The Application of Edge Detection and Edge Tracking Algorithms to the Determination of Tree Ring Properties

by

Sean Kalynuk

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba

ISBN 0-315-85906-7

Canada

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Computer Science
SUBJECT TERM

0984
SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Nom _____

Dissertation Abstracts International est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.

_____ ☐☐☐ U·M·I

SUJET — CODE DE SUJET

Catégories par sujets

HUMANITÉS ET SCIENCES SOCIALES

COMMUNICATIONS ET LES ARTS
Architecture 0729
Beaux-arts 0357
Bibliothéconomie 0399
Cinéma 0900
Communication verbale 0459
Communications 0708
Danse 0378
Histoire de l'art 0377
Journalisme 0391
Musique 0413
Sciences de l'information 0723
Théâtre 0465

ÉDUCATION
Généralités 515
Administration 0514
Art 0273
Collèges communautaires 0275
Commerce 0688
Économie domestique 0278
Éducation permanente 0516
Éducation préscolaire 0518
Éducation sanitaire 0680
Enseignement agricole 0517
Enseignement bilingue et multiculturel 0282
Enseignement industriel 0521
Enseignement primaire. 0524
Enseignement professionnel 0747
Enseignement religieux 0527
Enseignement secondaire 0533
Enseignement spécial 0529
Enseignement supérieur 0745
Évaluation 0288
Finances 0277
Formation des enseignants 0530
Histoire de l'éducation 0520
Langues et littérature 0279
Lecture 0535
Mathématiques 0280
Musique 0522
Orientation et consultation 0519
Philosophie de l'éducation 0998
Physique 0523
Programmes d'études et enseignement 0727
Psychologie 0525
Sciences 0714
Sciences sociales 0534
Sociologie de l'éducation 0340
Technologie 0710

LANGUE, LITTÉRATURE ET LINGUISTIQUE
Langues
- Généralités 0679
- Anciennes 0289
- Linguistique 0290
- Modernes 0291

Littérature
- Généralités 0401
- Anciennes 0294
- Comparée 0295
- Médiévale 0297
- Moderne 0298
- Africaine 0316
- Américaine 0591
- Anglaise 0593
- Asiatique 0305
- Canadienne (Anglaise) 0352
- Canadienne (Française) 0355
- Germanique 0311
- Latino-américaine 0312
- Moyen-orientale 0315
- Romane 0313
- Slave et est-européenne 0314

PHILOSOPHIE, RELIGION ET THÉOLOGIE
Philosophie 0422
Religion
- Généralités 0318
- Clergé 0319
- Études bibliques 0321
- Histoire des religions 0320
- Philosophie de la religion 0322

Théologie 0469

SCIENCES SOCIALES
Anthropologie
- Archéologie 0324
- Culturelle 0326
- Physique 0327

Droit 0398
Économie
- Généralités 0501
- Commerce-Affaires 0505
- Économie agricole 0503
- Économie du travail 0510
- Finances 0508
- Histoire 0509
- Théorie 0511

Études américaines 0323
Études canadiennes 0385
Études féministes 0453
Folklore 0358
Géographie 0366
Gérontologie 0351
Gestion des affaires
- Généralités 0310
- Administration 0454
- Banques 0770
- Comptabilité 0272
- Marketing 0338

Histoire
- Histoire générale 0578
- Ancienne 0579
- Médiévale 0581
- Moderne 0582
- Histoire des noirs 0328
- Africaine 0331
- Canadienne 0334
- États-Unis 0337
- Européenne 0335
- Moyen-orientale 0333
- Latino-américaine 0336
- Asie, Australie et Océanie 0332

Histoire des sciences 0585
Loisirs 0814
Planification urbaine et régionale 0999
Science politique
- Généralités 0615
- Administration publique 0617
- Droit et relations internationales 0616

Sociologie
- Généralités 0626
- Aide et bien-être social 0630
- Criminologie et établissements pénitentiaires 0627
- Démographie 0938
- Études de l' individu et de la famille 0628
- Études des relations interethniques et des relations raciales 0631
- Structure et développement social 0700
- Théorie et méthodes. 0344
- Travail et relations industrielles 0629

Transports 0709
Travail social 0452

SCIENCES ET INGÉNIERIE

SCIENCES BIOLOGIQUES
Agriculture
- Généralités 0473
- Agronomie. 0285
- Alimentation et technologie alimentaire 0359
- Culture 0479
- Élevage et alimentation 0475
- Exploitation des pâturages 0777
- Pathologie animale 0476
- Pathologie végétale 0480
- Physiologie végétale 0817
- Sylviculture et faune 0478
- Technologie du bois 0746

Biologie
- Généralités 0306
- Anatomie 0287
- Biologie (Statistiques) 0308
- Biologie moléculaire 0307
- Botanique 0309
- Cellule 0379
- Écologie 0329
- Entomologie 0353
- Génétique 0369
- Limnologie 0793
- Microbiologie 0410
- Neurologie 0317
- Océanographie 0416
- Physiologie 0433
- Radiation 0821
- Science vétérinaire 0778
- Zoologie 0472

Biophysique
- Généralités 0786
- Medicale 0760

SCIENCES DE LA TERRE
Biogéochimie 0425
Géochimie 0996
Géodésie 0370
Géographie physique 0368
Géologie 0372
Géophysique 0373
Hydrologie 0388
Minéralogie 0411
Océanographie physique 0415
Paléobotanique 0345
Paléoécologie 0426
Paléontologie 0418
Paléozoologie 0985
Palynologie 0427

SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT
Économie domestique 0386
Sciences de l'environnement 0768
Sciences de la santé
- Généralités 0566
- Administration des hôpitaux 0769
- Alimentation et nutrition 0570
- Audiologie 0300
- Chimiothérapie 0992
- Dentisterie 0567
- Développement humain 0758
- Enseignement 0350
- Immunologie 0982
- Loisirs 0575
- Médecine du travail et thérapie 0354
- Médecine et chirurgie 0564
- Obstétrique et gynécologie 0380
- Ophtalmologie 0381
- Orthophonie 0460
- Pathologie 0571
- Pharmacie 0572
- Pharmacologie 0419
- Physiothérapie 0382
- Radiologie 0574
- Santé mentale 0347
- Santé publique 0573
- Soins infirmiers 0569
- Toxicologie 0383

SCIENCES PHYSIQUES
Sciences Pures
Chimie
- Généralités 0485
- Biochimie 487
- Chimie agricole 0749
- Chimie analytique 0486
- Chimie minérale 0488
- Chimie nucléaire 0738
- Chimie organique 0490
- Chimie pharmaceutique 0491
- Physique 0494
- PolymÈres 0495
- Radiation 0754

Mathématiques 0405
Physique
- Généralités 0605
- Acoustique 0986
- Astronomie et astrophysique 0606
- Électronique et électricité 0607
- Fluides et plasma 0759
- Météorologie 0608
- Optique 0752
- Particules (Physique nucléaire) 0798
- Physique atomique 0748
- Physique de l'état solide 0611
- Physique moléculaire 0609
- Physique nucléaire 0610
- Radiation 0756

Statistiques 0463

Sciences Appliqués Et Technologie
Informatique 0984
Ingénierie
- Généralités 0537
- Agricole 0539
- Automobile 0540
- Biomédicale 0541
- Chaleur et thermodynamique 0348
- Conditionnement (Emballage) 0549
- Génie aérospatial 0538
- Génie chimique 0542
- Génie civil 0543
- Génie électronique et électrique 0544
- Génie industriel 0546
- Génie mécanique 0548
- Génie nucléaire 0552
- Ingénierie des systèmes 0790
- Mécanique navale 0547
- Métallurgie 0743
- Science des matériaux 0794
- Technique du pétrole 0765
- Technique minière 0551
- Techniques sanitaires et municipales 0554
- Technologie hydraulique 0545

Mécanique appliquée 0346
Géotechnologie 0428
Matières plastiques (Technologie) 0795
Recherche opérationnelle 0796
Textiles et tissus (Technologie) 0794

PSYCHOLOGIE
Généralités 0621
Personnalité 0625
Psychobiologie 0349
Psychologie clinique 0622
Psychologie du comportement 0384
Psychologie du développement 0620
Psychologie expérimentale 0623
Psychologie industrielle 0624
Psychologie physiologique 0989
Psychologie sociale 0451
Psychométrie 0632

THE APPLICATION OF EDGE DETECTION AND EDGE TRACKING ALGORITHMS TO THE DETERMINATION OF TREE RING PROPERTIES

BY

SEAN KALYNUK

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Acknowledgments

I would first like to express thanks to my supervisor Dr. Janet Hoskins for her support and guidance. Thank you to all my fellow graduate students for keeping graduate life interesting. I would also like to express my gratitude to Tom Dubinski and Gilbert Detillieux for their technical assistance.

Abstract

To assist in obtaining properties of the rings in a digital image of a tree cross section, edge detection can be employed. Various edge detection methods are tested on a sample tree cross section to determine which one can provide sufficient detection of the ring edges. Noise reduction and edge enhancement algorithms are examined for their usefulness in aiding the edge detection process.

Algorithms for obtaining ring area, and a count of the number of rings, are presented. The computation of ring area is achieved by obtaining spline approximations to the rings. This is accomplished by using an edge tracking algorithm to track each individual ring. Three different methods of edge tracking are examined for this purpose. A case study on the sample cross section is provided to demonstrate the procedure for obtaining the spline approximations to the rings.

Contents

Chapter 1
Introduction

1.1 Background

Image analysis techniques provide a very powerful way to obtain detailed information from a digital image. These techniques can range from simple feature enhancement to complex object recognition. A very important and widely studied technique is *edge detection*. This is an area concerned with locating edges that occur within a digital image. The research presented here deals with edge detection and its use in locating tree rings within an image of a tree cross section.

The ring structure of a tree is important to dendochronologists who use it to gauge tree age and growth patterns, as well as to determine past atmospheric conditions. The ability to analyze the ring structure with a computer introduces a certain level of automation in the process. Therefore, the goal of this study was to develop a strategy for extracting the tree ring contours from a cross sectional image so that they could be subjected to an algorithmic analysis. Although the actual analysis was not an integral part of the study, two useful applications are presented. One application concerns ring counting, and the other determines the ring area.

The procedure of extracting the tree rings requires three basic steps: noise reduction, edge detection and edge tracking. Noise reduction is used to improve image quality and reduce errors in the edge detection step. Edge detection is used to reduce the information in

the image such that only the ring edges remain. In certain cases, noise reduction occurs at the same time as edge detection because it is incorporated into the edge detector itself. Edge tracking is used on the edge detected image to obtain interpolating functions of the rings that can then be used for mathematical analysis.

1.2 Preliminary Discussion

Throughout each chapter, a sample tree cross section, known as a *biscuit*, is used to demonstrate the effects of the various image processing techniques applied. The biscuit, shown in Fig. 1.1, is from a Jack Pine that had a large mistletoe growth. The accelerated growth can be seen in the larger spacing of the rings. The focus was upon the main trunk rather than the mistletoe because the mistletoe has few complete rings. Most of the work was done on a grayscale image of the biscuit. In such an image, each pixel can take on one of 256 different intensities of gray. An 8 bit colour image (256 colours) was also used where necessary. The full colour image was not used due to a lack of proper hardware.

In a grayscale image, an *edge* corresponds to changes in gray level. These changes can take several forms [ROSE2]:

a) A border between two adjacent regions: the gray levels differ between the two regions, but are consistent within a region. The cross section of an abrupt grayscale change resembles a step and is therefore called a *step edge* (Fig. 1.2(a)). If the grayscale change occurs over a wider area, it is referred to as a *ramp edge* (Fig. 1.2(b).

b) A line or a curve: the gray levels are relatively constant within an area except for a thin strip which differs significantly. Such an edge is called a *spike edge* (Fig. 1.2(c)). If the grayscale change occurs over a wider area, it is called a *roof edge* (Fig. 1.2(d)).

c) A spot: the gray levels are relatively constant except for a small local area that differs significantly. The cross section of such an area will also resemble a spike or a roof.

Fig. 1.1 The sample biscuit scanned at 600 dpi. The top portion is the mistletoe growth.

In a colour image, an edge can occur because of an intensity change or a difference in hue. Edges can also occur between regions of differing patterns (Fig. 1.2(e)), but such situations are not considered here.

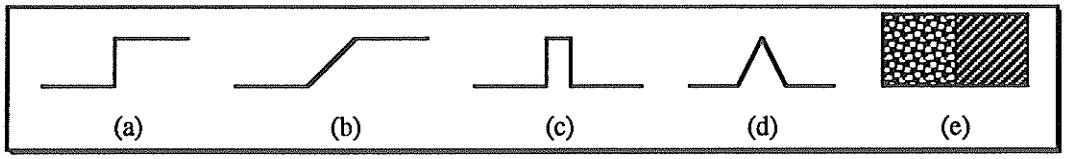


Fig. 1.2 a) step edge b) ramp edge c) spike edge d) roof edge e) Edge between regions of differing patterns.

In the grayscale biscuit image, the rings have a roof edge appearance. They are actually inverted roof edges because their intensity is lower than their surroundings. In a cross section of the rings, they appear as troughs. Fig. 1.3 shows an example cross section of three rings.

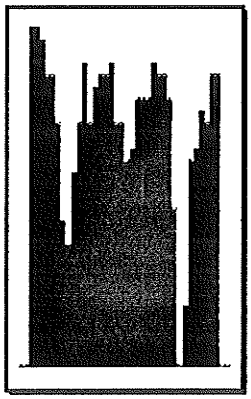


Fig. 1.3 An example tree ring cross section.

Edge enhancement is a process that will increase the contrast of edges within a digital image. Non-edge features are not affected. Edge detection is a process that will remove non-edge features, resulting in an image consisting entirely of edges. It differs with edge enhancement in the respect that edge direction as well as edge strength (i.e., contrast) is obtained. Edge tracking is a procedure for following an edge along its length in a sequential fashion.

1.3 Chapter Outline

In Chapter 2, various image enhancement techniques are discussed. These include thresholding, noise reduction and edge enhancement. The main focus is on edge preserving noise reduction, which is a very useful preprocessing step that can improve the output of many edge detection techniques. The methods presented are compared, in order to determine the one that produces the best results on the sample biscuit image.

Chapter 3 describes various edge detection techniques. The results of their application to the biscuit image are discussed, and a selection of the best method is made. In most cases, the edge detectors were applied to both the original biscuit image and a preprocessed version that had a noise reduction method applied. The majority of the edge detection methods discussed work with grayscale images only. Colour edge detection has not been covered as extensively in the literature, therefore only two methods are presented.

Edge tracking is described in Chapter 4. Three methods are presented, including one designed specifically for tracking tree ring edges. The problems that can occur with the edge tracking algorithms are examined, as well as methods that improve their robustness.

Chapter 5 describes the application that was developed to test the image processing techniques employed. There is a discussion of how ring counting can be achieved algorithmically, as well as the computation of ring area. A case study illustrating the procedure used to extract the rings of a biscuit is also presented. Finally, Chapter 6 provides a summary of the research.

Chapter 2
Image Enhancement

2.1 Introduction

There are various processing steps that can be employed to enhance features that exist in a digital image. In this chapter, three different techniques are examined: thresholding, noise reduction, and edge enhancement. Noise reduction and thresholding are particularly important to edge detection. Edge enhancement is examined to determine its usefulness in aiding the edge detection process.

Digitized images of biscuits will always contain some amount of noise. Noise is defined as the random error that occurs during the digitization process. This noise usually follows a Gaussian distribution and is therefore called *Gaussian white noise*. In the context of biscuits, noise can be defined as the subtle colour differences that occur naturally in the tree, which is not necessarily the result of scanner error. This form of noise becomes a problem when performing edge detection, because it may result in the detection of an improper edge. It is therefore desirable to reduce the noise before edge detection and edge tracking. Noise reduction is usually performed as a preprocessing step. However, it can be incorporated into the edge detector itself so that it occurs during edge detection. When performed as a preprocessing step, it can be applied to the image more than once. Increased application will improve the noise reduction.

All the noise reduction, or smoothing, techniques in this chapter work with localized groups of pixels. The size of a group is dictated by the desired resolution of smoothing. Since features that are smaller than the group size (e.g., thin edges or sharp corners) will ultimately be averaged out, it must be decided which features are to be retained while still achieving the desired noise reduction. In the biscuit image, the important features being examined are the tree rings. The size of these rings can vary, so the smallest group will be used with each noise reduction method to retain as much detail as possible. If larger groups are used there is a chance that important detail could be lost, which would greatly affect the edge tracking process.

This chapter describes thresholding and the following noise reduction techniques: low-pass filtering [LIND], median filtering [LIND], K-nearest neighbour smoothing [DAV2] and maximum homogeneity smoothing [NAGA]. Edge enhancement is briefly examined with a description of high-pass filters [LIND] and the Laplacian transform [LIND].

An evaluation is given in each section for the respective enhancement method. An enlarged portion of the biscuit image is used to illustrate their application to actual tree rings (Fig. 2.1). This test image contains two well-defined rings, a third poorly defined ring, and a light mark that intersects each one. The evaluations are based upon visual results. Objective results for the noise reduction methods were obtained from [CHIN] and are given where applicable. In the conclusion, the most appropriate noise reduction method is chosen, and the results are summarized.

Where necessary, grayscale images, such as that in Fig. 2.1, are shown in a larger size to increase their clarity. The contrast of each one of these images was increased to enhance details, so the gray intensities are not true to the originals. However, this does not detract from the conclusions given.

