12410 FOR L = K TO K + 1
12415 IF T(K,L) = 0 THEN GOTO 12425
12420 VAR = VAR + 1
12425 NEXT L,K
12430 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12435 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12440 PRINT :FARK = BLA - VAR
12445 FOR K = 2 TO BY - 1
12450 IF (T(K,K) = 0 AND T(K,K + 1) = 0) THEN ERR = ERR + 1
12455 NEXT K
12460 PRINT "UNCOVERED ROWS IN EDGE REGION(MISSING EDGE PNT)=";ERR
12465 PRINT "# OF EDGE POINTS=";BY - 2
12470 PRINT "# OF MISSING EDGE PNT./# OF EDGE PNT=";ERR / (BY - 2)
12475 ERR = ERR + BLA - VAR

```

```

12480 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12485 VAR = 0
12490 FOR K = 2 TO BY
12495 FOR L = 1 TO K - 1
12500 IF T(K,L) = 0 THEN GOTO 12510
12505 UZ = K - L:UZ = UZ ^ 2:VAR = VAR + UZ
12510 NEXT L
12515 FOR L = K + 2 TO BY
12520 IF L > BY THEN GOTO 12535
12525 IF T(K,L) = 0 THEN GOTO 12535
12530 UZ = L - K + 1:UZ = UZ ^ 2:VAR = VAR + UZ
12535 NEXT L
12540 NEXT K
12545 GOTO 12190
12600 REM CIRCULAR SHAPE REGION
12610 FOR K = 1 TO BY
12620 FOR L = 1 TO BY
12630 IF (((K - XC) ^ 2 + (L - YC) ^ 2) > = RA ^ 2 AND ((K - XC) ^ 2 + (L - YC) ^ 2) < = (RA + 1) ^ 2) THEN GOTO 12660
12640 IF T(K,L) = 0 THEN GOTO 12660
12650 VAR = VAR + 1
12660 NEXT L,K
12670 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12680 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12690 PRINT :FARK = BLA - VAR
12700 EDGE = 0
12710 FOR K = 1 TO BY
12720 FOR L = 1 TO BY
12730 IF (((K - XC) ^ 2 + (L - YC) ^ 2) > = RA ^ 2 AND ((K - XC) ^ 2 + (L - YC) ^ 2) < = (RA + 1) ^ 2) THEN GOTO 12750
12740 EDGE = EDGE + 1
12750 NEXT L,K
12760 PRINT "# OF EDGE POINTS=";EDGE
12770 PRINT "# OF MISSING EDGE PNT=";EDGE - VAR: PRINT
12780 PRINT "# OF MISSING EDGE PNT/# OF EDGE PNT=";(EDGE - VAR) / EDGE
12790 ERR = EDGE - VAR + FARK
12800 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12810 RETURN

```

```

10 REM **VERTICAL EDGE-KIRSCH MASKS WITH FIXED THRH. **
20 DIM MSK(3,3),G(8)
30 DIM GD(BY,BY),T(BY,BY),DR(BY,BY),R(BY,BY)
40 PRINT D$;"PR#1"
50 PRINT : PRINT "EDGE DETECTION BY USING KIRSCH MASKS ,CONNECTIVITY TEST AND FIX.THRESH."
60 PRINT "-----"
70 PRINT "CONTRASTS:D1=";D1
80 PRINT "      D2=";D2
85 IF A = 3 THEN GOTO 110
90 PRINT "      D3=";D3
95 IF A = 2 THEN GOTO 110
100 PRINT "      D4=";D4
110 PRINT : PRINT : PRINT "ORIGINAL IMAGE"
120 PRINT "-----"
130 PRINT
230 GOSUB 7000: REM BIT IMAGE PR.
2000 REM CONVOLUTION ALG.
2010 HOME
2020 THR = 0: PRINT : PRINT
2030 FOR K = 1 TO BY
2040 FOR L = 1 TO BY
2050 FOR MM = 1 TO 8
2060 FOR I = 1 TO 3
2070 FOR J = 1 TO 3
2080 READ MSK(I,J)
2090 NEXT J
2100 NEXT I
2110 DATA 5,5,5,-3,0,-3,-3,-3,-3
2120 DATA 5,5,-3,5,0,-3,-3,-3,-3
2130 DATA 5,-3,-3,5,0,-3,5,-3,-3
2140 DATA -3,-3,-3,5,0,-3,5,5,-3
2150 DATA -3,-3,-3,-3,0,-3,5,5,5
2160 DATA -3,-3,-3,-3,0,5,-3,5,5
2170 DATA -3,-3,5,-3,0,5,-3,-3,5
2180 DATA -3,5,5,-3,0,5,-3,-3,-3
2190 X = 1:Y = 1:CNV = 0
2200 FOR I = K - 1 TO K + 1
2210 FOR J = L - 1 TO L + 1
2220 IF (I = 0 OR J = 0) THEN GOTO 2250
2230 IF (I = BY + 1 OR J = BY + 1) THEN GOTO 2250
2240 CNV = CNV + MSK(I,Y) * A(I,J)
2250 Y = Y + 1
2260 NEXT J
2270 X = X + 1:Y = 1
2280 NEXT I
2290 G(MM) = CNV
2300 REM PRINT G(MM);" ";
2310 NEXT MM
2320 REM PRINT
2330 RESTORE
2340 GD(K,L) = G(1)
2350 FOR MM = 1 TO 8
2360 IF GD(K,L) > G(MM) THEN GOTO 2380
2370 GD(K,L) = G(MM)
2380 NEXT MM
2390 FOR MM = 1 TO 8
2400 IF GD(K,L) = G(MM) THEN GOTO 2420

```