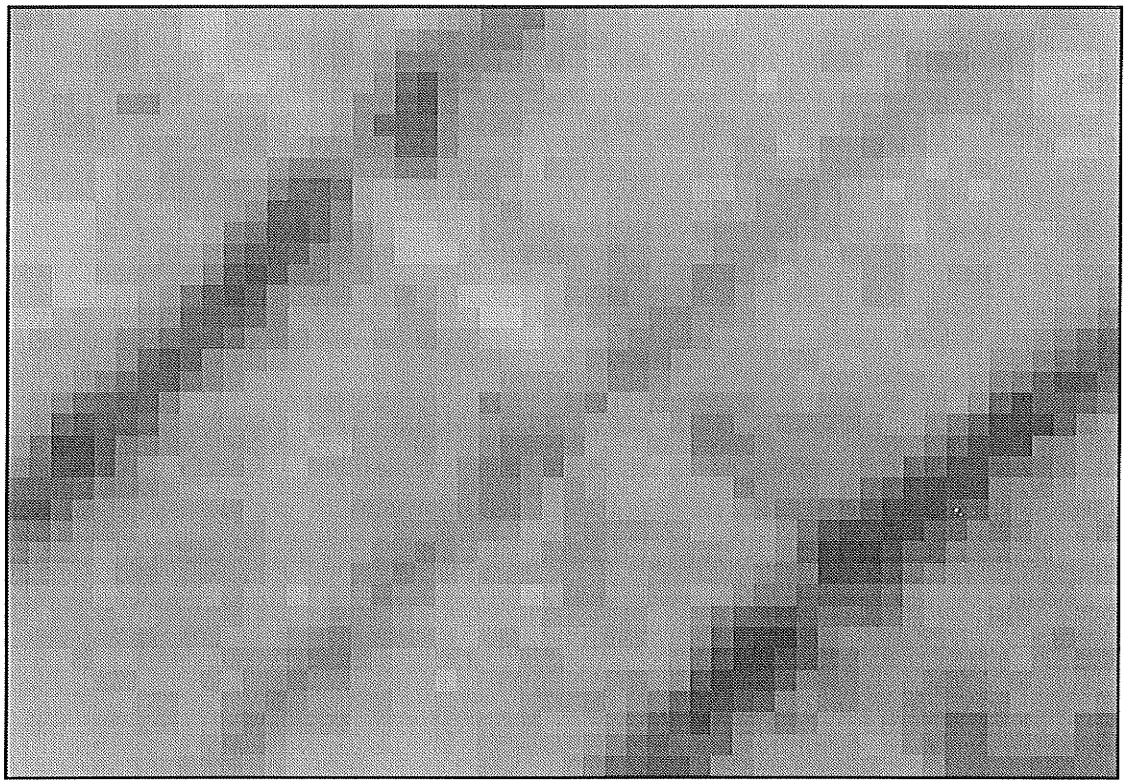


Fig. 2.1 The enlarged section from the sample biscuit used as a test image.

2.2 Thresholding

The most common, and simplest, way of enhancing a picture to bring out details, is thresholding [ROSE1]. This procedure takes an image f with N (N > 1) possible gray levels in the range $[G_{min},G_{max}]$ and reduces it to a bi-level image (only two gray levels, usually black and white). Given a value $t \in [G_{min}, G_{max}]$, each pixel (x,y) in the image is converted as follows:

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq t \\ 0 & \text{if } f(x,y) < t \end{cases}$$

Thresholding can also be done with a range of gray levels:

$$f(x,y) = \begin{cases} 1 & \text{if } u \leq f(x,y) \leq v \\ 0 & \text{otherwise} \end{cases}$$

for given gray levels $u,v \in [G_{min},G_{max}]$.

Another way to threshold an image is by testing whether or not each pixel's gray level is within some set $Z \subseteq [G_{min},G_{max}]$:

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \in Z \\ 0 & \text{otherwise} \end{cases}$$

A variation on the thresholding idea is semithresholding. Gray levels below the threshold become 0, while the remaining ones are left unchanged. For images with objects that are separate from the background, semithresholding will "remove" the background while leaving the objects themselves intact.

In certain cases, thresholding can be used as a simple form of edge enhancement. Consider an image that has a high contrast between the background, and the foreground objects. If intermediate gray levels occur only on the boundaries of the objects, then a range thresholding can be done to bring out the edges. This is only edge enhancement, as opposed to edge detection, because there is no edge direction information obtained.

Such thresholding techniques extract objects that have characteristic gray level ranges or textures. When trying to enhance the rings of a biscuit, we find that these rings do not have uniform gray levels. Some rings may be enhanced by thresholding while others may not (see Fig. 2.2). The threshold values chosen are entirely dependent on the image.

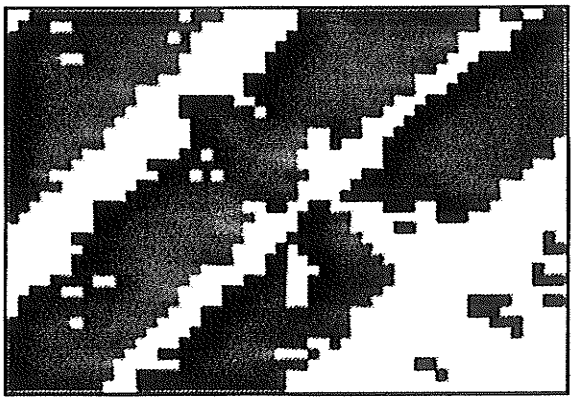


Fig. 2.2 Test image thresholded at 140. Excellent enhancement of two rings, but the third ring is merged with adjacent noise at the same gray level.

Thresholding only becomes important after some sort of edge detection has been performed. Many edge detection schemes produce a low intensity output with the added characteristic that the gray levels tend to be uniform along the detected edges. Simple thresholding can then be used to make these edges visible. This will become apparent when edge detection is discussed in Chapter 3.

2.3 Noise Reduction

2.3.1 Low-Pass Spatial Filters

To understand what a low-pass spatial filter is, it is necessary to think of the image as it pertains to the frequency domain. The rate of change of pixel intensity is referred to as *spatial frequency*. It is a two-dimensional quantity because pixel intensity can change in both the vertical and horizontal directions. A high spatial frequency corresponds to rapidly changing pixel intensities while a low spatial frequency corresponds to constant, or nearly constant, pixel intensities (Fig. 2.3).

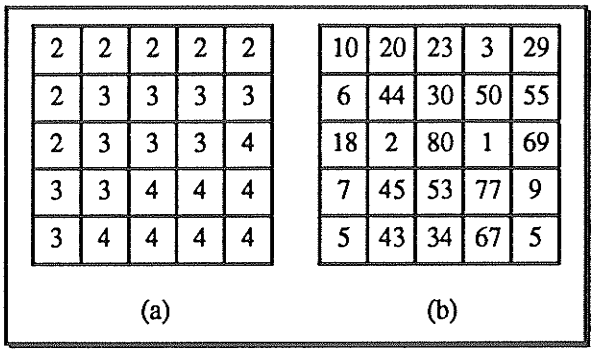


Fig. 2.3 a) A region of low spatial frequency. b) A region of high spatial frequency.

A low-pass spatial filter will leave the low frequency content of an image intact while attenuating (i.e., smoothing) the high frequency content. This works well to reduce the high intensity noise.

The low-pass spatial filters discussed here are referred to as *convolution* processes [LIND]. In such a process, a *convolution kernel* is used to perform a weighted summation over a neighbourhood of pixels. The central pixel is then replaced by the resulting sum (Fig. 2.4).

The following matrices are examples of low-pass filter kernels:

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \begin{bmatrix} 1/10 & 1/10 & 1/10 \\ 1/10 & 1/5 & 1/10 \\ 1/10 & 1/10 & 1/10 \end{bmatrix} \begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix}$$

Notice that the elements sum to 1 in each kernel. This is important in considering how low-pass filters work.

Since low-pass filters are meant to preserve, or enhance, the low frequency content of an image, they must not alter existing regions that have this property. Consider a region of constant pixel intensity, which is a region of low spatial frequency (zero frequency actually). When a low-pass filter is applied to such a region, the new pixel intensity that results is the same as the old one, and no change takes place. Thus, the low frequency content is preserved.

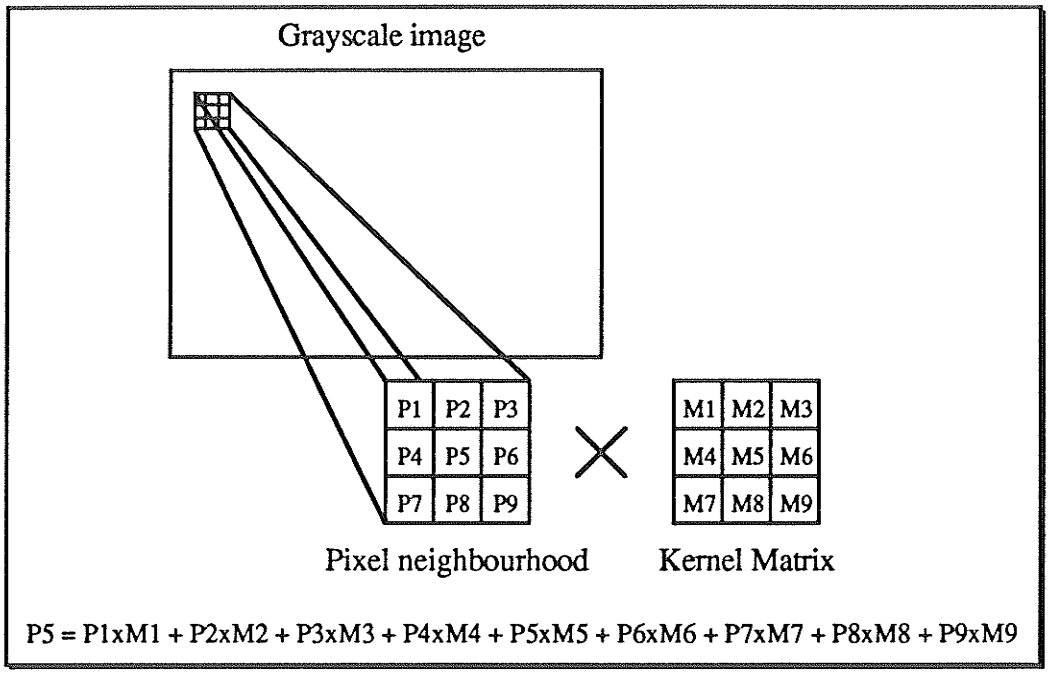


Fig. 2.4. Application of a 3x3 convolution kernel to a group of pixels.

The kernels shown above are weighted averages of pixel intensities, with the first matrix having weights of one (i.e., unweighted average). The other matrices give more weight to the centre pixel. Fig. 2.5 illustrates results of convolving the test image with each one of the kernels.

The conclusions from [CHIN] show that local averaging, while effective at smoothing noise, does not preserve sharp edges. Increased iteration will further reduce noise but will also blur sharp edges. These results are typical of all low-pass filters, not just local averaging. This blurring effect is apparent in Fig. 2.5. Even though noise is significantly reduced, the rings lose their clarity, and no longer have well-defined edges. Repeated application of a low-pass filter would blur the central faint ring to the point where its edges could not be detected. It is important to be able to reduce the noise in the image while still preserving edges. The following smoothing methods achieve this.

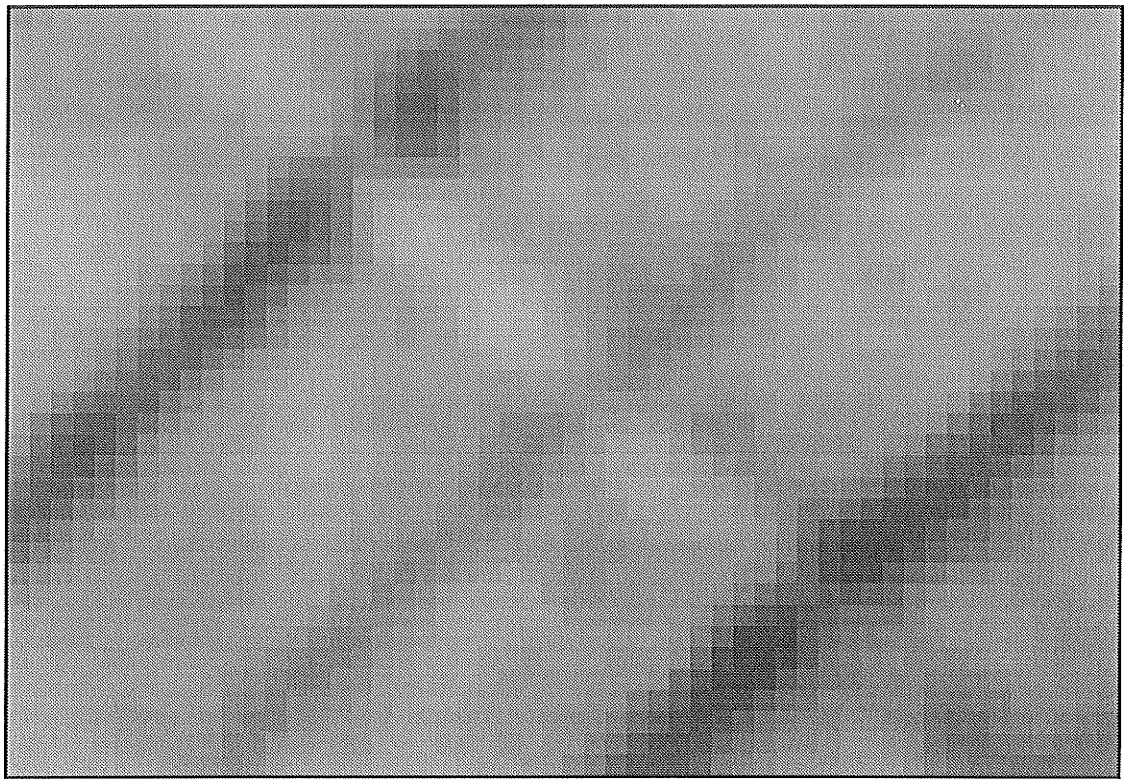


Fig. 2.5(a) Test image convolved with the first low-pass matrix.

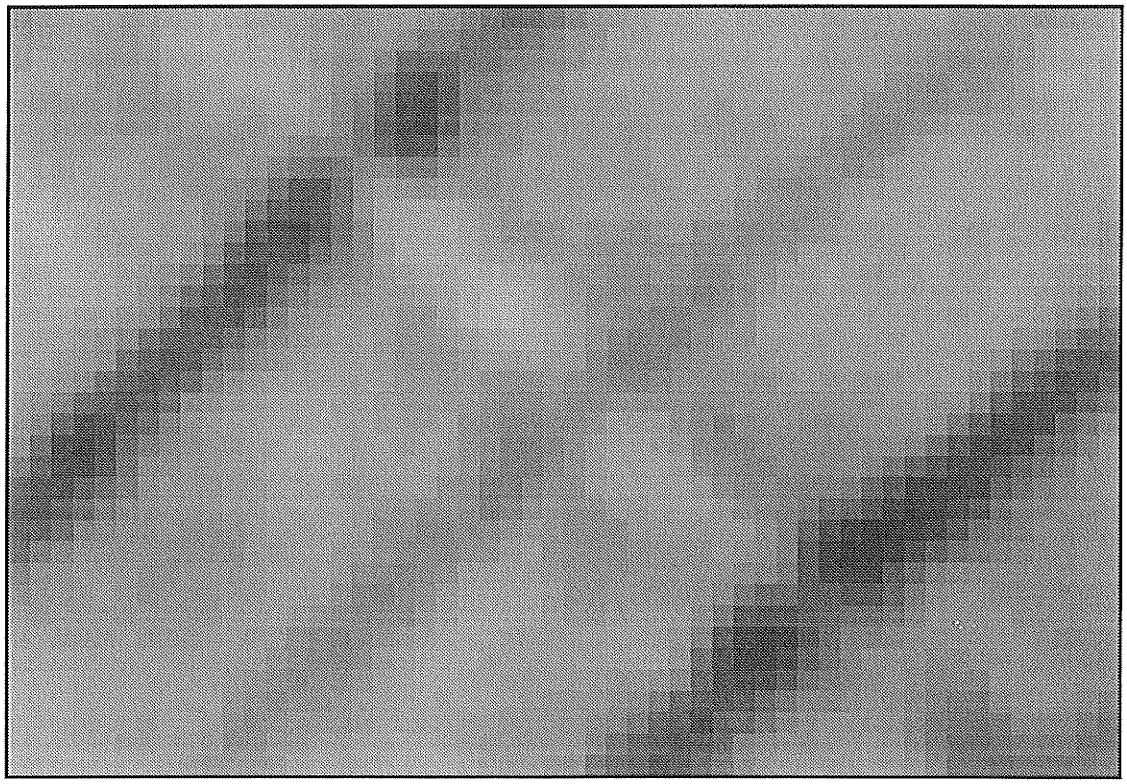


Fig. 2.5(b) Test image convolved wth the second low-pass matrix.

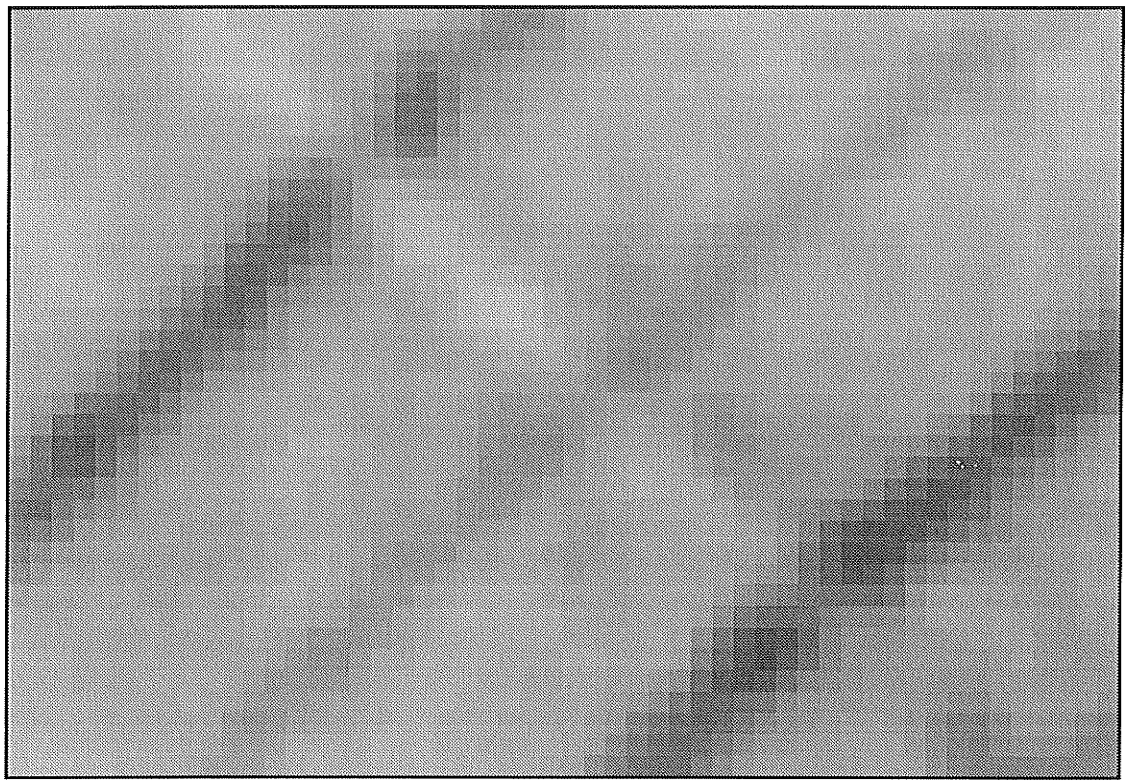


Fig. 2.5(c) Test image convolved with the third low-pass matrix.

2.3.2 Maximum Homogeneity Smoothing

A method proposed in [NAGA] smoothes an image by searching for the most homogeneous area around each pixel in the image, and replacing that pixel with the average of the area. It removes noise in areas of low spatial frequency and does not blur sharp edges. It also has the ability to sharpen existing blurred edges.

At each pixel, the mean and variance of the gray levels are computed for nine neighbouring regions. The centre pixel is replaced with the mean of the region for which the variance is minimum (i.e., maximum homogeneity). An example is shown in Fig. 2.6 which illustrates the shape of the nine regions. Since sharp edges should not be blurred, the averaging must not be applied to an area that contains a sharp edge. An area such as this will have a high variance. Therefore, the variance is a good measure of the homogeneity of the area. The hexagonal and pentagonal corners of the regions at the centre pixel assist in the prevention of the degradation of sharp edges.

This process can be iterated until the gray levels of almost all points in the image don't change. As can be seen in Fig. 2.7, the noise has been reduced substantially with only one iteration. The rings, however, become blurred because they are not perfect step edges. With increased iteration, such faint rings may be blurred to the point where they are indistinguishable from their surroundings. This is very noticeable in the test image by the third iteration. It was noted in [CHIN] that this method had a tendency to distort edges in the form of blurring and over-enhancement and the authors recommended that this method be iterated no more than twice, in order to preserve edges.

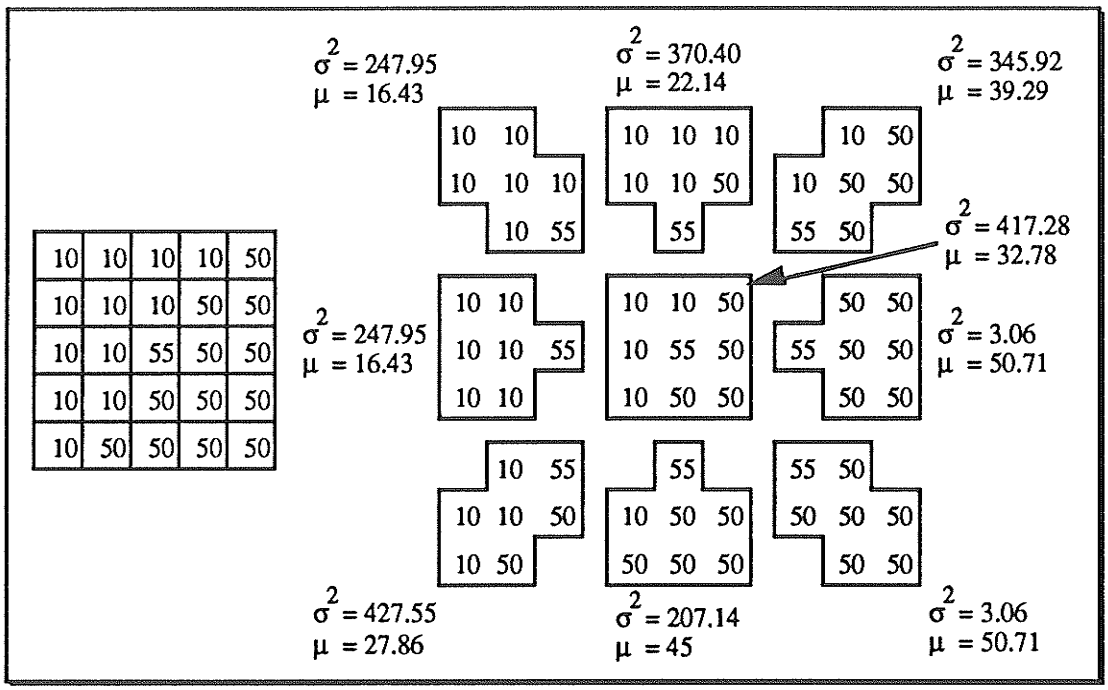


Fig. 2.6 The gray levels in the square area to the left exhibit the structure of a step edge. The nine regions are illustrated along with their mean and variance (shown separate for clarity). The lowest variance is 3.06, therefore the centre pixel is replaced with the corresponding mean of 50.71.

The blurring that does result is not as severe as that of a low-pass filter since there is some edge preservation. Therefore, the maximum homogeneity method offers an improvement over simple local averages. Its edge preservation, however, is unsatisfactory for this application.

2.3.3 Median Filter

The median filter is a common smoothing algorithm that will reduce noise while preserving the edge structure of an image [LIND]. It is similar to a convolution process in that it works with areas of pixels, but it does not use a weighted summation. Instead, the centre pixel intensity is replaced by the median of all the pixel intensities in the neighbourhood (including itself).

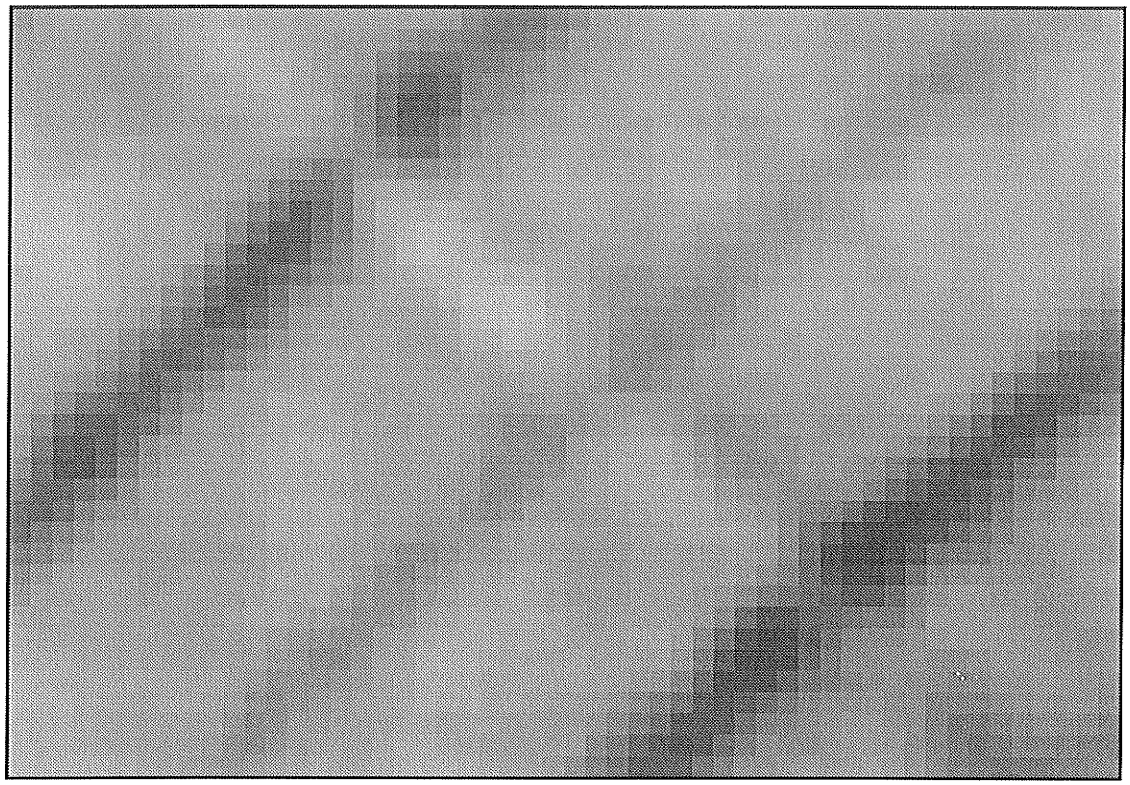


Fig. 2.7(a) Test image processed with one iteration of the maximum homogeneity noise reduction method.

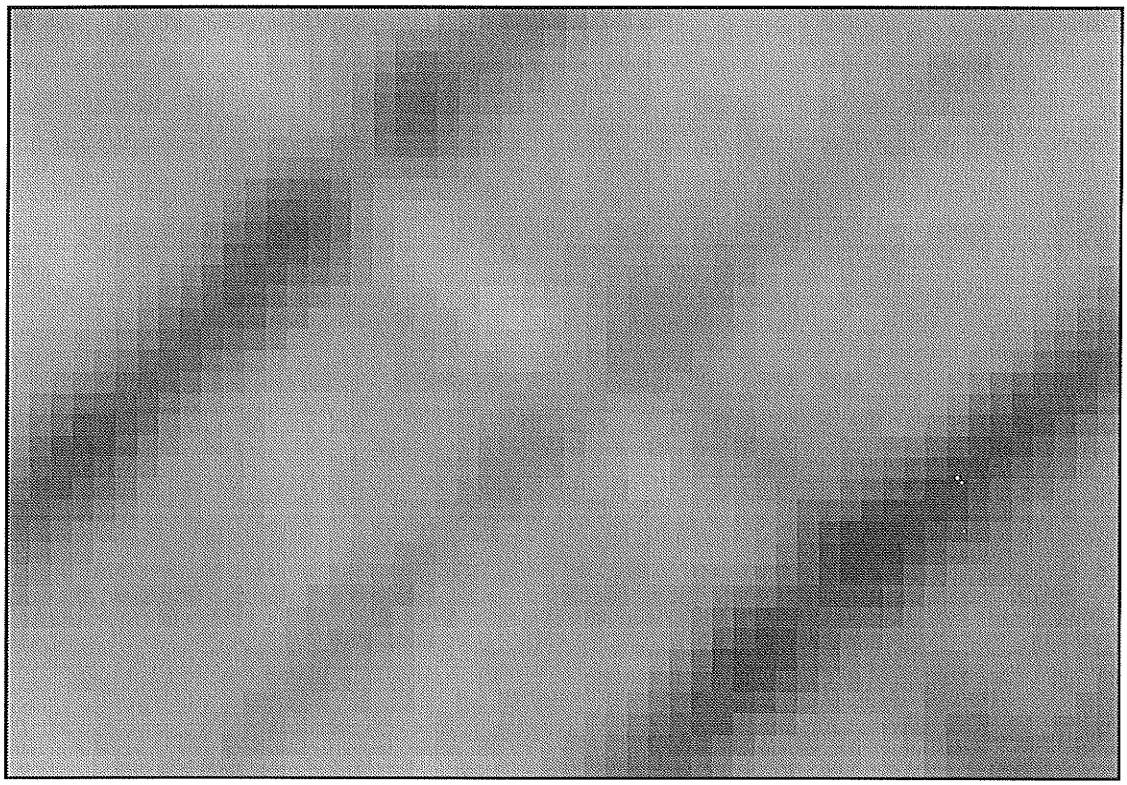


Fig. 2.7(b) Test image processed with two iterations of the maximum homogeneity noise reduction method.

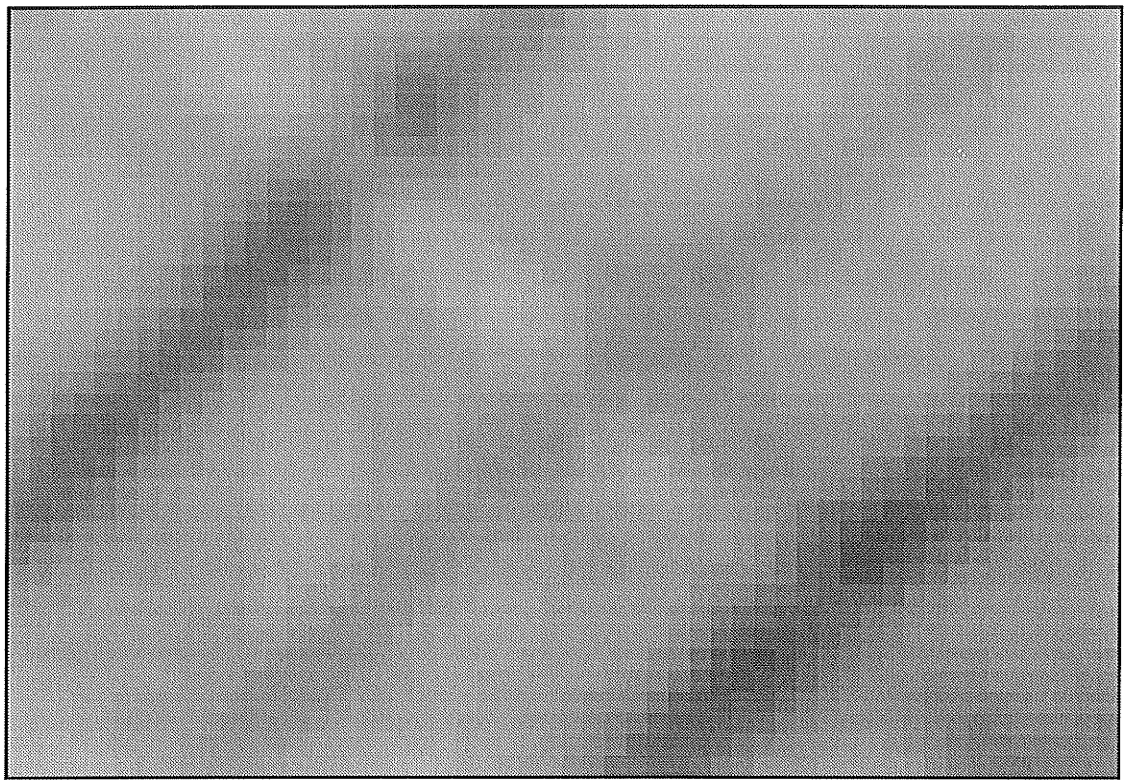


Fig. 2.7(c) Test image processed with three iterations of the maximum homogeneity noise reduction method.

The idea behind the median filter is that, in areas of homogeneous pixel intensity, noise will appear as large deviations from the average intensity of the region. Once the gray levels in the area have been sorted, the gray level of the noisy pixel will appear near one of the ends of the list. If this noisy pixel was at the centre of the area, it will be replaced by a more appropriate gray level (relative to the region). An example is shown in Fig. 2.8.

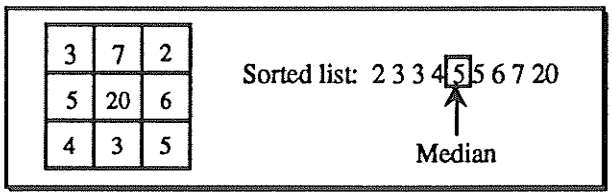


Fig. 2.8 An application of a 3x3 median filter. The centre pixel has a large deviation from the mean of the area and will be corrected by being replaced with 5 (the median).

Since the median filter works to eliminate single pixel errors, edges will be preserved (provided that they are sufficiently larger than the operator). Problems occur when there are features that are significantly smaller than the operator size. As an example, a 3x3 operator applied to a single pixel wide edge will result in removal of the edge (larger operators will also have the same effect). Fortunately, the odds of encountering single pixel wide tree rings, at a scanning resolution of 600 dpi, is highly unlikely. Therefore, the median filter works very well to remove most of the noise in a biscuit image. This is apparent in Fig. 2.9 which shows that the noise is reduced upon each iteration while the rings remain well defined. The mark is also enhanced which may cause it to be detected as an edge since its intensity differs significantly from its surroundings. This is unavoidable unless one enhances only low intensity edges, which is too specific an operation to cover the wide range of possible edge intensities that can occur in a biscuit image.

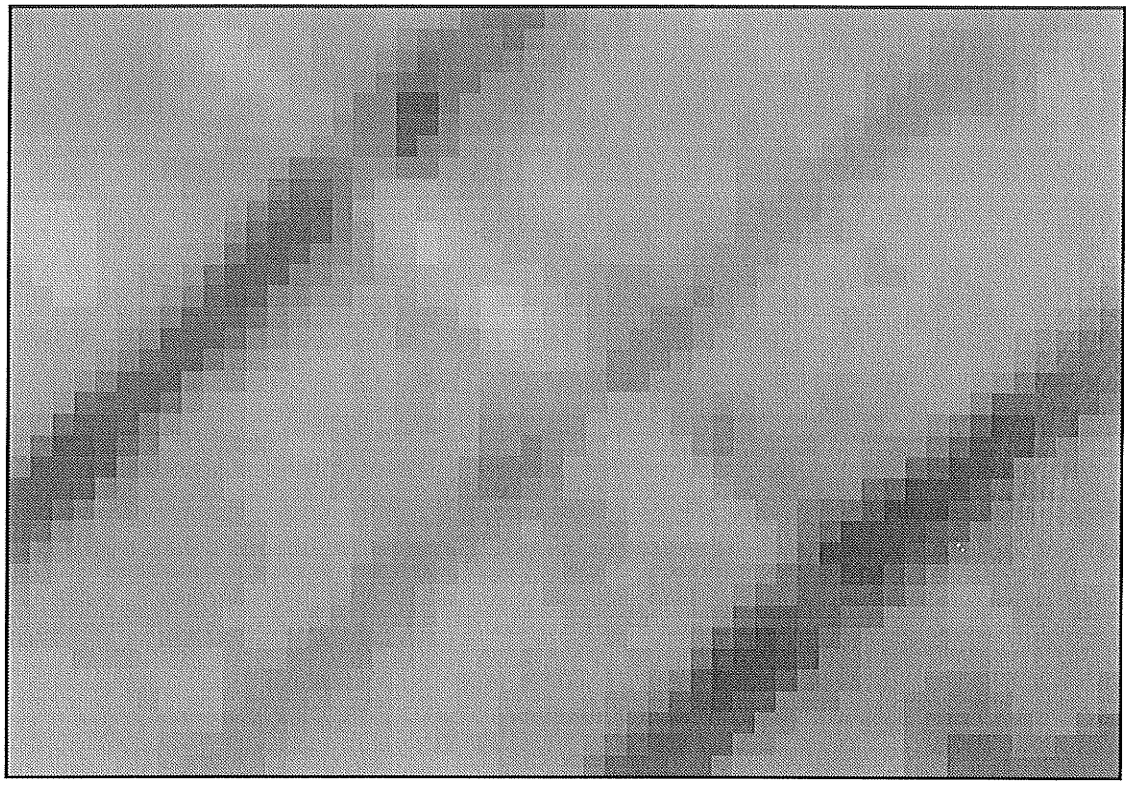


Fig. 2.9(a) Test image processed with one iteration of the median filter.