```

2410 NEXT MM
2420 DR(K,L) = MM
2430 THR = THR + GD(K,L)
2440 NEXT L
2450 NEXT K
2460 THR = THR / (BY * BY)
2470 POKE 1657,130
2480 PRINT : PRINT : PRINT "GRADIANT IMAGE"
2490 PRINT "-----"
2590 FOR I = 1 TO BY
2600 FOR J = 1 TO BY
2610 A(I,J) = GD(I,J)
2620 NEXT J
2630 NEXT I
2640 GOSUB 7000: REM BITIMAGE PR.
2650 PRINT : PRINT
4000 REM ** THRESHOLDING **
4010 PRINT "THRESHOLDING"
4020 PRINT "-----"
4025 PRINT "EDGE DETEC. WITHOUT CONNECTIVITY"
4026 PRINT "-----"
4030 PRINT
4040 FOR K = 1 TO BY
4050 FOR L = 1 TO BY
4052 IF (K = 1 OR L = 1) THEN GOTO 4065
4055 IF (K = BY OR L = BY) THEN GOTO 4065
4060 IF GD(K,L) > THR THEN T(K,L) = 1: GOTO 4120
4065 T(K,L) = 0
4070 PRINT CHR$(27); "K"; CHR$(8); CHR$(0);
4080 FOR Q = 1 TO 8
4090 PRINT CHR$(B(12,Q));
4100 NEXT Q
4110 GOTO 4160
4120 PRINT CHR$(27); "K"; CHR$(8); CHR$(0);
4130 FOR Q = 1 TO 8
4140 PRINT CHR$(B(1,Q));
4150 NEXT Q
4160 NEXT L
4170 PRINT
4180 NEXT K
4190 GOSUB 12000: REM ANALYSIS
5000 REM *** EDGE DIRECTION MAP ***
5010 PRINT : PRINT : PRINT "EDGE DIRECTION MAP"
5020 PRINT "-----"
5030 PRINT
5040 FOR K = 1 TO BY
5050 FOR L = 1 TO BY
5060 PRINT DR(K,L); " ";
5070 NEXT L
5080 PRINT
5090 NEXT K
6000 REM ** DECISION !!!! **
6010 FOR K = 1 TO BY
6020 FOR L = 1 TO BY
6030 GD(K,L) = 0
6040 NEXT L
6050 NEXT K
6060 FOR K = 1 TO BY
6070 FOR L = 1 TO BY
6080 IF T(K,L) = 0 THEN GOTO 6290

```



```

6100 FOR J = 1 TO 3
6110 MSK(I,J) = 0
6120 NEXT J
6130 NEXT I
6135 X = 1:Y = 1:O = 1
6140 FOR I = K - 1 TO K + 1
6150 FOR J = L - 1 TO L + 1
6160 IF (I = 0 OR J = 0) THEN MSK(X,Y) = 1: GOTO 6195
6170 IF (I = BY + 1 OR J = BY + 1) THEN MSK(X,Y) = 1: GOTO 6195
6180 IF (DR(I,J) = DR(K,L) OR DR(I,J) = DR(K,L) + 1 OR DR(I,J) = DR(K,L) - 1) THEN MSK(X,Y) = 1: GOTO 6195
6185 IF (DR(I,J) = DR(K,L) + 2 AND O = 1) THEN MSK(X,Y) = 1: GOTO 6192
6186 IF (DR(I,J) = DR(K,L) - 2 AND O = 1) THEN MSK(X,Y) = 1: GOTO 6192
6187 IF (DR(I,J) = DR(K,L) + 2 AND O = 2) THEN MSK(X,Y) = 1: GOTO 6192
6188 IF (DR(I,J) = DR(K,L) - 2 AND O = 2) THEN MSK(X,Y) = 1: GOTO 6192
6190 MSK(X,Y) = 0: GOTO 6195
6192 O = O + 1
6195 Y = Y + 1
6200 NEXT J
6205 X = X + 1:Y = 1
6210 NEXT I
6220 FOR I = 1 TO 3
6230 FOR J = 1 TO 3
6240 IF MSK(I,J) = 1 THEN GOTO 6260
6250 GD(K,L) = 0: GOTO 6290
6260 NEXT J
6270 NEXT I
6280 GD(K,L) = 1
6285 REM IF (MSK(1,2) = 1 AND MSK(2,1) = 1 AND MSK(2,3) = 1 AND MSK(3,2) = 1) THEN GD(K,L) = 1: GOTO 6290
6288 REM GD(K,L) = 0
6290 NEXT L
6300 NEXT K
6310 PRINT : PRINT
6320 PRINT "EDGE DETECTION USING CONNECTIVITY TEST"
6330 PRINT "-----": PRINT
6340 FOR K = 1 TO BY
6350 FOR L = 1 TO BY
6360 IF GD(K,L) = 0 THEN GOTO 6420
6370 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
6380 FOR Q = 1 TO 8
6390 PRINT CHR$(B(1,Q));
6400 NEXT Q
6410 GOTO 6460
6420 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
6430 FOR Q = 1 TO 8
6440 PRINT CHR$(B(12,Q));
6450 NEXT Q
6460 T(K,L) = GD(K,L)
6465 NEXT L
6470 PRINT
6480 NEXT K
6485 GOSUB 12000: REM ANALYSIS
6490 PRINT D$
6500 PRINT D$;"PR#0"
6510 END
7000 REM *** BITIMAGE PRINTING ***
7010 MAX = A(1,1)
7020 FOR K = 1 TO BY
7030 FOR L = 1 TO BY
7040 IF A(K,L) > MAX THEN MAX = A(K,L)
7050 NEXT L
7060 NEXT K
7070 MULT = 400 / MAX
7080 PRINT : PRINT "MULTIPLICATION VALUE=";MULT: PRINT
7090 FOR K = 1 TO BY

```

```
7100 FOR L = 1 TO BY
7110 R(K,L) = INT (A(K,L) * MULT)
7120 NEXT L
7130 NEXT K
7250 PRINT CHR$(18)
7260 FOR J = 1 TO BY
7270 FOR I = 1 TO BY
7280 IF R(J,I) > 280 THEN 8010
7290 IF R(J,I) > 240 THEN 8060
7300 IF R(J,I) > 210 THEN 8120
7310 IF R(J,I) > 190 THEN 8180
7320 IF R(J,I) > 175 THEN 8240
7330 IF R(J,I) > 160 THEN 8300
7340 IF R(J,I) > 135 THEN 8360
7350 IF R(J,I) > 110 THEN 8420
7360 IF R(J,I) > 80 THEN 8480
7370 IF R(J,I) > 50 THEN 8540
7380 IF R(J,I) > 40 THEN 8600
7390 IF R(J,I) > 20 THEN 8660
7400 IF R(J,I) > 10 THEN 8720
7410 GOTO 8780
7420 NEXT I
7430 PRINT
7440 NEXT J
7450 RETURN
8000 REM
8010 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8020 FOR Q = 1 TO 8
8030 PRINT CHR$(B(1,Q));
8040 NEXT Q
8050 GOTO 7420
8060 REM
8070 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8080 FOR Q = 1 TO 8
8090 PRINT CHR$(B(2,Q));
8100 NEXT Q
8110 GOTO 7420
8120 REM
8130 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8140 FOR Q = 1 TO 8
8150 PRINT CHR$(B(3,Q));
8160 NEXT Q
8170 GOTO 7420
8180 REM
8190 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8200 FOR Q = 1 TO 8
8210 PRINT CHR$(B(4,Q));
8220 NEXT Q
8230 GOTO 7420
8240 REM
8250 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8260 FOR Q = 1 TO 8
8270 PRINT CHR$(B(5,Q));
8280 NEXT Q
8290 GOTO 7420
8300 REM
8310 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8320 FOR Q = 1 TO 8
8330 PRINT CHR$(B(6,Q));
8340 NEXT Q
8350 GOTO 7420
8360 REM
8370 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8380 FOR Q = 1 TO 8
8390 PRINT CHR$(B(7,Q));
```