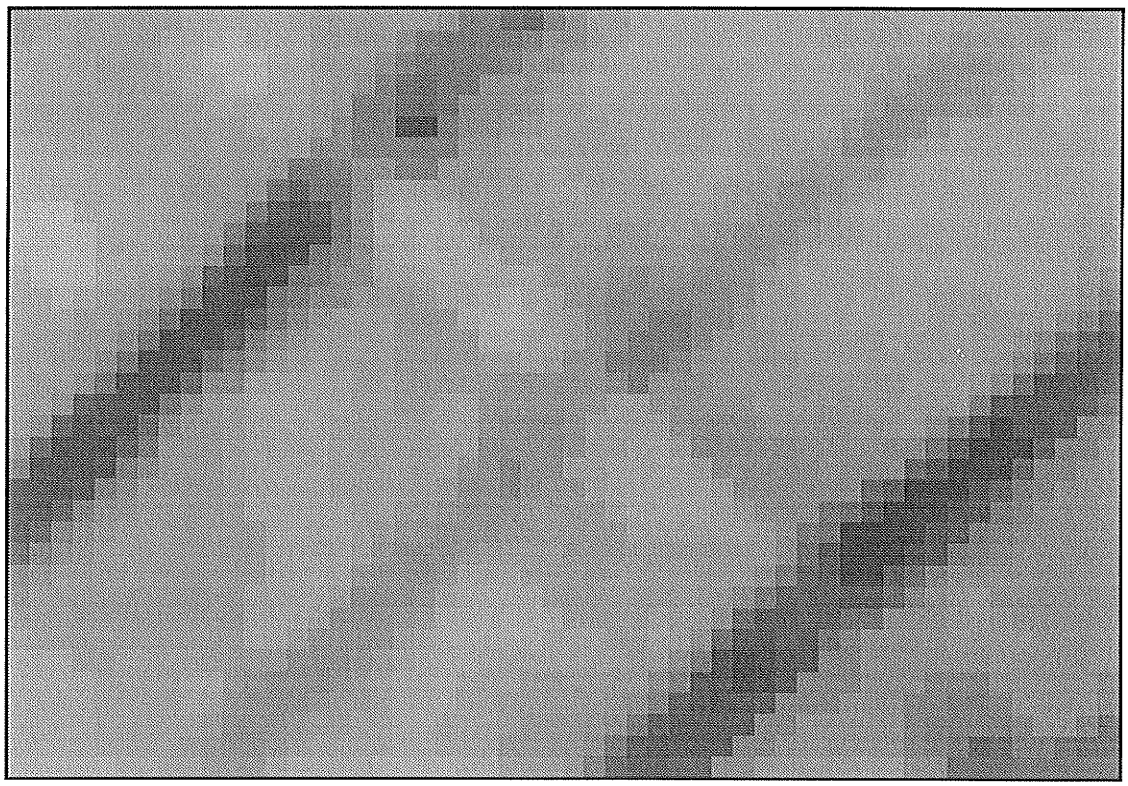


Fig. 2.9(b) Test image processed with two iterations of the median filter.

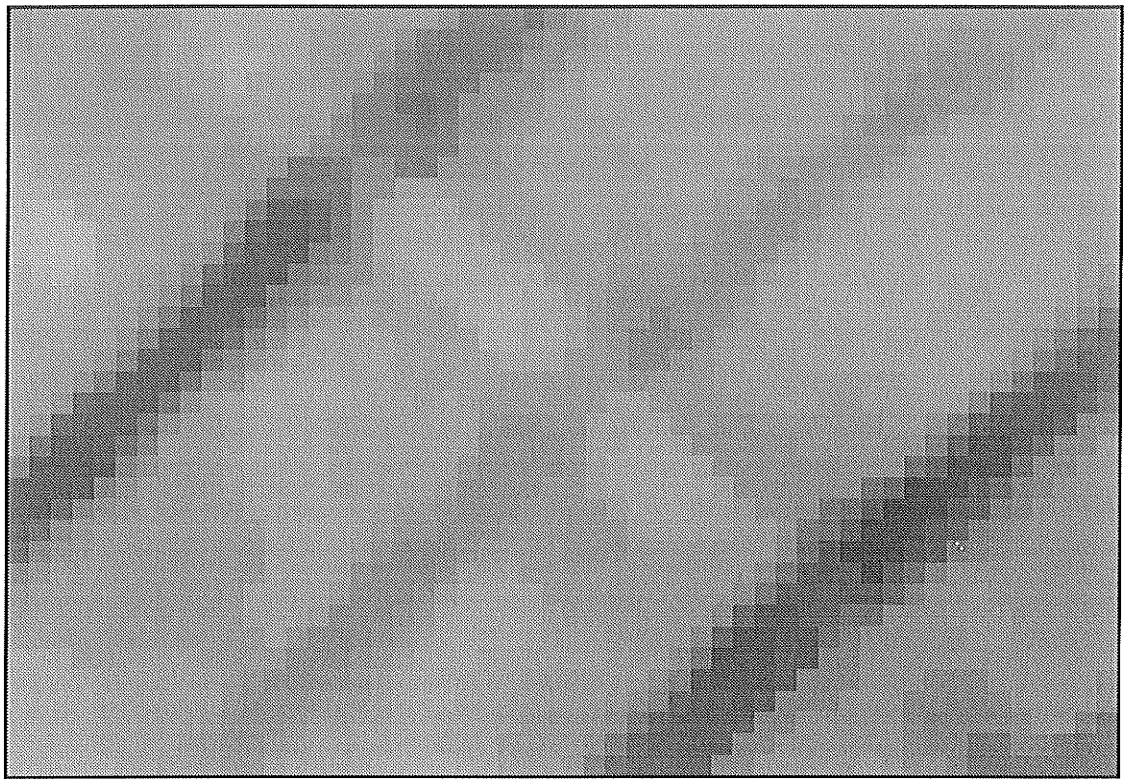


Fig. 2.9(c) Test image processed with three iterations of the median filter.

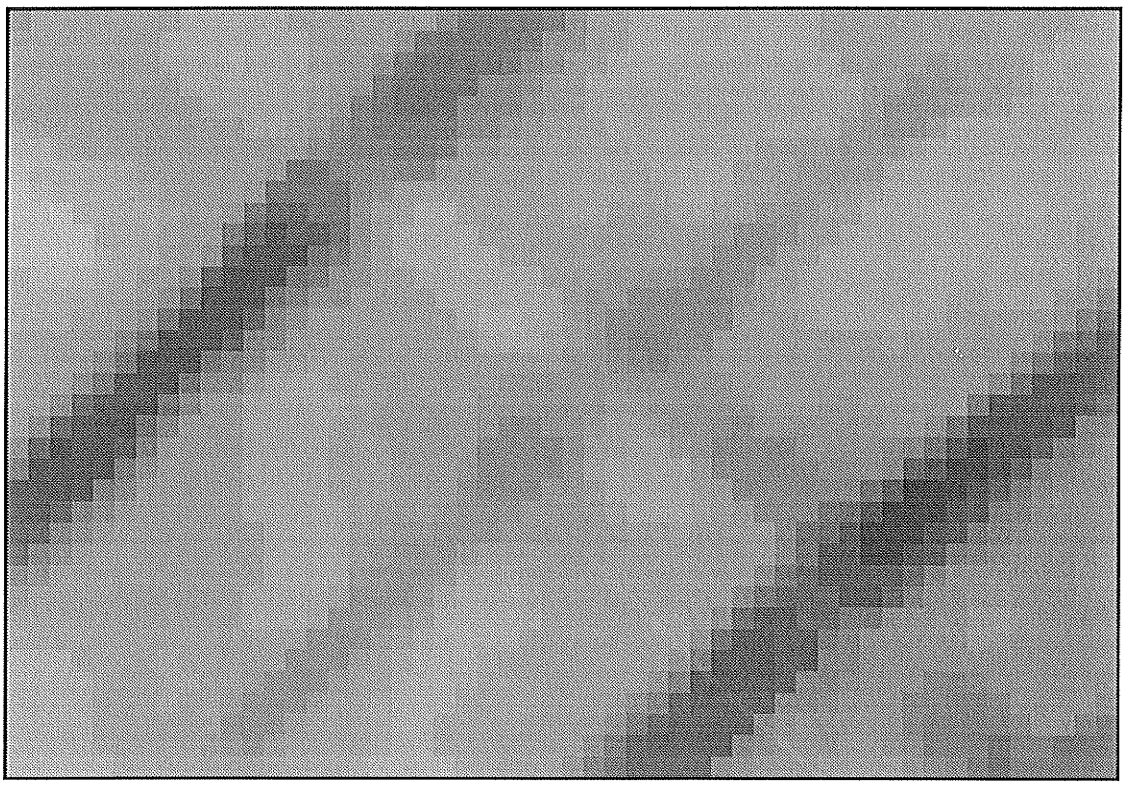


Fig. 2.9(d) Test image processed with four iterations of the median filter.

One noticeable effect of the median filter is the generation of localized areas of constant pixel intensity. Since the median filter does not work with weighted averages, the transition from one such area to the next is abrupt. It is possible that such a transition could be detected as an edge if the difference in pixel intensity is large enough.

The median filter produced very favourable results in the evaluation by Chin and Yeh [CHIN]. It is consistent in reducing noise at all levels in areas of both high and low spatial frequency. It was also shown to preserve sharp edges and ramp edges very well. This is desirable because ring edges resemble ramp edges.

2.3.4 K-Nearest Neighbour Averaging

As was mentioned previously, a simple noise reduction method is unweighted averaging. The disadvantage is that such an operation does not preserve the edge structure of an image. As an extension to this, one can perform K-nearest neighbour averaging [DAV2].

This method works by replacing the gray level at each pixel P by the average of the k neighbours (in an NxN neighbourhood) of P whose gray levels are nearest to that of P. It is possible that within the NxN area, some of the pixels will not lie within the same region as P (see Fig. 2.10). If these pixels are included in the averaging, then the mean of P's region will shift to that of the other region(s). This is what happens with large k and will result in loss of information. Small k will lead to an average with larger variance from the region.

The K-nearest averaging method had, along with the median filter, the best all around results in the evaluation in [CHIN]. It effectively reduced noise and preserved edges consistently at all levels of noise. One test performed in their evaluation was the effect of changing edge slope. The K-nearest method exhibited improved edge preservation with decreasing edge slope. This is important since ring edges are more similar to ramp edges than step edges.

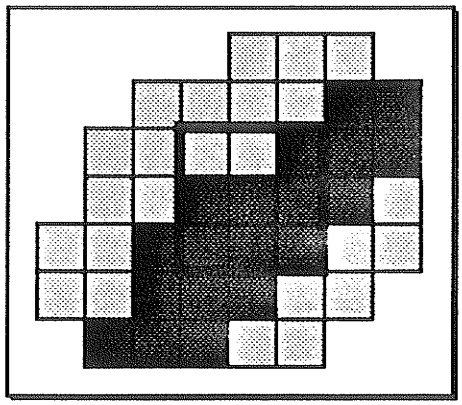


Fig. 2.10 The outlined area encloses two different regions. The centre pixel will get the correct average only if the 6-nearest neighbours are averaged.

The results from [DAV2] suggest that 6-nearest neighbour averaging provided the best subjective enhancement. Fig. 2.11 shows the results on the test image using a 3x3 neighbourhood. Even though the noise was not significantly reduced until at least the third iteration, the rings continued to become even more defined. Even the faint ring became more distinguishable after a few iterations. However, this method shares the same disadvantage as the median filter because it enhances the mark as well.

2.3.5 Colour Noise Reduction

Performing noise reduction upon an image with 256 colours rather than 256 shades of gray is not trivial. The gray levels that result from applying a noise reduction method to a grayscale image will fall in the range 0-255. Depending on the method used, some round off error may occur. The low-pass filters or the K-nearest noise reduction method use averages which may not result in integer gray levels, while the median filter will always produce integer values.

If these noise reduction methods are to be used upon an image with 256 colours, two problems occur. Since each colour is defined by the intensity of its red, green and blue components, there is the problem of how to apply the filter to these components and then combine the resulting values. The other problem is the possible occurrence of colours that do not lie within the colour table.

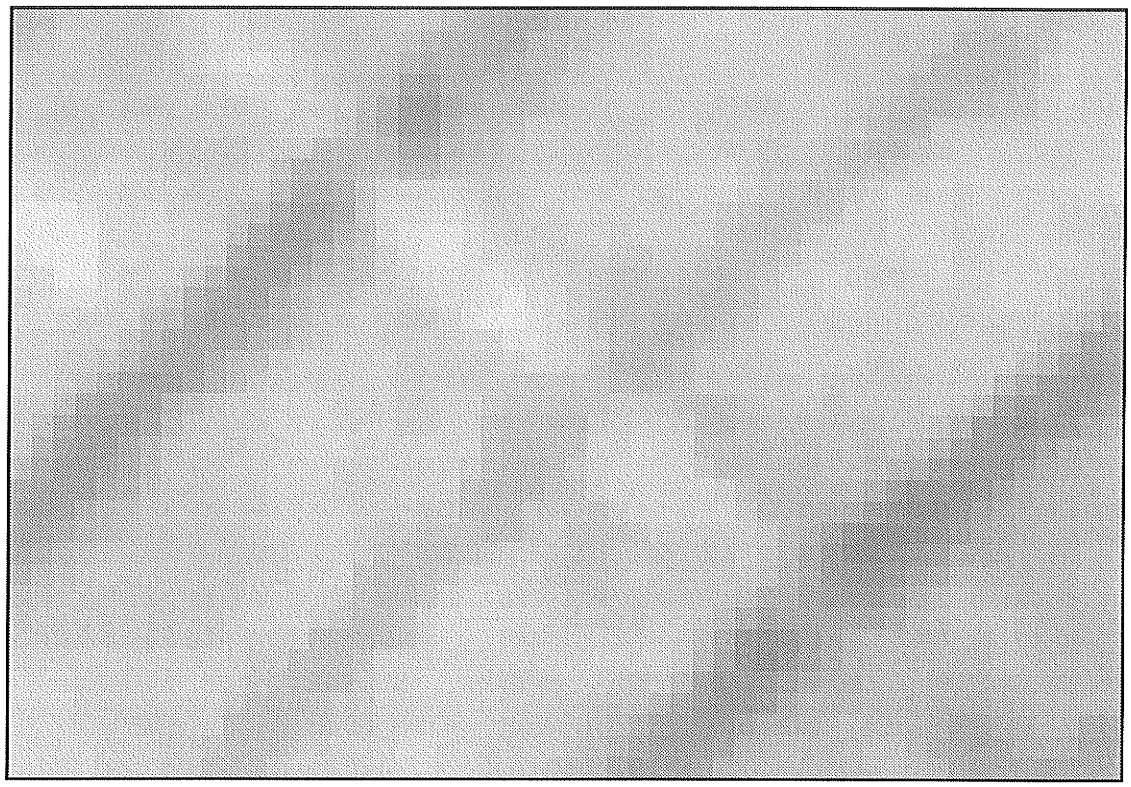


Fig. 2.11(a) The test image processed with one iteration of the K-nearest noise reduction method (K=6).

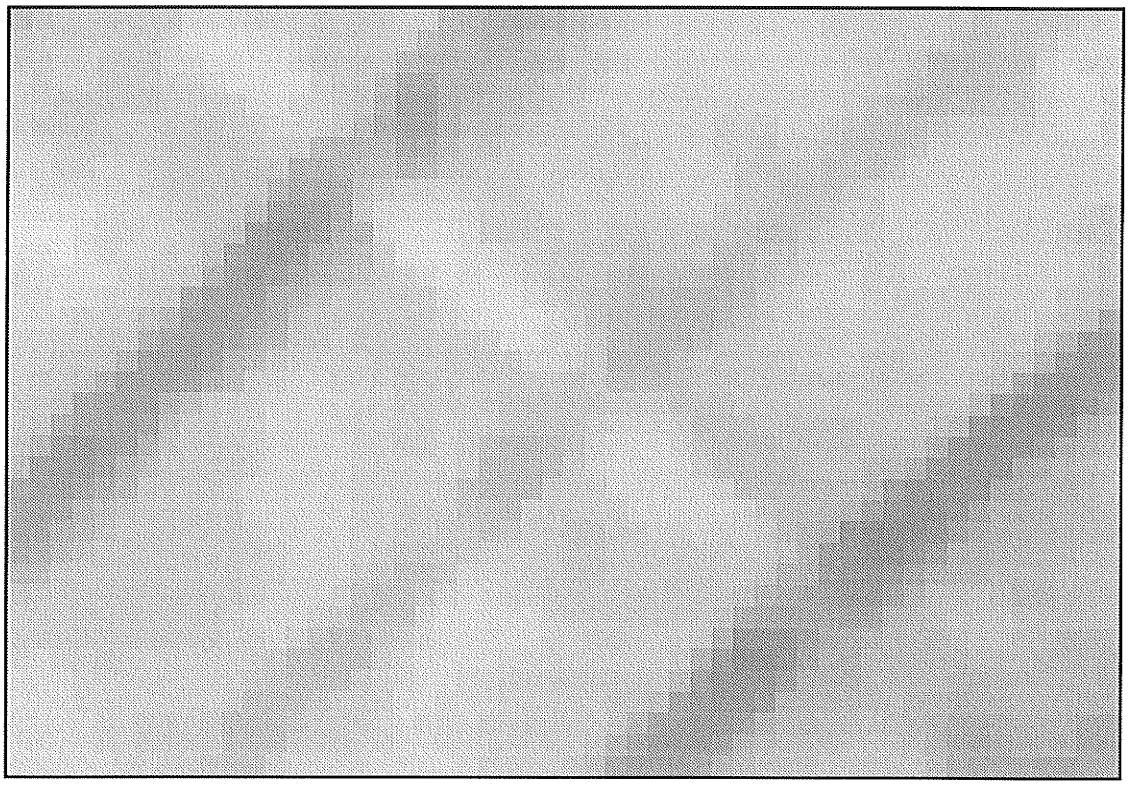


Fig. 2.11(b) The test image processed with two iterations of the K-nearest noise reduction method (K=6).