```
8400 NEXT Q
8410 GOTO 7420
8420 REM
8430 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8440 FOR Q = 1 TO 8
8450 PRINT CHR$(B(8,Q));
8460 NEXT Q
8470 GOTO 7420
8480 REM
8490 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8500 FOR Q = 1 TO 8
8510 PRINT CHR$(B(9,Q));
8520 NEXT Q
8530 GOTO 7420
8540 REM
8550 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8560 FOR Q = 1 TO 8
8570 PRINT CHR$(B(10,Q));
8580 NEXT Q
8590 GOTO 7420
8600 REM
8610 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8620 FOR Q = 1 TO 8
8630 PRINT CHR$(B(11,Q));
8640 NEXT Q
8650 GOTO 7420
8660 REM
8670 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8680 FOR Q = 1 TO 8
8690 PRINT CHR$(B(12,Q));
8700 NEXT Q
8710 GOTO 7420
8720 REM
8730 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8740 FOR Q = 1 TO 8
8750 PRINT CHR$(B(13,Q));
8760 NEXT Q
8770 GOTO 7420
8780 REM
8790 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8800 FOR Q = 1 TO 8
8810 PRINT CHR$(B(14,Q));
8820 NEXT Q
8830 GOTO 7420
12000 REM **Z ERROR S/R **
12010 PRINT : PRINT
12020 BLA = 0:VAR = 0:ERR = 0
12030 FOR K = 1 TO BY
12040 FOR L = 1 TO BY
12050 IF T(K,L) = 0 THEN GOTO 12070
12060 BLA = BLA + 1
12070 NEXT L,K
12075 ON A GOTO 12087,12400,12600
12087 REM VERTICAL-EDGE REGION
12090 FOR K = 2 TO BY - 1
```

```

12100 FOR L = BY / 2 TO BY / 2 + 1
12110 IF T(K,L) = 0 THEN GOTO 12130
12120 VAR = VAR + 1
12130 NEXT L,K
12150 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12160 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12165 PRINT :FARK = BLA - VAR
12170 FOR K = 2 TO BY - 1
12180 IF (T(K,BY / 2) = 0 AND T(K,BY / 2 + 1) = 0) THEN ERR = ERR + 1
12190 NEXT K
12200 PRINT "UNCOVERED ROWS IN EDGE REGION(MISSING EDGE PNT)=";ERR
12205 PRINT "# OF EDGE POINTS=";BY - 2
12210 PRINT "# OF MISSING EDGE PNT./# OF EDGE PNT=";ERR / (BY - 2)
12220 ERR = ERR + BLA - VAR
12230 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12233 VAR = 0
12234 FOR K = 1 TO BY
12235 FOR L = 1 TO BY / 2 - 1
12236 IF T(K,L) = 0 THEN GOTO 12240
12237 UZ = BY / 2 - 2 - L:UZ = UZ ^ 2:VAR = VAR + UZ
12240 NEXT L
12242 FOR L = BY / 2 + 2 TO BY
12243 IF T(K,L) = 0 THEN GOTO 12246
12244 UZ = L - BY / 2 + 1:UZ = UZ ^ 2:VAR = VAR + UZ
12245 NEXT L
12246 NEXT K
12247 IF FARK = 0 THEN GOTO 12250
12248 VAR = VAR / FARK
12249 PRINT "MEAN SQUARE DISTANCE OF ERROR=";VAR
12250 PRINT
12260 RETURN
12400 REM DIAGONAL EDGE REGION
12410 FOR K = 2 TO BY - 1
12420 FOR L = K TO K + 1
12430 IF T(K,L) = 0 THEN GOTO 12450
12440 VAR = VAR + 1
12450 NEXT L,K
12470 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12480 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12485 PRINT :FARK = BLA - VAR
12490 FOR K = 2 TO BY - 1
12500 IF (T(K,K) = 0 AND T(K,K + 1) = 0) THEN ERR = ERR + 1
12510 NEXT K
12520 PRINT "UNCOVERED ROWS IN EDGE REGION(MISSING EDGE PNT)=";ERR
12530 PRINT "# OF EDGE POINTS=";BY - 2
12540 PRINT "# OF MISSING EDGE PNT./# OF EDGE PNT=";ERR / (BY - 2)
12550 ERR = ERR + BLA - VAR
12560 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12570 VAR = 0
12580 FOR K = 2 TO BY
12585 FOR L = 1 TO K - 1
12586 IF T(K,L) = 0 THEN GOTO 12590
12588 UZ = K - L:UZ = UZ ^ 2:VAR = VAR + UZ
12590 NEXT L
12591 FOR L = K + 2 TO BY

```

```

12592 IF L > BY THEN GOTO 12596
12594 IF T(K,L) = 0 THEN GOTO 12596
12595 UZ = L - K + 1:UZ = UZ ^ 2:VAR = VAR + UZ
12596 NEXT L
12597 NEXT K
12599 GOTO 12248
12600 REM CIRCULAR SHAPE REGION
12610 FOR K = 1 TO BY
12620 FOR L = 1 TO BY
12630 IF (((K - YC) ^ 2 + (L - YC) ^ 2) < RA ^ 2 AND ((K - YC) ^ 2 + (L - YC) ^ 2) > (RA + 1) ^ 2) THEN GOTO 12640
12633 IF T(K,L) = 0 THEN GOTO 12640
12636 VAR = VAR + 1
12640 NEXT L,K
12660 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12670 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12673 PRINT :FARK = BLA - VAR
12674 EDGE = 0
12680 FOR K = 1 TO BY
12690 FOR L = 1 TO BY
12700 IF (((K - YC) ^ 2 + (L - YC) ^ 2) < RA ^ 2 AND ((K - YC) ^ 2 + (L - YC) ^ 2) > (RA + 1) ^ 2) THEN GOTO 12710
12705 EDGE = EDGE + 1
12710 NEXT L,K
12720 PRINT "# OF EDGE POINTS=";EDGE
12725 PRINT "# OF MISSING EDGE PNT=";EDGE - VAR: PRINT
12730 PRINT "# OF MISSING EDGE PNT/# OF EDGE PNT=";(EDGE - VAR) / EDGE
12740 ERR = EDGE - VAR + FARK
12750 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12760 RETURN

```