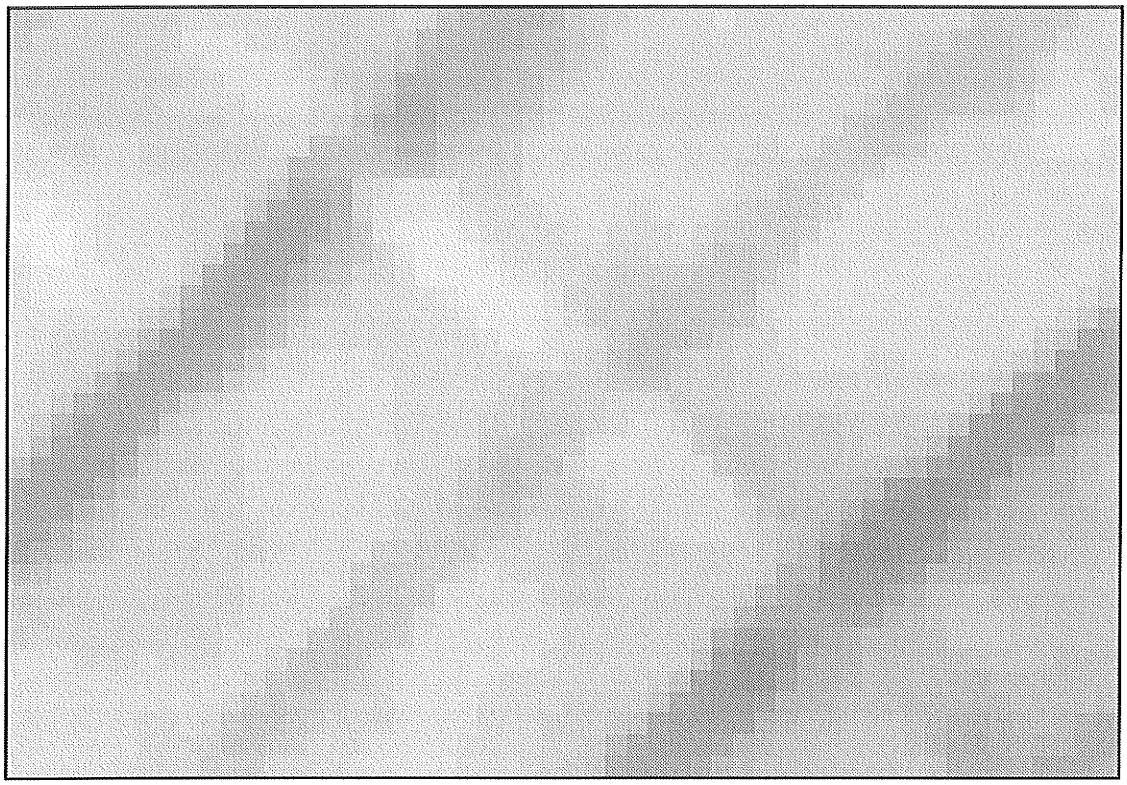


Fig. 2.11(c) The test image processed with three iterations of the K-nearest noise reduction method (K=6).

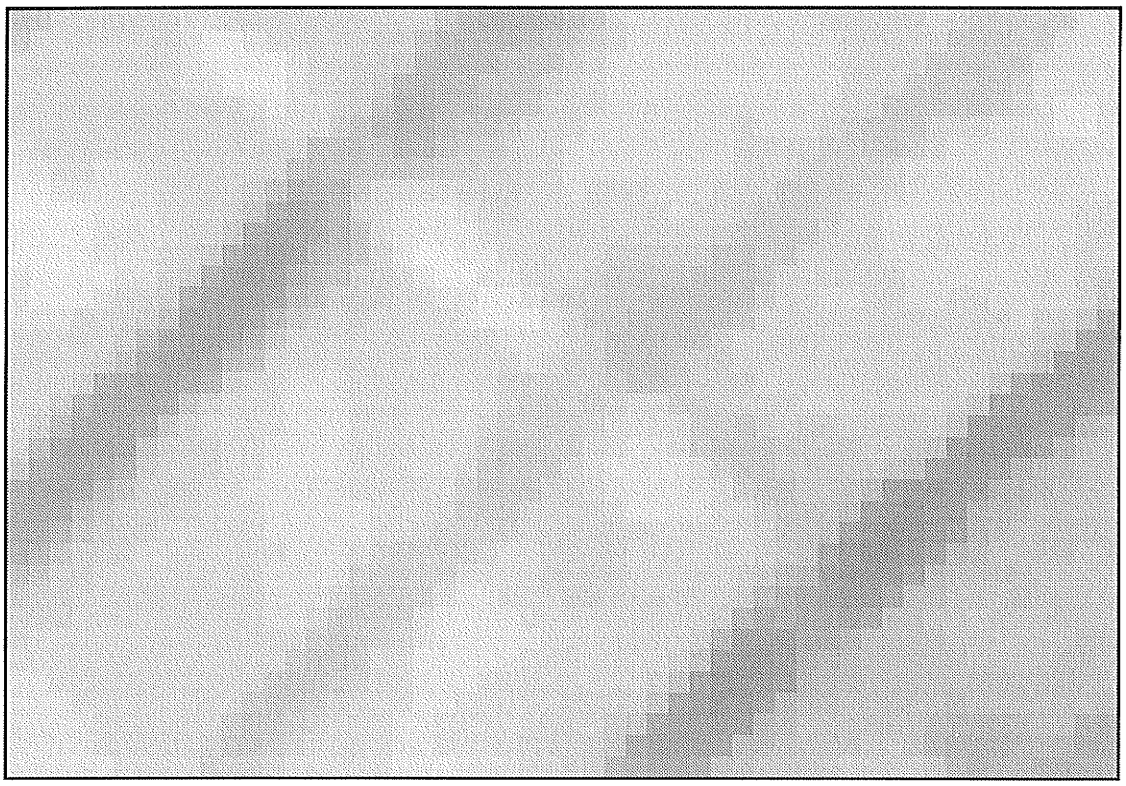


Fig. 2.11(d) The test image processed with four iterations of the K-nearest noise reduction method (K=6).

For an example, consider the median filter. The first problem is solved by simply applying the median filter to each component separately, so that, given n colours, the median can be obtained from the n red components, the n green components and the n blue components, resulting in three medians which represent the red, green and blue components of the replacement colour. However, it is quite possible that these three new components do not match any of the colours in the colour table. If the colours are treated as 3-vectors, then the colour in the colour table that has the minimum Euclidean distance from the new colour can be used instead. However, this introduces a large amount of error.

This error made it undesirable to modify any of the above noise reduction methods to work on an image with only 256 colours. Rather than reducing noise and enhancing edges, they would corrupt the image and reduce detail. This would severely effect any edge detection performed.

2.4 High-Pass Filters

High-pass filters are similar to low-pass filters, but they enhance the high frequency content of an image instead of the low frequency content. A high-pass filter will enhance edges because of the high spatial frequency characteristic of an edge, but will leave the low frequency content intact. The disadvantage is that noise is also enhanced because it shares the same characteristics as step edges. However, in the presence of low noise, high-pass filters are very adept at enhancing edges. The following matrices are some high-pass convolution kernels given in [LIND]:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

The key to the operation of a high-pass filter is in the large centre coefficient. This causes the gray level of the centre pixel to be multiplied by many times its value. The

smaller negative coefficients work to reduce the effect of this larger weighting value. The result is that the high frequency content is enhanced. The coefficients must sum to one so that areas of low spatial frequency are not affected (as explained earlier).

High-pass filters are not useful for enhancing tree rings unless the noise is removed. Fig. 2.12 illustrates the application of each of the above kernels to the test image. The result in each case is a very noisy image. Fig. 2.13 illustrates their application to the test image which has been smoothed with 4 iterations of the 6-nearest neighbour method. As expected, there is less noise, but the rings are actually reduced in detail. Thresholding will bring out the details, but this defeats the purpose of performing edge detection. Therefore, high-pass filters are not a useful preprocessing step.

2.5 Laplacian Edge Enhancement

Laplacian edge enhancement is similar to high-pass filter enhancement in that it enhances the high frequency content of an image. A significant difference is that areas of constant or linearly increasing intensity are attenuated and become black (which will become apparent later). It is also omnidirectional which means that it highlights edges regardless of their direction. These different qualities make it superior to simple high-pass filter enhancement methods.

Laplacian edge enhancement is based on the Laplacian transform. The Laplacian transform L of a function $f(x,y)$ is as follows:

$$\mathrm{L}(f(x,y)) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

For discrete functions, the partial derivatives can be approximated by

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) - 2f(x) + f(x-1)$$

$$\frac{\partial^2 f}{\partial y^2} = f(y+1) - 2f(y) + f(y-1).$$

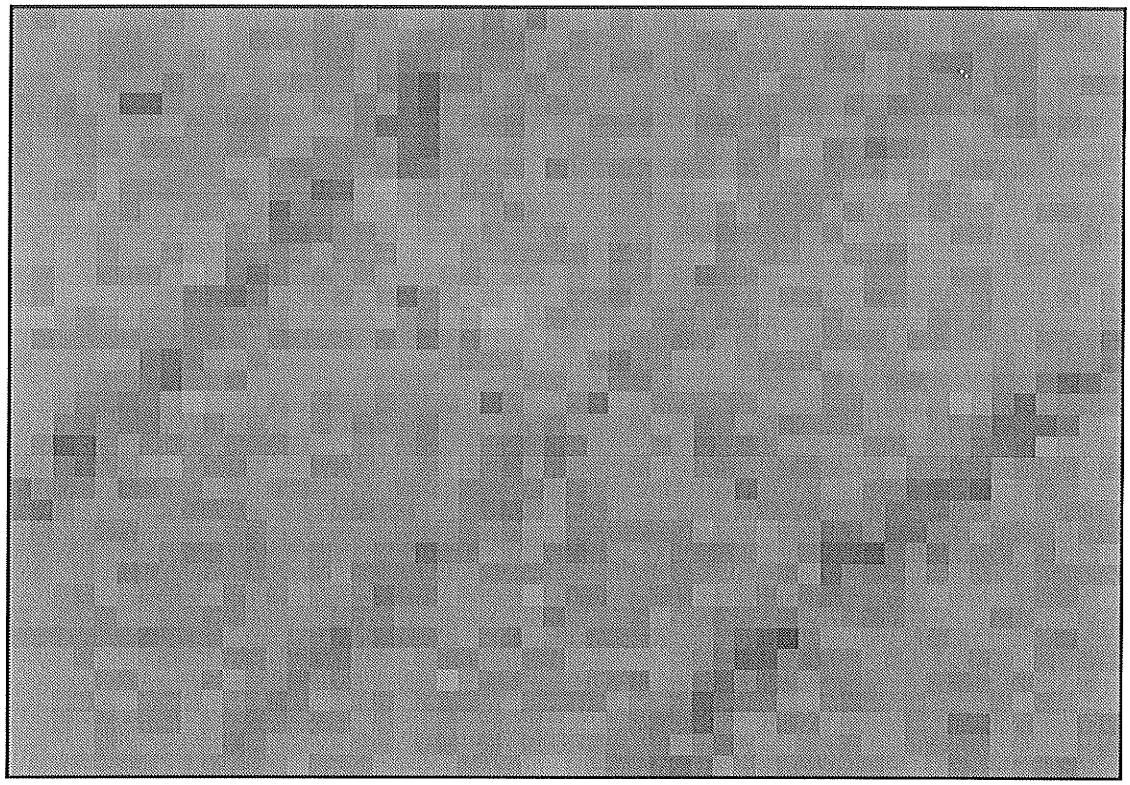


Fig. 2.12(a) Unsmoothed test image convolved with the first high-pass matrix.

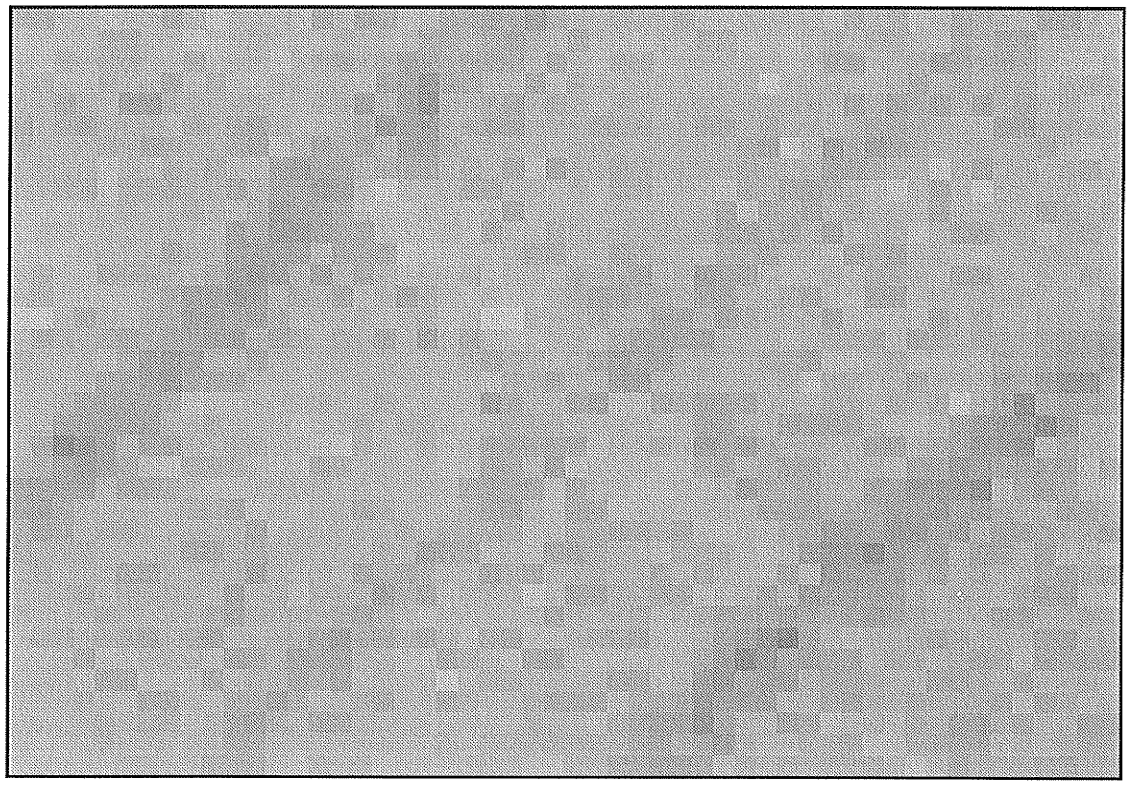


Fig. 2.12(b) Unsmoothed test image convolved with the second high-pass matrix.

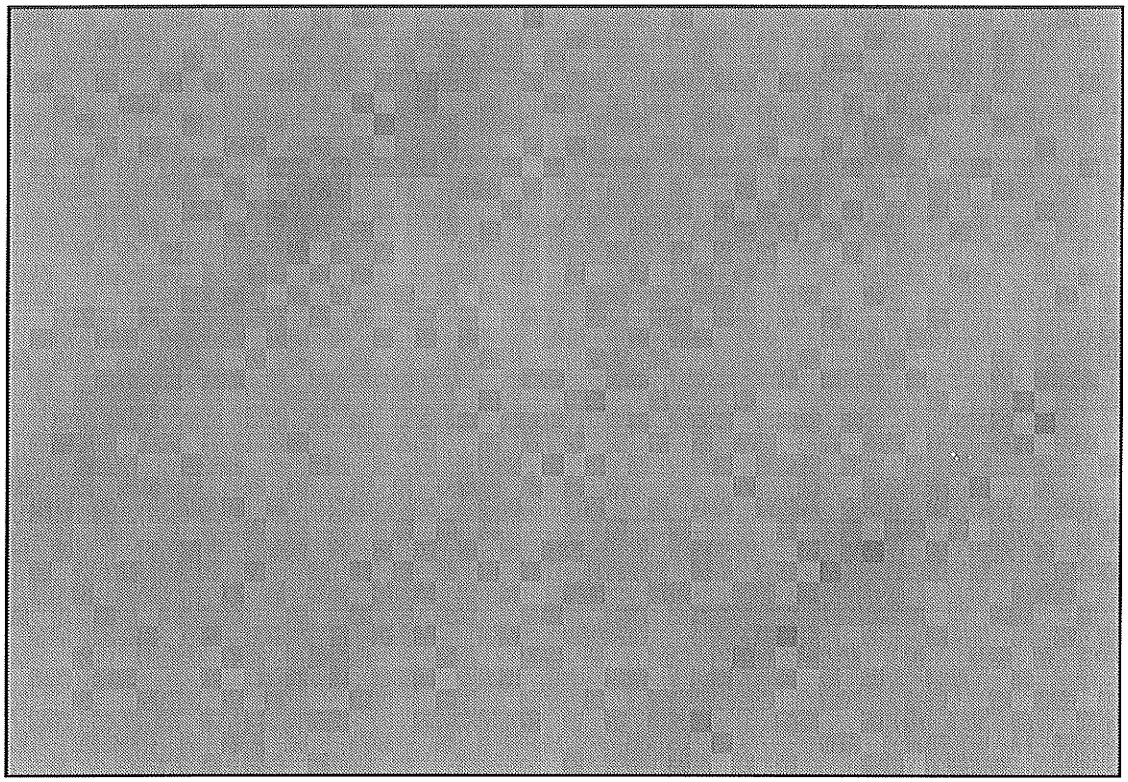


Fig. 2.12(c) Unsmoothed test image convolved with the third high-pass matrix.

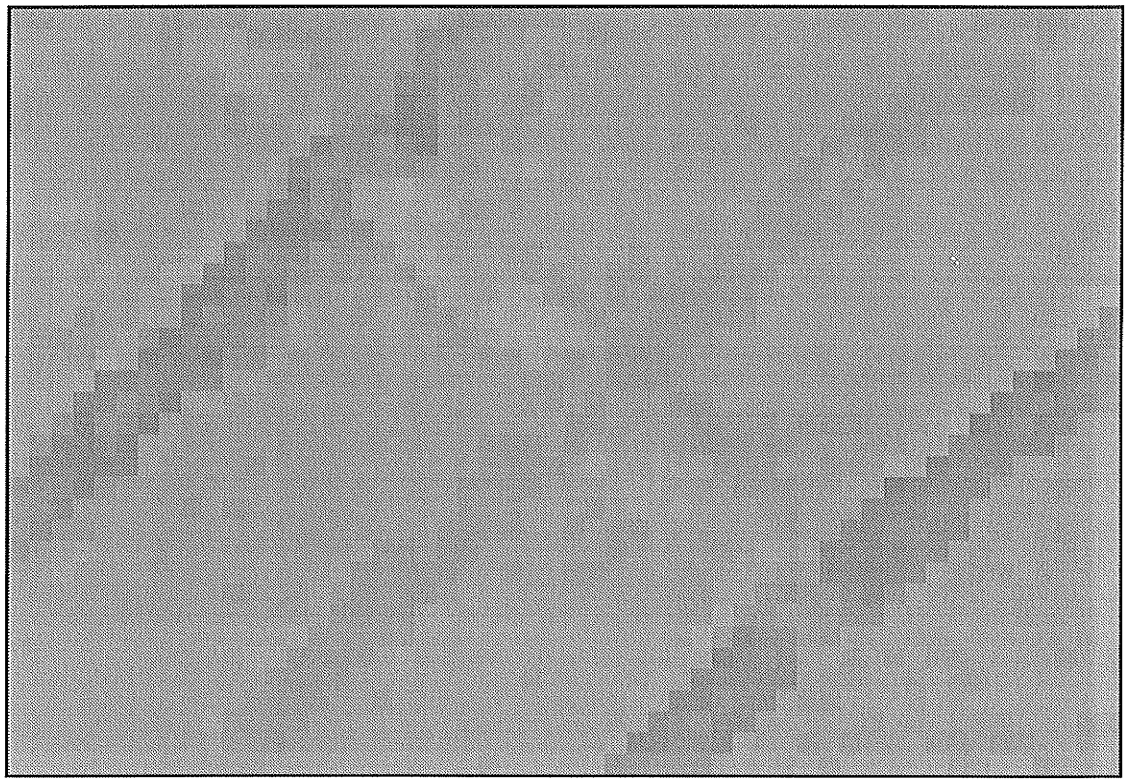


Fig. 2.13(a) Pre-smoothed test image convolved with the first high-pass matrix. Smoothing done with four iterations of K-near noise reduction (K=6).

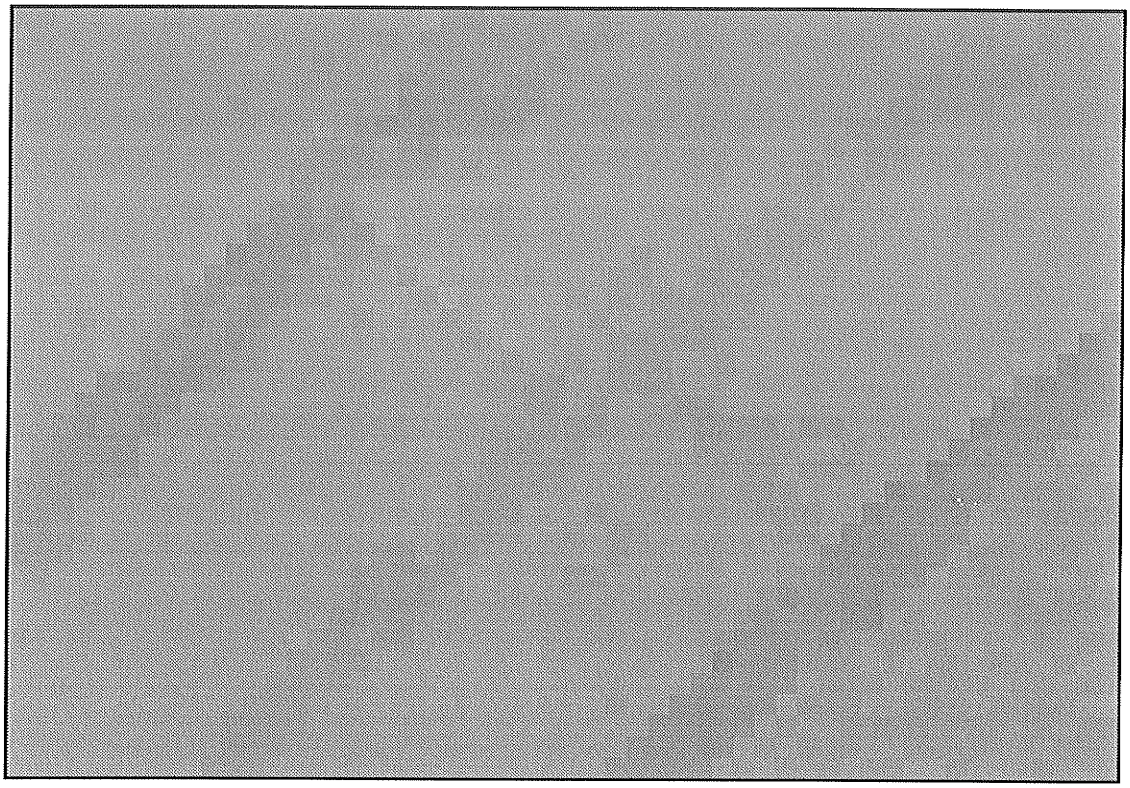


Fig. 2.13(b) Pre-smoothed test image convolved with the second high-pass matrix. Smoothing done with four iterations of K-near noise reduction (K=6).

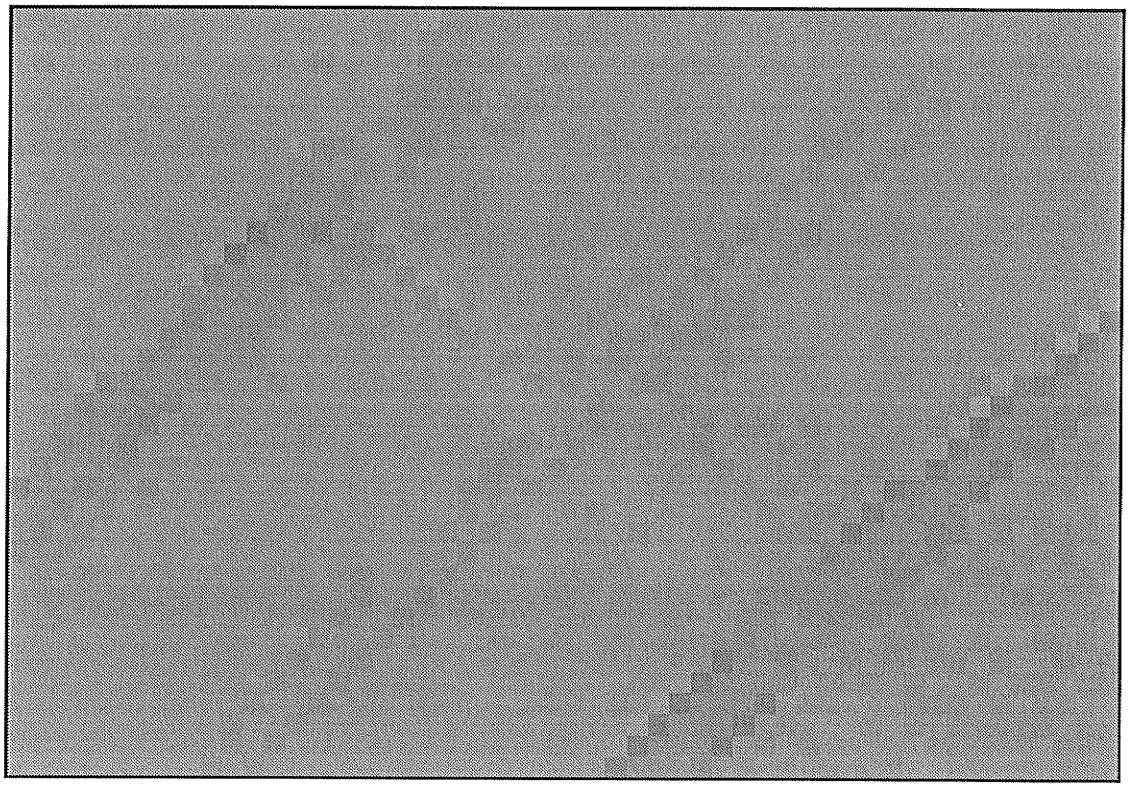


Fig. 2.13(c) Pre-smoothed test image convolved with the third high-pass matrix. Smoothing done with four iterations of K-near noise reduction (K=6).

Therefore, the Laplacian can be approximated by

$$L(f(x,y)) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y).$$

This can be expressed as a convolution kernel:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

An important attribute is that the coefficients sum to zero. This will cause the low spatial frequencies to be attenuated (i.e., areas of constant or linearly increasing pixel intensity will become black).

Laplacian edge enhancement has the same drawbacks as high-pass filters because it enhances noise and edges equally. Fig. 2.14(a) illustrates its application to the test image. In the figure, positive values are displayed as white, zero values are displayed as a medium gray, and negative values are displayed as black. The image has a large amount of noise and the rings are barely visible. Removal of negative values improves the image, but there is still a fair amount of positive noise left.

Smoothing the image first greatly improves the results (Fig. 2.14(b)). In comparison to the unsmoothed test image, the faint ring is enhanced very well and there is very little noise. However, there are breaks in the middle ring. Again, removal of the negative values would lessen the noise without effecting the ring enhancement, but the positive values do not provide enough definition by themselves.

Together with a noise reduction method, the Laplacian works very well to enhance edges visually. The goal, however, is to obtain as much information and detail as possible. Since the Laplacian filter can reduce edge detail, because of its sensitivity to noise, it will not improve the results of any edge detectors. As will be seen in Chapter 3, the Laplacian transform is far more important for a particular form of edge detection that makes use of the Gaussian function.

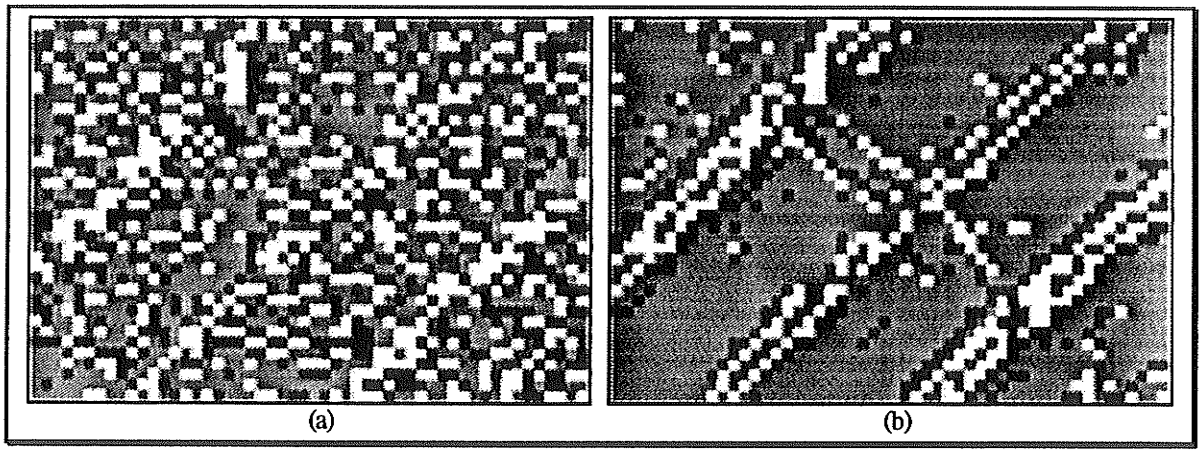


Fig. 2.14 a) Unsmoothed test image convolved with Laplace filter. b) Smoothed test image convolved with Laplace filter. Smoothing done with 4 iterations of 6-nearest filter.

2.6 Conclusions

Noise reduction is an important preprocessing step. Edge detection methods that do not have any noise immunity (i.e., cannot distinguish between noise and true edges) need to have noise removed before they can perform satisfactorily. Each of the smoothing algorithms described in this chapter provides some level of noise reduction. However, there are important differences that dictate which one is most appropriate for this application.

Low-pass filter noise reduction does not produce desirable results. The image will be blurred regardless of the underlying information and this loss of edge information will result in poor edge detection. The maximum homogeneity method offers an improvement, but will still blur the rings since they are not perfect step edges.

The median and K-nearest methods are more desirable because they effectively reduce noise while preserving both ramp and step edges. These subjective observations accompanied with the results obtained from [CHIN] suggest that these two image enhancement methods are the most beneficial in aiding edge detection. Although the results are similar, the K-nearest noise reduction method was used when testing the edge detection

algorithms presented in the next chapter. Its ability to enhance ramp edges well makes it more effective since rings tend to have such edges.

It was shown that high-pass filters and Laplacian edge enhancement are not useful preprocessing steps if edge detection is to be performed. Their enhancement characteristics will not improve edge detection and in some cases may hinder it. Thresholding on the other hand, is an important image enhancement technique that is necessary to bring out the details of an edge detected image.

Chapter 3
Edge Detection

3.1 Introduction

Edge detection is the process of obtaining edge information from a digital image. This information will indicate which pixels are part of an edge and each one of these edge pixels will be assigned the direction of the edge that passes through it. In addition to this, each edge pixel can also be assigned an edge weight corresponding to the strength of the local edge.

By edge detecting the biscuit image, it is possible to create another image that consists solely of the ring edges. The quality of the edge detection method used depends on the resulting *signal to noise ratio* (SNR) and *edge localization.* The SNR refers to the ratio of the edge operator's response to actual edges and its response to noise. For example, a SNR of 1 means that the response to both noise and edges is identical, and a distinction cannot be made between the two. A SNR of 100 would mean that noise has only a 1% contribution to the edge operator output. If an edge operator does not have a high SNR then an edge-preserving noise reduction method must be used as a preprocessing step.

Edge localization refers to the actual position of the detected edge. Good localization corresponds to the output of detected edges exactly at the positions of the original edges. Poor localization corresponds to the output of detected edges that are offset from the original edges. The poorer the localization, the thicker the edges due to the displacement.

The size of the edge operator will sometimes dictate the localization. Smaller operators will have good localization while larger ones may not. As with the image enhancement operators, if the edge operator is small, then smaller features will be detected. If the edge operator is large, then larger features will be detected. With a larger edge operator, smaller features may be detected, but with poorer localization.

To compensate for possible noise and edge localization problems, the sample biscuit was scanned at 600 dpi. Lower scanning resolutions were not sufficient for this particular biscuit. It is necessary to scan at a high enough resolution so that the rings are separate from one another. For the particular sample biscuit, 600 dpi provided good ring definition as well as an image that was not too large to work with. It is possible that other biscuits may not require such high scanning resolutions, while others may require scanning resolutions even finer than 600 dpi. In any case, finer scanning can only improve edge detection and edge tracking results because of the better definition of small details.

In this chapter, various edge detection methods and their effectiveness in detecting ring edges are discussed. To illustrate their edge detection capabilities and limitations, the test image from Chapter 2 is used and will be referred to as test image 1. For further illustration, two other test images are used. The first one shows three relatively closely spaced well defined rings, with breaks in them, and a small artifact in the form of a spot edge (Fig. 3.1(a)). The other is an example of poor ring definition (Fig. 3.1(b)). The upper ring is well defined, the lower ring is rather faint, and the centre ring is barely distinguishable at all, with many breaks in it. These two images are referred to as test image 2 and test image 3, respectively.

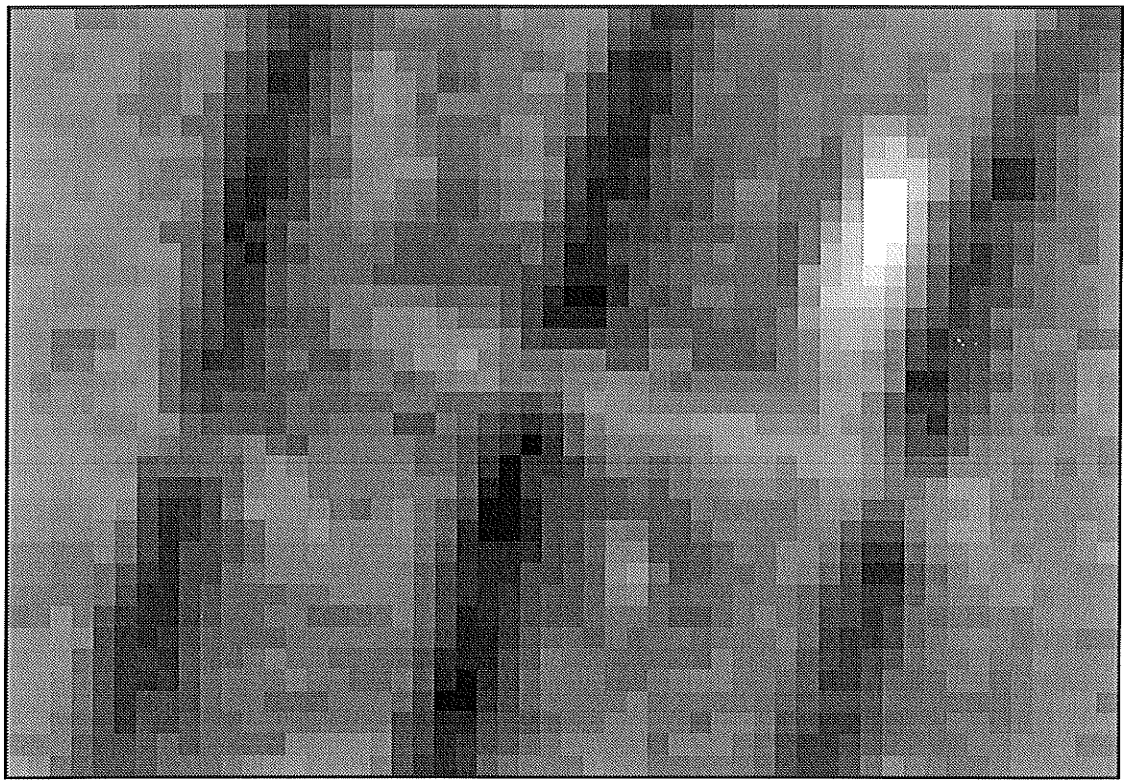


Fig. 3.1(a) Test image 2.