```

10 REM ** VERTICAL EDGE-ROSENFELD ALGORITHM: **
20 DIM HR(BY,BY,1),VR(BY,BY,2),BIT(BY,BY),T(BY,BY)
30 PRINT D$;"PR#1"
40 PRINT : PRINT "EDGE DETECTION BY USING ROSENFELD ALGORITHM"
50 PRINT "-----"
60 PRINT "CONTRASTS:D1=";D1
70 PRINT "      D2=";D2
75 IF A = 3 THEN GOTO 100
80 PRINT "      D3=";D3
85 IF A = 2 THEN GOTO 100
90 PRINT "      D4=";D4
100 PRINT : PRINT : PRINT "ORJINAL IMAGE"
110 PRINT "-----"
120 PRINT CHR$(15): POKE 1657,130
130 FOR J = 1 TO BY
140 FOR I = 1 TO BY
180 BIT(J,I) = A(J,I)
190 NEXT I
210 NEXT J
220 GOSUB 7000: REM BIT IMAGE PR.
2000 REM ** ROSENFELD ALGORITHM **
2010 FOR K = 1 TO BY
2020 FOR L = 1 TO BY
2030 VR(K,L,2) = 1
2040 NEXT L,K
2050 KK = 1: GOTO 2070
2060 KK = KK * 2
2070 FOR K = 1 TO BY
2080 FOR L = 1 TO BY
2090 V1 = 0:V2 = 0:H1 = 0:H2 = 0
2100 FOR I = KK TO 1 STEP -1
2110 IF (K + I) > BY THEN 2150
2120 IF (K - I + 1) < 1 THEN 2150
2130 V1 = V1 + A(K + I,L)
2140 V2 = V2 + A(K - I + 1,L)
2150 IF (L + 1) > BY THEN 2190
2160 IF (L - I + 1) < 1 THEN 2190
2170 H1 = H1 + A(K,L + 1)
2180 H2 = H2 + A(K,L - I + 1)
2190 NEXT I
2200 V1 = V1 / KK:V2 = V2 / KK:H1 = H1 / KK:H2 = H2 / KK
2210 VR(K,L,1) = ABS (V1 - V2):VR(K,L,1) = INT (VR(K,L,1))
2220 HR(K,L,1) = ABS (H1 - H2):HR(K,L,1) = INT (HR(K,L,1))
2230 NEXT L,K
2450 PRINT : PRINT
2460 PRINT "EDGE DETECTION-KK=";KK
2470 PRINT "-----"
2475 PRINT CHR$(15)
2480 FOR K = 1 TO BY
2490 FOR L = 1 TO BY
2500 IF HR(K,L,1) > VR(K,L,1) THEN VR(K,L,1) = HR(K,L,1)
2540 BIT(K,L) = VR(K,L,1):VR(K,L,2) = VR(K,L,2) & VR(K,L,1)
2550 NEXT L
2570 NEXT K
2575 IF (KK < > 1 AND KK < > 8 AND KK < > 32) THEN GOTO 2582
2580 GOSUB 7000: REM BITIMAGE
2582 GOTO 4000: REM THRESH.
2585 PRINT : PRINT
2590 PRINT "MULTIPLE EDGE DETECTION"
2595 IF KK = 1 THEN PRINT "KK=1": GOTO 2650
2600 IF KK = 2 THEN PRINT "KK=2#1": GOTO 2650

```

```

2610 IF KK = 4 THEN PRINT "KK=4*2*1": GOTO 2650
2620 IF KK = 8 THEN PRINT "KK=8*4*2*1": GOTO 2650
2630 IF KK = 16 THEN PRINT "KK=16*8*4*2*1": GOTO 2650
2640 IF KK = 32 THEN PRINT "KK=32*16*8*4*2*1"
2650 PRINT "-----"
2655 PRINT CHR$(15)
2660 FOR K = 1 TO BY
2670 FOR L = 1 TO BY
2710 BIT(K,L) = VR(K,L,2)
2720 NEXT L
2725 NEXT K
2727 IF (KK < > 1 AND KK < > 8 AND KK < > 32) THEN GOTO 2735
2730 GOSUB 7000: REM BITIMAGE PRT.
2735 GOTO 4360
4000 REM ** THRESHOLDING **
4010 PRINT : PRINT
4020 PRINT "THRESHOLDING-(ind.differ.)-KK=";KK
4030 PRINT "-----"
4050 THR = 0
4060 FOR K = 1 TO BY
4070 FOR L = 1 TO BY
4080 THR = THR + VR(K,L,1)
4090 NEXT L
4100 NEXT K
4110 THR = THR / (BY * BY)
4120 FOR K = 1 TO BY
4130 FOR L = 1 TO BY
4140 IF VR(K,L,1) > THR THEN VR(K,L,1) = 1: GOTO 4205
4150 VR(K,L,1) = 0
4155 IF (KK < > 1 AND KK < > 8 AND KK < > 32) THEN GOTO 4255
4160 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
4170 FOR Q = 1 TO 8
4180 PRINT CHR$(B(12,Q));
4190 NEXT Q
4200 GOTO 4250
4205 IF (KK < > 1 AND KK < > 8 AND KK < > 32) THEN GOTO 4255
4210 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
4220 FOR Q = 1 TO 8
4230 PRINT CHR$(B(1,Q));
4240 NEXT Q
4250 T(K,L) = VR(K,L,1)
4255 NEXT L
4260 PRINT
4270 NEXT K
4275 GOSUB 12000: REM ANALYSIS
4350 IF KK = 1 THEN GOTO 2060
4355 GOTO 2585
4360 REM MULTIP.DIFF.
4370 PRINT : PRINT
4380 PRINT "THRESHOLDING-(multiple differ.)"

```