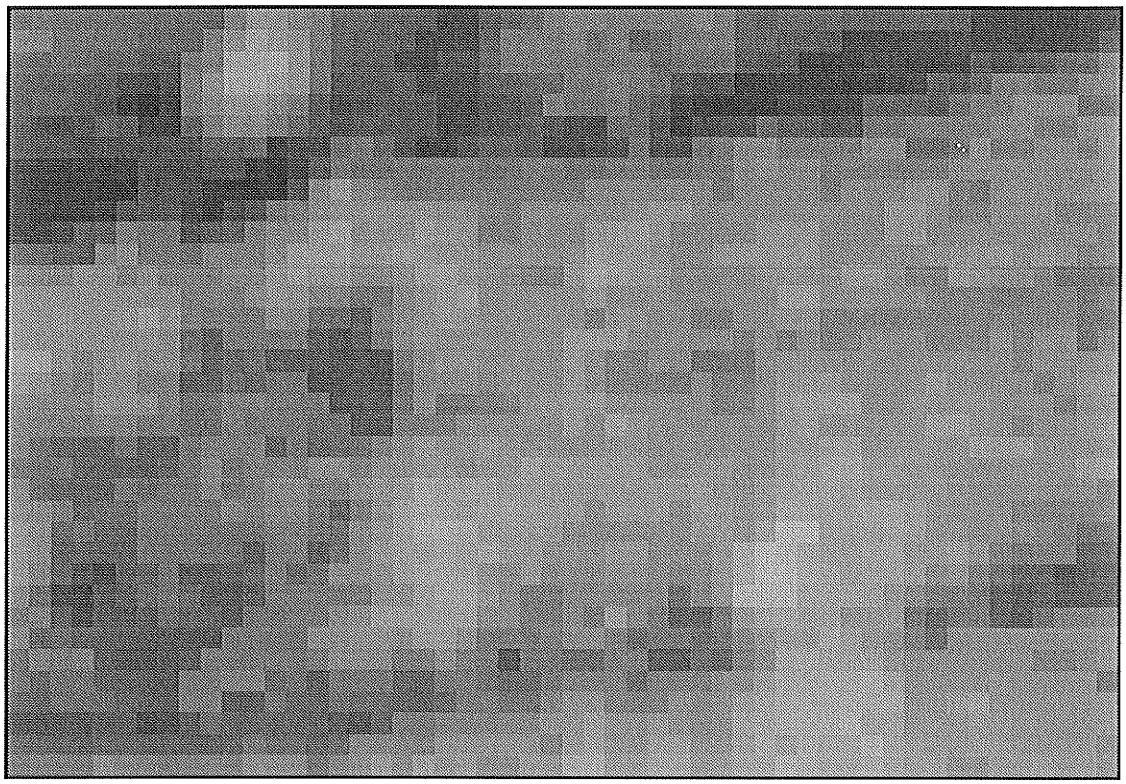


Fig. 3.1(b) Test image 3.

The edge detection methods that were tested represent a general overview of existing techniques. To begin, section 3.2.1 describes the earlier approaches to edge detection that made use of the gradient [GONZ,DAV1,ROSE2]. These ideas did not take into account noise immunity. Section 3.2.2 discusses the Prewitt and Sobel operators which make use of averaging to provide a small amount of noise immunity [ROSE2,LIND]. Two iterative methods are discussed in section 3.2.3. One method makes use of the gradient [EBER], the other makes use of local averaging [KASV]. Both methods also have a second iterative stage that thins the edges.

Section 3.4.4 discusses a method that uses a multiple template matching scheme to achieve edge detection at more than one scale [BRZA]. This method provides good localization as well as noise immunity. A different approach to edge detection is accomplished by using the difference of two operators, one linear and the other non-linear. A method which uses the difference between a low-pass filter and a median filter is discussed in section 3.2.5 [DEFE].

Section 3.2.6 deals with a rather unique edge detection method that makes use of the concept of entropy [SHIO]. In section 3.2.7, two edge detection methods that work in the frequency domain rather than the spatial domain are introduced. The first is an optimally recursive algorithm [DER1,DER2], and the other uses an operator created by taking the Laplacian of the Gaussian [MARR,HILD].

Finally, section 3.3 describes three methods of performing colour edge detection. The first two methods apply the idea of the gradient to a colour image [ZENZ,CUMA]. The third method is an extension to entropy edge detection in a grayscale image [SHIO].

As a matter of convention, when referring to a pixel in an image, a Cartesian coordinate system with a reversed Y axis is used. This arrangement was adopted because the window system used for the implementation has such a coordinate system. In all the

following examples, the X axis increases from left to right, and the Y axis increases from top to bottom.

3.2 Grayscale Edge Detection

3.2.1 Gradient

A common and simple method used for edge detection is the *gradient.* ([GONZ], [DAV1], [ROSE2]). The gradient vector **G** of a function *f*(x,y) is defined as

$$\mathbf{G}[f(x,y)] = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right).$$

Derivatives taken in any pair of perpendicular directions can be used [ROSE2]. The magnitude of this vector is given by

$$|\mathbf{G}[f(x,y)]| = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}$$

and the direction of this vector is given by

$$\arctan\left(\frac{\left(\frac{\partial f}{\partial y}\right)}{\left(\frac{\partial f}{\partial x}\right)}\right).$$

A large magnitude will represent a steep slope in the function, which is analogous to an edge (Fig. 3.2). Assuming an edge is present, its direction will be perpendicular to the direction of the gradient vector.

The above equations are for continuous functions. For a digital image, the gradient vector components can be approximated with first differences instead of first derivatives. An example pixel arrangement, shown in Fig 3.3(a), is the following:

$$(\Delta_x f)(x,y) = f(x,y) - f(x+1,y)$$
$$(\Delta_y f)(x,y) = f(x,y) - f(x,y+1)$$

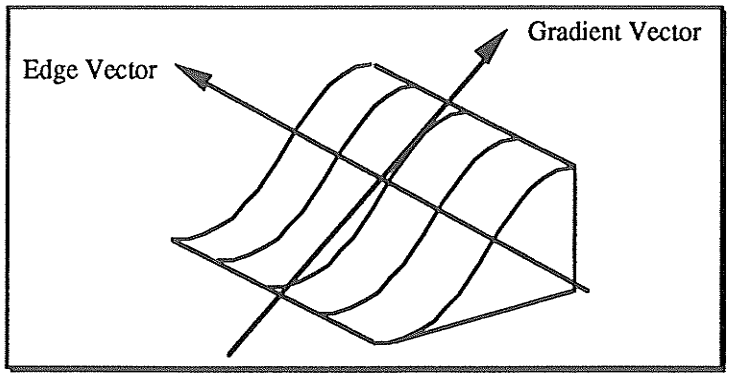


Fig. 3.2 A continuous slope showing the appropriate edge and gradient vectors.

These two differences can then be combined to compute the magnitude of the gradient. The problem with this arrangement is that the differences are not located symmetrically with respect to (x,y). They are instead centred at (x+1/2,y) and (x,y+1/2). This can be avoided by using

$$(\Delta_x f)(x,y) = f(x-1,y) - f(x+1,y) \qquad (1.1)$$
$$(\Delta_y f)(x,y) = f(x,y-1) - f(x,y+1)$$

which is symmetric around (x,y) (Fig. 3.3(b)). Another arrangement is the *Robert's gradient* (Fig. 3.3(c)):

$$(\Delta_x f)(x,y) = f(x,y) - f(x+1,y+1) \qquad (1.2)$$
$$(\Delta_y f)(x,y) = f(x+1,y) - f(x,y+1)$$

This is symmetric around (x+1/2, y+1/2).

In order to obtain the magnitude of the gradient, the directional differences given above must be combined in some way. One way is to use the square root of the sum of their squares. This method, however, introduces bias either for or against diagonal edges depending on whether method (1.1) or (1.2) is used. The method suggested in [ROSE2] makes use of the maximum of the two absolute differences

$$mag(\mathbf{G}) = \max\left(\left|f(x-1,y)-f(x+1,y)\right|,\left|f(x,y-1)-f(x,y+1)\right|\right) \text{ or}$$

$$mag(\mathbf{G}) = \max\left(\left|f(x,y)-f(x+1,y+1)\right|,\left|f(x+1,y)-f(x,y+1)\right|\right)$$

which eliminates the bias.

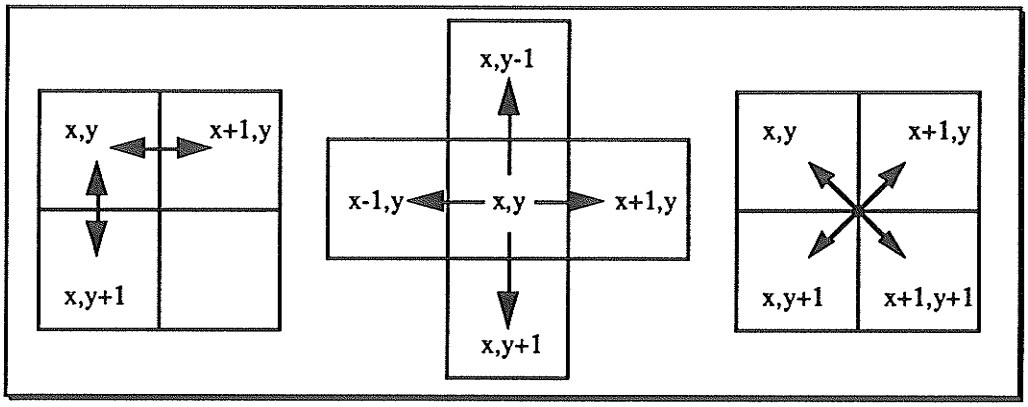


Fig. 3.3 a) non-symmetric gradient b) symmetric gradient c) Robert's gradient.

By thresholding the gradient magnitude, edges can be revealed. Larger threshold values will detect larger intensity changes, and therefore steeper edges, while smaller threshold values will detect edges with smaller slopes.

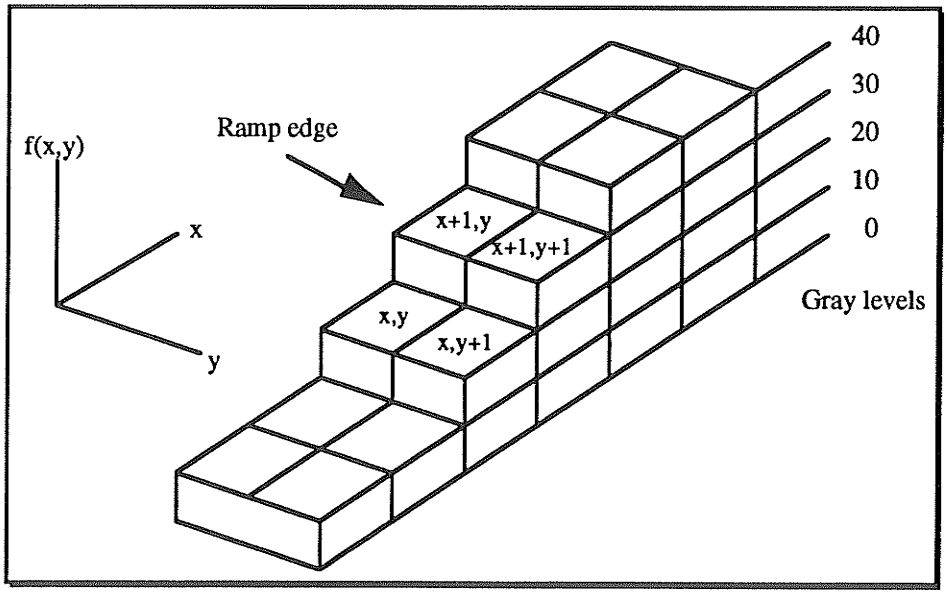


Fig. 3.4 Applying the Robert's operator to the ramp edge at (x,y), gives a value of 10. If the threshold value is greater than 10, then the edge will not be detected.

Gradient operators work very well for images that do not contain much noise. As soon as any reasonable amount of noise is introduced, they are very poor edge detectors. This is primarily due to their small size. Since they are sensitive to large changes in intensity, noise will be enhanced as well edges. Ramp edges with small slopes will be missed, unless the threshold value is very small (Fig. 3.4). Smaller threshold values will enhance more edges but will also result in increased noise enhancement.

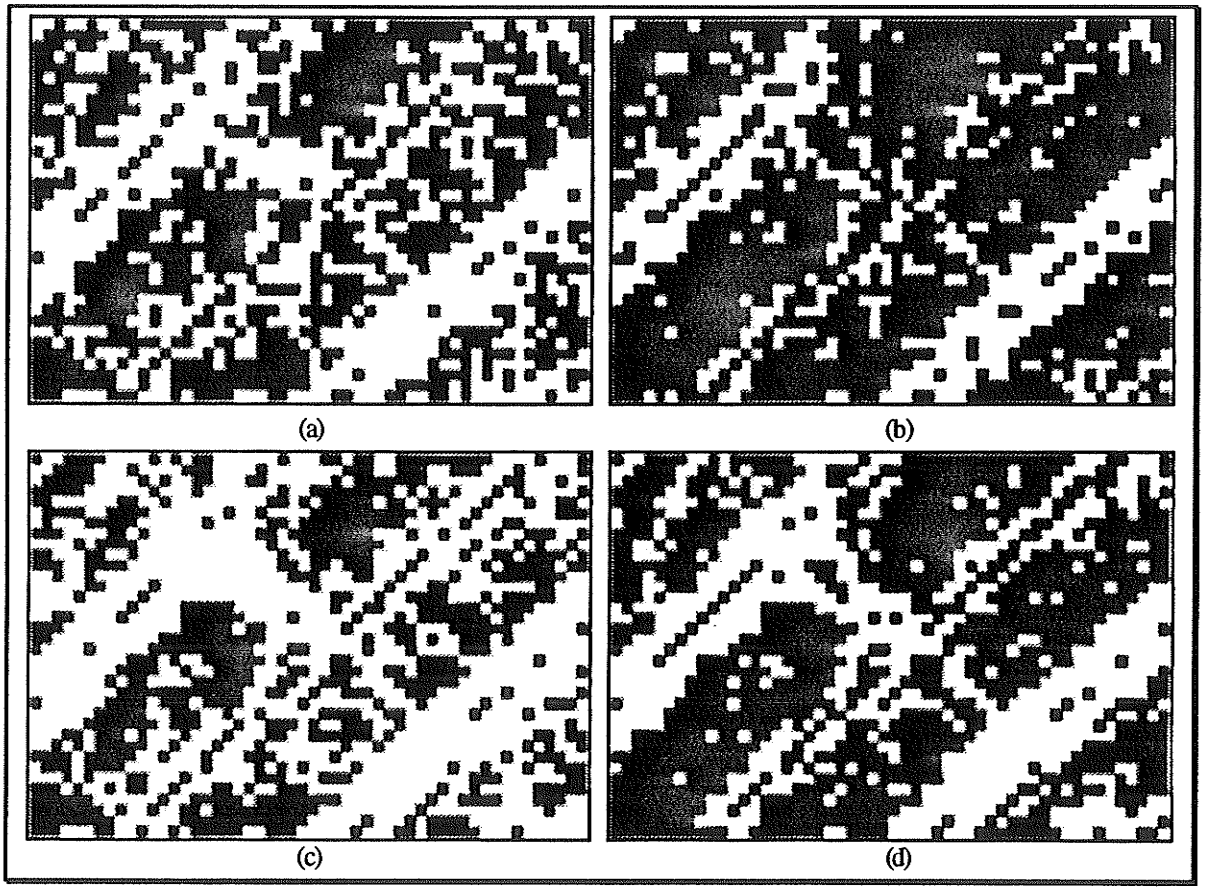


Fig. 3.5 Unsmoothed test image 1: a) Robert's: thresholded at 4; b) Robert's: thresholded at 5; c) Symmetric: thresholded at 4; d) Symmetric: thresholded at 5.

When the Robert's and symmetric gradient operators were applied to test image 1 with thresholding, their sensitivity to noise became apparent (Fig. 3.5). Depending upon what threshold value is chosen, a tradeoff between edge detection and noise immunity can be obtained. A larger threshold value decreases the amount of noise picked up, but reduces the number of edges detected. Faint rings become disconnected with rather large gaps. A

smaller threshold value will improve detection of the faint ring, but also increases the amount of noise picked up between rings. An advantage of gradient operators is that they will detect both sides of the ring. This is useful to obtain a measure of a ring's width.

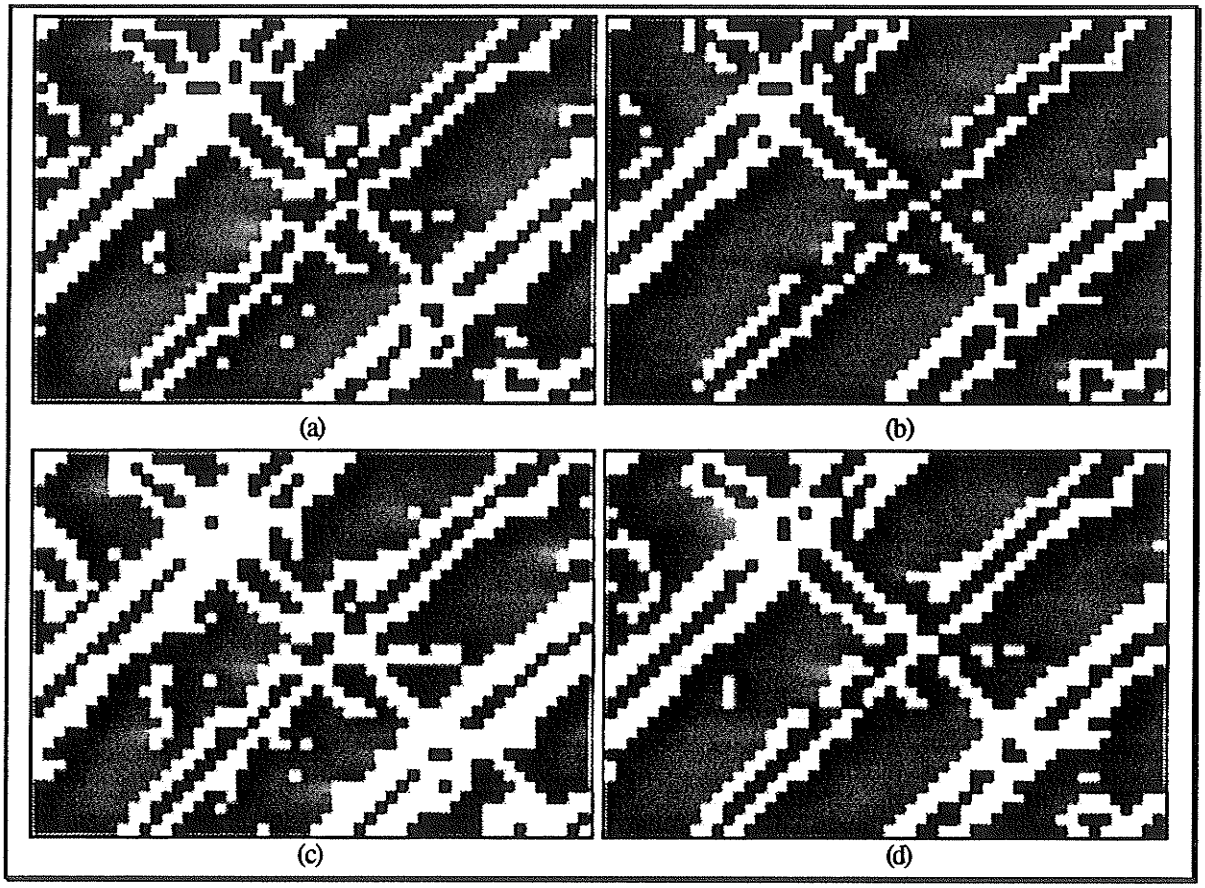


Fig. 3.6 Smoothed test image 1: a) Robert's: thresholded at 3; b) Robert's: thresholded at 4; c) Symmetric: thresholded at 3; d) Symmetric: thresholded at 4.