```

4390 IF KK = 4 THEN PRINT "MULT.=4*2*1": GOTO 4430
4400 IF KK = 8 THEN PRINT "MULT.=8*4*2*1": GOTO 4430
4410 IF KK = 16 THEN PRINT "MULT.=16*8*4*2*1": GOTO 4430
4420 IF KK = 32 THEN PRINT "MULT.32*16*8*4*2*1"
4430 PRINT "-----"
4440 PRINT
4450 THR = 0
4460 FOR K = 1 TO BY
4470 FOR L = 1 TO BY
4480 THR = THR + VR(K,L,2)
4490 NEXT L,K
4500 THR = THR / (BY * BY)
4510 FOR K = 1 TO BY
4520 FOR L = 1 TO BY
4530 IF VR(K,L,2) > THR THEN VR(K,L,2) = 1: GOTO 4595
4540 VR(K,L,2) = 0
4545 IF (KK < > 1 AND KK < > 8 AND KK < > 32) THEN GOTO 4645
4550 PRINT CHR$(27); "K"; CHR$(8); CHR$(0);
4560 FOR Q = 1 TO 8
4570 PRINT CHR$(B(12,Q));
4580 NEXT Q
4590 GOTO 4640
4595 IF (KK < > 1 AND KK < > 8 AND KK < > 32) THEN GOTO 4645
4600 PRINT CHR$(27); "K"; CHR$(8); CHR$(0);
4610 FOR Q = 1 TO 8
4620 PRINT CHR$(B(1,Q));
4630 NEXT Q
4640 T(K,L) = VR(K,L,2)
4645 NEXT L
4650 PRINT
4660 NEXT K
4665 GOSUB 12000: REM ANALYSIS
4740 IF KK < 32 THEN GOTO 2060
4750 PRINT D$
4760 PRINT D$; "PR#0"
4770 END
7000 REM *** BITIMAGE PRINTING ***
7010 MAX = BIT(1,1)
7020 FOR K = 1 TO BY
7030 FOR L = 1 TO BY
7040 IF BIT(K,L) > MAX THEN MAX = BIT(K,L)
7050 NEXT L,K
7070 MULT = 400 / MAX
7090 FOR K = 1 TO BY
7100 FOR L = 1 TO BY
7110 BIT(K,L) = INT (BIT(K,L) * MULT)
7120 NEXT L,K
7250 PRINT CHR$(18)
7260 FOR J = 1 TO BY
7270 FOR I = 1 TO BY
7280 IF BIT(J,I) > 280 THEN 8010
7290 IF BIT(J,I) > 240 THEN 8060
7300 IF BIT(J,I) > 210 THEN 8120
7310 IF BIT(J,I) > 190 THEN 8180
7320 IF BIT(J,I) > 175 THEN 8240
7330 IF BIT(J,I) > 160 THEN 8300
7340 IF BIT(J,I) > 135 THEN 8360
7350 IF BIT(J,I) > 110 THEN 8420
7360 IF BIT(J,I) > 80 THEN 8480
7370 IF BIT(J,I) > 50 THEN 8540
7380 IF BIT(J,I) > 40 THEN 8600
7390 IF BIT(J,I) > 20 THEN 8660

```



```
7400 IF BIT(J,1) > 10 THEN 8720
7410 GOTO 8780
7420 NEXT I
7430 PRINT
7440 NEXT J
7450 RETURN
8000 REM
8010 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8020 FOR Q = 1 TO 8
8030 PRINT CHR$(B(1,Q));
8040 NEXT Q
8050 GOTO 7420
8060 REM
8070 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8080 FOR Q = 1 TO 8
8090 PRINT CHR$(B(2,Q));
8100 NEXT Q
8110 GOTO 7420
8120 REM
8130 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8140 FOR Q = 1 TO 8
8150 PRINT CHR$(B(3,Q));
8160 NEXT Q
8170 GOTO 7420
8180 REM
8190 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8200 FOR Q = 1 TO 8
8210 PRINT CHR$(B(4,Q));
8220 NEXT Q
8230 GOTO 7420
8240 REM
8250 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8260 FOR Q = 1 TO 8
8270 PRINT CHR$(B(5,Q));
8280 NEXT Q
8290 GOTO 7420
8300 REM
8310 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8320 FOR Q = 1 TO 8
8330 PRINT CHR$(B(6,Q));
8340 NEXT Q
8350 GOTO 7420
8360 REM
8370 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8380 FOR Q = 1 TO 8
8390 PRINT CHR$(B(7,Q));
8400 NEXT Q
8410 GOTO 7420
8420 REM
8430 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8440 FOR Q = 1 TO 8
8450 PRINT CHR$(B(8,Q));
8460 NEXT Q
8470 GOTO 7420
```

```

8480 REM
8490 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8500 FOR Q = 1 TO 8
8510 PRINT CHR$(B(9,Q));
8520 NEXT Q
8530 GOTO 7420
8540 REM
8550 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8560 FOR Q = 1 TO 8
8570 PRINT CHR$(B(10,Q));
8580 NEXT Q
8590 GOTO 7420
8600 REM
8610 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8620 FOR Q = 1 TO 8
8630 PRINT CHR$(B(11,Q));
8640 NEXT Q
8650 GOTO 7420
8660 REM
8670 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8680 FOR Q = 1 TO 8
8690 PRINT CHR$(B(12,Q));
8700 NEXT Q
8710 GOTO 7420
8720 REM
8730 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8740 FOR Q = 1 TO 8
8750 PRINT CHR$(B(13,Q));
8760 NEXT Q
8770 GOTO 7420
8780 REM
8790 PRINT CHR$(27);"K"; CHR$(8); CHR$(0);
8800 FOR Q = 1 TO 8
8810 PRINT CHR$(B(14,Q));
8820 NEXT Q
8830 GOTO 7420
12000 REM ** Z ERROR S/R **
12005 PRINT : PRINT
12010 BLA = 0:VAR = 0:ERR = 0
12015 FOR K = 1 TO BY
12020 FOR L = 1 TO BY
12025 IF T(K,L) = 0 THEN GOTO 12035
12030 BLA = BLA + 1
12035 NEXT L,K
12040 ON A GOTO 12045,12400,12600
12045 REM VERTICAL EDGE REGION
12050 FOR K = 2 TO BY - 1
12055 FOR L = BY / 2 TO BY / 2 + 1
12060 IF T(K,L) = 0 THEN GOTO 12070
12065 VAR = VAR + 1
12070 NEXT L,K
12075 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12080 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12085 PRINT :FARK = BLA - VAR
12090 FOR K = 2 TO BY - 1
12095 IF (T(K,BY / 2) = 0 AND T(K,BY / 2 + 1) = 0) THEN ERR = ERR + 1
12100 NEXT K
12105 PRINT "UNCOVERED ROWS IN EDGE REGION(MISSING EDGE PNT)=";ERR

```

```

12110 PRINT "# OF EDGE POINTS=";BY - 2
12115 PRINT "# OF MISSING EDGE PNT./# OF EDGE PNT=";ERR / (BY - 2)
12120 ERR = ERR + BLA - VAR
12125 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12130 VAR = 0
12135 FOR K = 1 TO BY
12140 FOR L = 1 TO BY / 2 - 1
12145 IF T(K,L) = 0 THEN GOTO 12155
12150 UZ = BY / 2 - 2 - L:UZ = UZ ^ 2:VAR = VAR + UZ
12155 NEXT L
12160 FOR L = BY / 2 + 2 TO BY
12165 IF T(K,L) = 0 THEN GOTO 12180
12170 UZ = L - BY / 2 + 1:UZ = UZ ^ 2:VAR = VAR + UZ
12175 NEXT L
12180 NEXT K
12185 IF FARK = 0 THEN GOTO 12200
12190 VAR = VAR / FARK
12195 PRINT "MEAN SQUARE DISTANCE OF ERROR=";VAR
12200 PRINT
12205 RETURN
12400 REM DIAGONAL EDGE REGION
12405 FOR K = 2 TO BY - 1
12410 FOR L = K TO K + 1
12415 IF T(K,L) = 0 THEN GOTO 12425
12420 VAR = VAR + 1
12425 NEXT L,K
12430 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12435 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12440 PRINT :FARK = BLA - VAR
12445 FOR K = 2 TO BY - 1
12450 IF (T(K,K) = 0 AND T(K,K + 1) = 0) THEN ERR = ERR + 1
12455 NEXT K
12460 PRINT "UNCOVERED ROWS IN EDGE REGION(MISSING EDGE PNT)=";ERR
12465 PRINT "# OF EDGE POINTS=";BY - 2
12470 PRINT "# OF MISSING EDGE PNT./# OF EDGE PNT=";ERR / (BY - 2)
12475 ERR = ERR + BLA - VAR
12480 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12485 VAR = 0
12490 FOR K = 2 TO BY
12495 FOR L = 1 TO K - 1
12500 IF T(K,L) = 0 THEN GOTO 12510
12505 UZ = K - L:UZ = UZ ^ 2:VAR = VAR + UZ
12510 NEXT L
12515 FOR L = K + 2 TO BY
12520 IF L > BY THEN GOTO 12535
12525 IF T(K,L) = 0 THEN GOTO 12535
12530 UZ = L - K + 1:UZ = UZ ^ 2:VAR = VAR + UZ
12535 NEXT L
12540 NEXT K
12545 GOTO 12190
12600 REM CIRCULAR SHAPE REGION
12610 FOR K = 1 TO BY

```

```

12620 FOR L = 1 TO BY
12630 IF (((K - XC) ^ 2 + (L - YC) ^ 2) > = RA ^ 2 AND ((K - XC) ^ 2 + (L - YC) ^ 2) < = (RA + 1) ^ 2) THEN GOTO 12660
12640 IF T(K,L) = 0 THEN GOTO 12660
12650 VAR = VAR + 1
12660 NEXT L,K
12670 PRINT "NUMBER OF 1'S IN THE WHOLE PICTURE=";BLA
12680 PRINT "NUMBER OF 1'S IN THE EDGE REGION =" ;VAR
12690 PRINT ;FARK = BLA - VAR
12700 EDGE = 0
12710 FOR K = 1 TO BY
12720 FOR L = 1 TO BY
12730 IF (((K - XC) ^ 2 + (L - YC) ^ 2) > = RA ^ 2 AND ((K - XC) ^ 2 + (L - YC) ^ 2) < = (RA + 1) ^ 2) THEN GOTO 12750
12740 EDGE = EDGE + 1
12750 NEXT L,K
12760 PRINT "# OF EDGE POINTS=";EDGE
12770 PRINT "# OF MISSING EDGE PNT=";EDGE - VAR: PRINT
12780 PRINT "# OF MISSING EDGE PNT/# OF EDGE PNT=";(EDGE - VAR) / EDGE
12790 ERR = EDGE - VAR + FARK
12800 PRINT "# OF ERROR/WHOLE PICTURE=";ERR / (BY * BY)
12810 RETURN

```

BIBLIOGRAPHY

- [1] - A.K.Jain, "Advances in Mathematical Models for Image Processing", Proc.of the IEEE, Vol.69, No.5, pp502-528, May 1981
- [2] - W.F.Schreiber, "Picture Coding", Proceedings of the IEEE, Vol.55, No.3, pp320-330, March 1967
- [3] - A.Habibi and G.S.Robinson, "A Survey of Digital Picture Coding", IEEE Computer, Vol.7, No.5, pp22-34, May 1974
- [4] - T.S.Huang, W.F.Schreiber, O.J.Tretiak, "Image Processing", Proceed.of IEEE, Vol.59, No.11, pp1586-1609, November 1971
- [5] - S.Jeyat, "Waveform Quantization and Coding", IEEE Press, 1976
IEEE Press, Bell Laboratories, 1976
- [6] - A.Habibi, P.A.Wintz, "Image Coding by Linear Transformation and Block Quantization", IEEE Trans.on Commun.Tech., Vol Com-19, No.1, pp50-62, February 1971
- [7] - W.D.Montgomery, "Reconstruction of Pictures from Scanned Records" IEEE Trans.on Inform.Theory, Vol IT-11, pp204-206, April 1965
- [8] - D.P.Peterson and D.Middleton, "Sampling and Reconstruction of Wavenumber-Limited Functions in N-Dimensional Euclidian Spaces" Information and Control, Vol.5, pp279-323, December 1962
- [9] - A.K.Jain, "Multidimensional Techniques in Digital Image Processing"
- [10]- W.K.Pratt, "Digital Image Processing", New York, Wiley, 1978
- [11]- A.K.Jain, S.Ranganath, "Image Coding by Autoregressive Synthesis" Proc.IEEE ICASSP 80, pp 770-773, April 1980
- [12]- N.Ahmed, T.Naterajan, K.R.Rao, "Discrete Cosine Transform" IEEE Trans.Coputers, Vol.C-23, pp90-93, January 1974
- [13]- M.Hamidi, J.Pearl, "Comparison of the Cosine and Fourier Transforms of Markov-1 Signals", IEEE Trans.ASSP, Vol.ASSP-24, pp428-429, October 1976
- [14]- N.Ahmed, K.R.Rao, "Orthogonal Transforms for Digital Signal Processing", New York, Springer Verlag, 1975
- [15]- A.K.Jain, "A Sinusoidal Family of Unitary Transforms", IEEE Trans.Pattern Anal.Mach.Intelligence, Vol.PAMI-1, pp356-365, October 1979
- [16]- H.P.Kramer, M.V.Hathews, "A Linear Coding for Transmitting a Set of Correlated Signals", IRE Trans.Information Theory Vol.IT-2, pp41-46, September 1956

- [17]- J.R.Jain, A.K.Jain, "Interframe Adaptive Data Compression Techniques for Images", Tech.Rep.SIPL-79-2, Signal and Image Processing Lab.Dep.Elec.Computer.Eng.Univ.California Davis, August 1979
- [18]- A.K.Jain, "Image Data Compression - A Review", Proc.IEEE, Vol.69, pp349-389, March 1981
- [19]- A.K.Jain, S.Ranganath, "Image Coding by Autoregressive Synthesis" Proc.IEEE, ICASSP'80 (Denver,Co), pp 770-773, April 1980
- [20]- N.E.Nahi, T.Assefi, "Bayesian Recursive Image Estimation" IEEE Trans.Comput.(Short Notes), Vol.C-21, pp734-738, July 1972
- [21]- A.K.Jain, "Noncausal Representations for Finite Discrete Signals" in Proc.IEEE Conj.Decision and Control, 1974
- [22]- A.K.Jain, J.R.Jain, "Partial Differential Equations and Finite Difference Methods in Image Processing,Part 2;Image Restoration" IEEE Trans.Automat.Control, Vol.AC-23, pp817-834, October 1978
- [23]- A.K.Jain, "A Fast Karhunen Loewe Transform for a Class of Random Processes", IEEE Trans.Commun., Vol.COM-24,pp1023-1029,Sept 1976
- [24]- A.Habibi, "Two-Dimensional Bayesian Estimate of Images" Proc.IEEE, Vol.60, pp878-883, July 1972
- [25]- M.Strintzis, "Comments on Two-Dimensional Bayesian Estimate of Images", Proc.IEEE, Vol.64, pp1255-1257, August 1976
- [26]- N.E.Nahi,A.Habibi,"Decision Directed Recursive Image Enhancement" IEEE Trans.Circuits Syst., Vol CAS-22, pp286-293, March 1975
- [27]- J.W.Wodds, Ch.Radewan, "Kalman Filtering in Two Dimensions" IEEE Trans.Inform.Theory, Vol IT-23, pp473-482, July 1977
- [28]- A.K.Jain, "A Semicausal Model for Recursive Filtering of Two-Dimensional Images", IEEE Trans.ASSP, Vol.ASSP-26, pp121-126, October 1978
- [29]- B.R.Hunt,"The Application of Constained Least Squares Estimation to Image Restoration by Digital Computers" Vol.C-22, pp805-812, September 1973
- [30]- C.Cherry,M.H.Kubba,D.E.Pearson,P.Barkes, "An Experimental Study of the Possible Bandwith Compression of Visual Image Signals" Proc.IEEE, Vol 51, pp1507-1517, 1963
- [31]- G.G.Gouriet, "Bandwith Compression of Television Signals" IEEE Proc.1043,Pt.8, pp256-272, 1957
- [32]- A.H.Robinson, C.Cherry, "Results of Prototype Television Bandwith Compression Schemes", Proc.IEEE,Vol.55,pp356-364,1967
- [33]- D.N.Graham,"Image Transmission by Two-Dimensional Contour Coding"

Proc.IEEE, Vol.55, pp336-346, 1967

- [34]- L.G.Roberts, "Picture Coding Using Pseudo-Random Noise"
IRE Trans.Inform.Theory, Vol.IT-8, pp145-154, 1962
- [35]- F.W.Maunts, "A Video Encoding System Using Conditional
Picture-Element Replenishment", Bell Syst.Tech.Journal, Vol.48,
No.7, pp2545-2555, September 1969
- [36]- B.G.Haskell, F.W.Maunts, T.C.Candy, "Interface Coding of Video
Telephone Pictures", Proc.IEEE, Vol.60, No.7, pp792-800, July 1972
- [37]- B.G.Haskell, "Buffer and Channel Shaping by Several Interframe
Picture Phone Coders", Bell Syst.Tech.Journal, Vol.51, No.1,
pp261-291, January 1972
- [38]- W.K.Pratt, "Image Transmission Techniques", New York, Wiley, 1977
- [39]- T.Huang, "Picture Bandwidth Compression"
- [40]- A.Rosenfeld, A.C.Kale, "Digital Picture Processing"
Academic Press, New York, 1976
- [41]- A.G.Constantinides, V.Cappellini, "Signal Processing", 1980
- [42]- Proceedings, "Two-Dimensional Digital Signal Processing
Conference", Columbia, Missouri, October 6-8, 1971
- [43]- Taub and Schilling, "Principles of Communication Systems"
- [44]- A.Rosenfeld, "Iterative Methods in Image Analysis",
Pattern Recognition, Vol.10, pp181-187, January 1978
- [45]- C.A.Andrew, J.M.Davies, G.R.Schwarz, "Adaptive Data Compression"
Proc.IEEE, Vol.55, No.3, pp267-277, March 1967
- [46]- T.G.Stockham, "Image Processing in the Context of a Visual Model"
Proc.IEEE, Vol.60, No.7, pp828-842, July 1972
- [47]- B.Smith, "Instantaneous companding of quantized signals"
Bell syst.Tech.J., pp653-709, 1957
- [48]- J.Max, "Quantizing for minimum distortion"
IRE Trans.on Inf.Theory, Vol IT-6, pp7-12, 1960
- [49]- M.D.Paez, T.H.Glisson, "Minimum Mean Square Error Quantization
in Speech PCM and DPCM Systems"
IEEE Trans.on Commun., Com-20, pp225-230, April 1972
- [50]- P.Noll, "Adaptive Quantization in Speech Coding Systems"
IEEE Proceedings, March 1974
- [51]- N.S.Jayant, "Adaptive Quantization with a one-word Memory"
Bell Syst.Tech.J., pp1119-1144, Sept.1973

- [52]- Allen Gersho, "Quantization"
IEEE Commun.Society Magazin, pp16-29, Sept.1977
- [53]- T.Berger, "Rate Distortion Theory", Englewood Cliffs, NS
Prentice-Hall, 1971
- [54]- H.P.Kramer and M.W.Mathews, "A linear coding for transmitting
a set of correlated signals"
IRE Trans.on Inf.Theory, Vol IT-2, pp41-46, Sept.1956
- [55]- J.J.Y.Huang and P.M.Schultheiss, "Block quantization of
correlated gaussian random variables"
IEEE Trans.on Commun.Systems, Vol CS-11, pp289-296, Sept.1963
- [56]- A.Habibi and P.A.Wintz, "Image coding by linear transformations
and Block Quantization"
IEEE Trans.on Commun.Tech., Vol COM-19, pp50-62, Febr.1971
- [57]- N.S.Jayant, "Digital Coding of Speech Waveforms:PCM, DPCM
and DM Quantizers"
IEEE Proccedings, Vol.62,No 5,pp611-632, May 1974
- [58]- T.S.Huang, O.J.Tretiak, B.Prasada, Y.Yamaguchi,
"Design considerations in PCM Transmission of low-resolution
monochrome still pictures"
IEEE Proccedings, Vol.55,No 3,pp331-335, March 1967
- [59]- T.S.Huang, "PCM Picture Transmission"
IEEE Spectrum, Vol.2,pp57-63, December 1965
- [60]- D.P.Peterson and D.Middleton, "Sampling and reconstruction of
wave-number-limited functions in n-dimensional Euclidean spaces"
Information and Control, Vol.5, pp279-323, 1962
- [61]- T.S.Huang and O.J.Tretiak, "Research in Picture Processing"
in optical and electro-optical Inf.Processing, T.Tippett,Ed.
Cambridge Mass. M.I.T.Press, Ch:3, 1965
- [62]- L.G.Roberts, "Picture coding using pseudo-random noise"
IRE Trans.on Inform.Theory, Vol IT-8, pp145-154, Febr 1962
- [63]- S.Deutsch, "Narrow band TV uses pseudo random scan"
Electronics, Vol.35, pp49-51, 27 April 1962
- [64]- I.T Young and J.C.Matt-Smith, "On weighted PCM"
IEEE Trans.on Inform.Theory, Vol IT-11, pp596-597, October 1965
- [65]- P.Noll, "A Comparative study of various schemes for
speech encoding"
Bell System Tech. J., Vol.54, No.9, pp1597-1614, November 1975
- [66]- A.J.Seyler, "Statistics of Television Frame differences"
Proc.IEEE (Correspondance), Vol.53, pp2127-2128, December 1965
- [67]- J.B.O'Neal, "Delta Modulation quantizing noise analytical and

computer simulation results for Gaussian and TV input signals"
Bell System Tech.J., Vol.45, pp117-142, January 1965

- [68]- B.M.Oliver, J.R.Pierce, C.E.Shannon, "The Philosophy of PCM"
Waveform Quantization and Coding by Jayant,
IEEE Press, pp8-15, 1976
- [69]- L.S.Golding and P.M.Schultheiss, "Study of an Adaptive
Quantizer"
Proceedings of the IEEE, Vol.55, No.3, pp293-297, March 1967
- [70]- B.Girod, "Adaptive Prediction for DPCM Coding of TV Signals"
IEEE Trans.on Acc.,Speech and Signal Processing, Vol.ASSP-29,
No.6, pp1142-1147, December 1981
- [71]- W.D.Montgomery, "Reconstruction of pictures from
scanned records", IEEE Trans.on Information Theory,
Vol.IT-11, pp204-206, April 1965
- [72]- D.P.Peterson, D. Middleton, "Sampling and Reconstruction of
Wavenumber-limited Functions in N-dimensional Euclidian Spaces"
Information and Control, Vol.5, pp279-323, December 1962
- [73]- W.F.Schreiber, "The Mathematical Foundation of the Synthetic
Highs System", Research Lab.of Electronics, M.I.T., Cambridge,
Mass., Quarterly Progress Report 68, pp140, January 1963
- [74]- E.Argyle, "Techniques for Edge Detection",
Proc.IEEE, Vol.59, No.2, pp285-287, February 1971
- [75]- I.D.G.Macload, "Comments on Techniques for Edge Detection"
Proc.IEEE, Vol.60, No.3, page 344, March 1972
- [76]- M.Hueckel, "A Local Visual Operator which recognizes edges
and lines", J.Assoc.Computer Mach., Vol.20, No.4, pp634-647,
October 1973
- [77]- L.G.Roberts, "Machine Perception of Three- Dimensional Solids"
in optical and Elec.Opt.Inf.Process, M.I.T Press,
Cambridge, Mass, pp159-197, 1965
- [78]- R.O.Duda and P.E.Hart, "Pattern Classification and
Scene Analysis" Wiley, New York, page 271, 1973
- [79]- R.Kirsch, "Computer Determination of the Constituent Structure
of Biological Images", Computers and Biomedical Research
Vol.4, No.3, pp315-328, 1971
- [80]- J.M.S.Prewitt, "Object Enhancement and Extraction"
in Picture Processing and Psychopictorics, B.S.Lipkin and A.
Rosenfeld, eds., Academic Press, New York, pp75-149, 1970
- [81]- A.Rosenfeld, "A Nonlinear Edge Detection Technique",
Proc.IEEE Letters, Vol.58, No.5, pp814-816, May 1970

- [82]- A.Rosenfeld, "Picture Processing by Computer", Academic Press, New York, pp94-103, 1969
- [83]- A.Rosenfeld and M.Thurston, "Edge and Curve Detection for Visual Scene Analysis", IEEE Trans.on Computers, Vol.C-20, No.5, pp562-567, May 1971
- [84]- Guner S.Robinson, "Detection and Coding of Edges using Directional Masks", Proceedings of the Society of Photo-optical Inst.Engineers, Vol.87, Advances in Image Transmission Techniques, pp117-125, 1976
- [85]- Guner S.Robinson, "Color Edge Detection", Proc. of the Society of Photo-Optical Instrumentation Engineers Vol.87, Advances in Image Trans.Techn., pp126-132, 1976
- [86]- J.R.Fram and E.S.Deutsch, "On the Quantitative Evaluation of Edge Detection Schemes and Their Comparison with Human Performance" IEEE Trans.on Computers, Vol.C-24, No.6, pp616-628, June 1975
- [87]- E.Alpaslan, "Yönlü Laplace Değişim Maskeleriyle görüntü kenarlarının bulunması" TÜBİTAK Elek.Araşt.Ün., Ünite içi Rapor No.UI-79/05, May 1979
- [88]- J.R.Fram and E.S.Deutsch, "A quantitative study of the orientation bias of some Edge Detector Schemes", IEEE Trans. Computers, Vol.C-27, NO.3, pp205-213, March 1978
- [89]- A.Herskovits and T.O.Binford, "On Boundary Detection" M.I.T.Project MAC Artificial Intell. Memo, page 183, July 1970
- [90]- T.G.Newman and H.Dirilten, "A nonlinear transformation for Digital Picture Processing" IEEE Trans.Computer, Corres., pp869-873, 1973
- [91]- M.Kunt, "Edge Detection: A Tutorial Review" 1982 Intern.Confer.on Acoust.Speech and Sig.Proc., May 1982
- [92]- M.Kunt, "Image Data Compression by Contour Texture Modelling" Sec.3, Segmentation and Coding, Application of Digital Image Processing, Vol.397, pp131-139
- [93]- Carol S.Clark, Anthony L.Luk, Charles A.McNary "Feature-Based Scene Analysis and Model Matching" Hughes Research Laboratories, pp251-276
- [94]- J.C.Dunn, "Group Avaraged Linear Transforms that Detect Corner and Edges", IEEE Trans.on Computers, Vol.C-24, No.12, Dec. 1975
- [95]- E.Alpaslan and Fuat İnce, "Image Enhancement by Local Histogram Stretching", TÜBİTAK, Elektronik Araştırma Ünitesi, Technical Report No.2, April 1979