The success of gradient operators can be improved by smoothing the biscuit image with one of the edge preserving noise reduction methods mentioned in Chapter 2. The K-nearest noise reduction method was chosen because it produced the best results. By eliminating some of the noise, lower threshold values can be used, resulting in better edge detection (Fig. 3.6). This is desirable since lower threshold values will bring out the faint rings. However, even with such an improvement, gradient edge detection is not an appropriate method to enhance the rings in the test image. Certain threshold values may produce good results in one part of the image but poor results in others.

The symmetric gradient tended to produce better results than the Robert's gradient. Its larger size resulted in thicker edges and therefore fewer gaps. Thus, it provides more information than the Robert's gradient at lower thresholds.

3.2.2 Differences of Averages

By using differences of local averages, edge detection can be performed with the added benefit of some noise smoothing. Two implementations of this idea are the Prewitt and Sobel operators [ROSE2,LIND]. They are examples of unweighted and weighted averages, respectively.

Edge detection is achieved by computing two local averages and taking their difference. An example is shown in Fig. 3.7. Since averages are taken, the smoothing is done at the same time as the edge detection. The smoothing effect is somewhat less than that of a 3x3 low-pass filter because only three gray levels rather than nine are averaged.

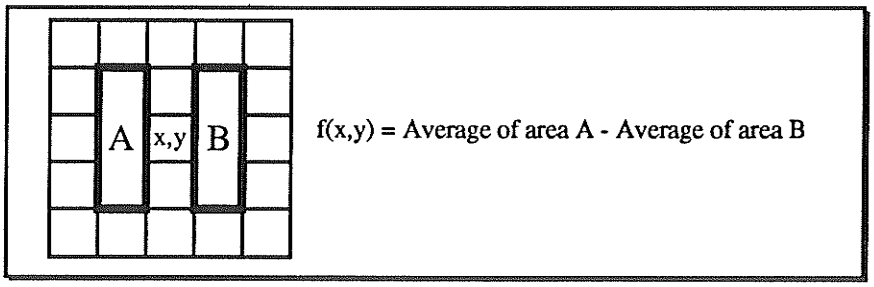


Fig. 3.7 One method of a difference of averages. The centre gray level is replaced by the difference of the averages of the two areas.

To obtain orientation independence, the operators are implemented as two convolution matrices which are perpendicular to each other. One matrix will obtain edge strength in the X direction, and the other matrix with give the edge strength in the Y direction. The operators are 3x3 matrices so that they are symmetric around (x,y). The Prewitt operator is implemented as the two convolution kernels

$$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

for X and Y respectively. These are unweighted averages. The orientation and magnitude of the edge slope are obtained in a similar manner to the gradient:

$$\text{Magnitude} = \left(X^2 + Y^2\right)^{\frac{1}{2}}$$

$$\text{Direction} = \arctan\left(\frac{Y}{X}\right)$$

As noted before, the actual edge is in a direction perpendicular to the above angle. This operator has a weaker response to edges that are not optimally oriented (i.e., edges that are not at 0 or 90 degrees) as opposed to those that are.

The Sobel operator attempts to correct this unbalanced response by giving a larger weight to pixels closer to the centre pixel. The resulting convolution kernels are

$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

for X and Y respectively. The magnitude and direction are obtained as with the Prewitt operator.

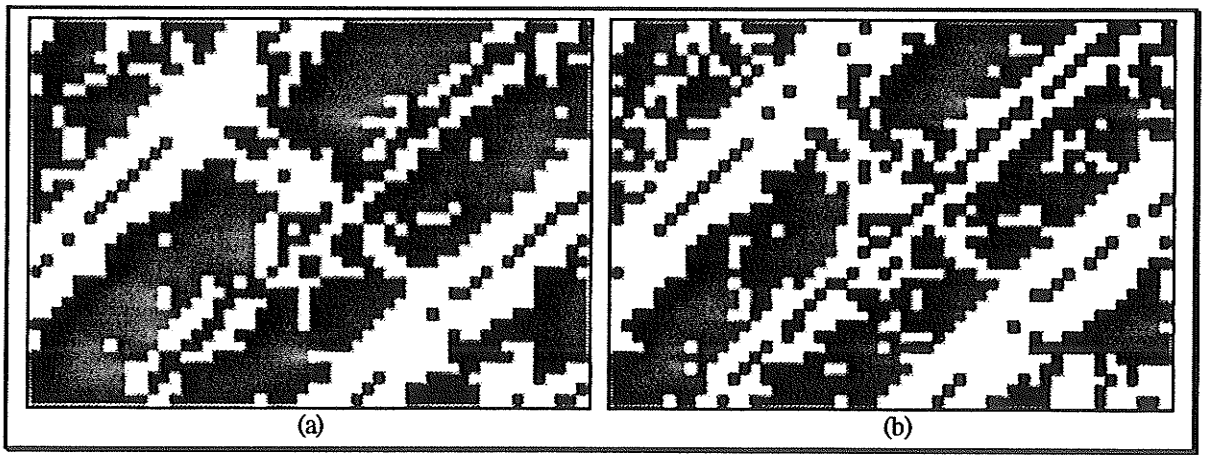


Fig. 3.8 Unsmoothed test image 1: a) Prewitt: thresholded at 4; b) Sobel: thresholded at 4.

Both operators exhibit improved results over the gradient operators. Their smoothing characteristics and larger size make them more suitable for noisy images. When applied to test image 1, both methods showed an improvement over the gradient methods when using

identical thresholds (Fig. 3.8). Their averaging characteristics resulted in less noise being detected. The Sobel operator tended to enhance more noise because of the extra weight given to the centre pixels, but it also detected the lighter ring better.

When applied to the smoothed test image, there was much less noise detected, but these methods did not perform any better than the gradient operators at the same threshold (Fig. 3.9). The Prewitt operator performed worse than the gradient operators, while the Sobel operator performed comparably to the symmetric gradient method. The better performance of the Sobel operator over the Prewitt operator can again be attributed to its weighted average. When given a smoothed image, a noise suppressing edge detection method is not really needed. This is why the Prewitt and Sobel operators did not perform any better upon the smoothed image.

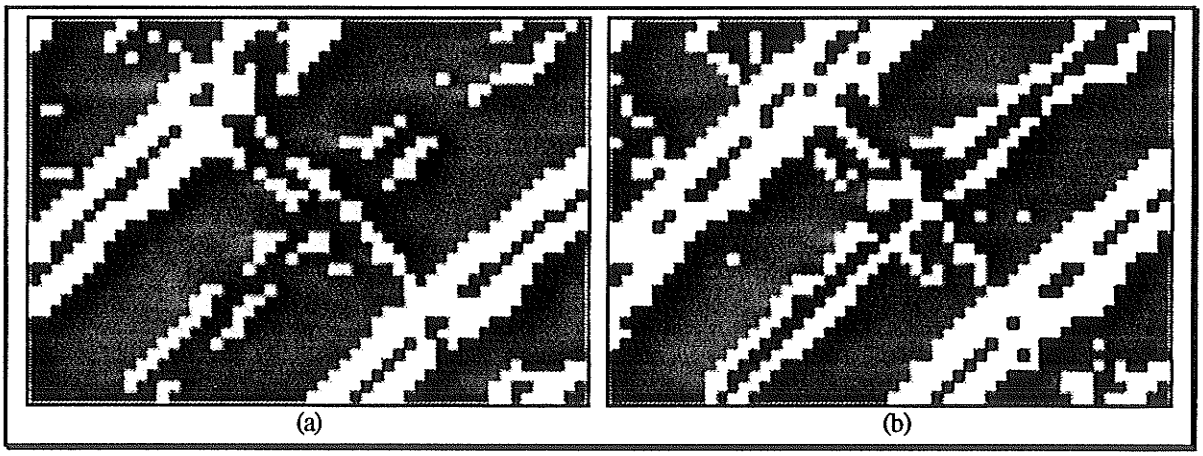


Fig. 3.9 Smoothed test image 1: a) Prewitt: thresholded at 4; b) Sobel: thresholded at 4.

3.2.3 Iterative Methods

A method presented in [KASV] suggests an iterative approach to edge detection. Each pixel (x,y) in an image f, is altered as follows:

$$f_k(x,y) = \begin{cases} f_{k-1}(x,y) - Avg_{k-1}(x,y) & \text{if } f_k(x,y) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $Avg_{k-1}(x,y)$ is a local average taken over an NxN area (centred on (x,y)). The above iterative definition states that a pixel value at step k is generated by subtracting the local average from the pixel value at step k-1. If the new pixel value is negative then it is set to zero. This process is iterated as many times as necessary. As with the Prewitt and Sobel operators, this iterative method has noise smoothing characteristics due to the averaging involved.

The edge detection is followed with an iterated edge thinning procedure. This thinning approach favours straight line segments by using four convolution matrices representing ideal lines in 45 degree increments. For a 5x5 local neighbourhood, the matrices would be

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 & -1 & -1 & 0 & 2 \\ -1 & -1 & 0 & 2 & 0 \\ -1 & 0 & 2 & 0 & -1 \\ 0 & 2 & 0 & -1 & -1 \\ 2 & 0 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix} \begin{bmatrix} 2 & 0 & -1 & -1 & 0 \\ 0 & 2 & 0 & -1 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & -1 & 0 & 2 & 0 \\ 0 & -1 & -1 & 0 & 2 \end{bmatrix}$$

for 0°, 45°, 90° and 135°, respectively. Convolving with each matrix gives sums S^0, S^{45}, S^{90} and S^{135}. The replacement gray level is then computed as follows:

$$f_k(x,y) = \begin{cases} \max(S_{k-1}^0, S_{k-1}^{45}, S_{k-1}^{90}, S_{k-1}^{135}) & \text{if } f_k(x,y) \geq 0 \\ 0 & \text{if max value is negative} \end{cases}$$

This procedure is then iterated upon the image that resulted from applying the first iteration procedure. Increasing the number of thinning iterations produces thinner edges.

Even though averaging is used, this method is still sensitive to noise. For a pixel in an area of low spatial frequency, there will most likely be a small difference between the local average and the pixel's intensity. The intensity of a pixel in an area of high spatial frequency may be quite different from the local average. This larger intensity difference will occur when the operator is centred along an edge, but will also occur when the operator is centred upon single noisy pixels (Fig. 3.10). Thus, noise is detected equally well as edges, and, as shown in the figure, it can produce a larger response than an edge.

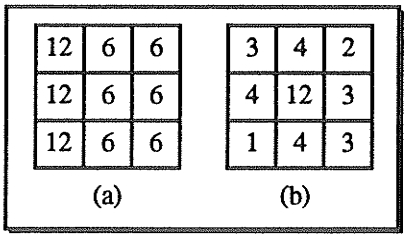


Fig. 3.10 An example of how noise can produce a larger response than an edge pixel. a) The centre pixel is next to an edge. In the first iteration it will be replaced by 2. b) The centre pixel represents noise. In the first iteration it will be replaced by 8.

This method performed very poorly on test image 1. The edge points tended to have very small intensities and therefore required a small threshold to enhance them. Any threshold value above 1 eliminated most of the detail while a threshold value equal to 1 enhanced too much detail (Fig. 3.11). The edge detected image had a large amount of noise, which could not be thinned properly. Without any well defined edges, the thinning procedure simply enlarged any "holes" formed by non-edge points (Fig. 3.12(a-c)).

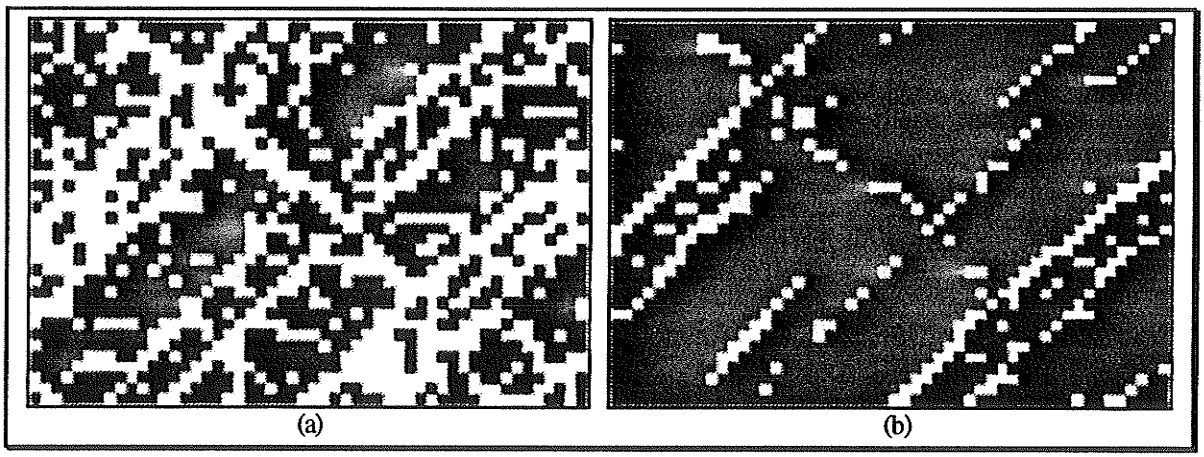


Fig. 3.11 Smoothed test image 1 with one iteration of edge detection and one iteration of edge thinning. a) thresholded at 1 b) thresholded at 2.

As expected, better results were obtained by smoothing the image first. The effect of increased iteration of the edge detection phase is shown in Fig. 3.12(d-f). Each step produces fewer edge points, so only a few iterations were performed. By the third iteration, the ring edges are more distinguishable. Notice that both sides of the ring are detected.

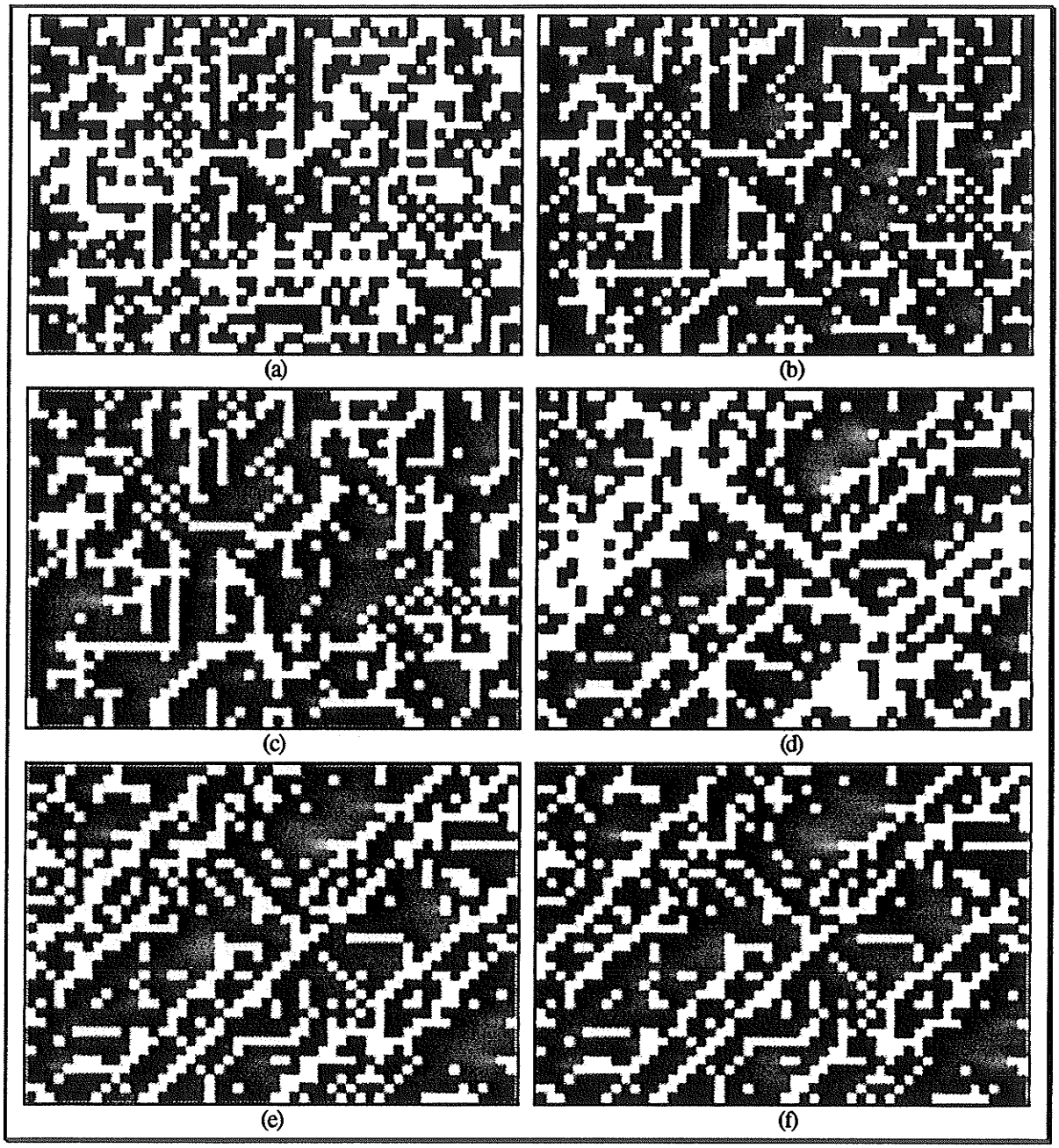


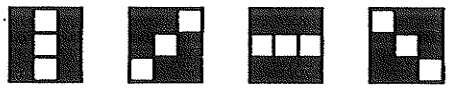
Fig. 3.12 (a)-(c) Unsmoothed test image 1 with three iterations of edge detection and one, two and three thinning iterations, respectively (thresholded at 1). (d)-(f) Smoothed test image 1 using only one thinning iteration and one, two and three edge detection iterations, respectively (thresholded at 1).

With the pre-smoothed image, the thinning phase did not thin the edges as expected. Increased iteration actually enlarged the detected ring edges by filling in empty spaces. This is the reverse of what occurred with the unsmoothed image, because there were detected

edges and less noise in this case. The thinning procedure can actually strengthen existing edges by lengthening them. This is advantageous because disconnected edges due to gaps in the rings may become connected. Unfortunately, even with the smoothed image, the thinning phase could not improve upon the edge detection phase, because there was too much noise present.

Another iterative method, proposed in [EBER], makes use of the edge vector for smoothing and thinning. To begin, the gradient vector is obtained at each pixel. The edge vector is then computed from the gradient and passed through an iterative smoothing phase, and then an iterative thinning phase.

There are two ways to perform the smoothing. In either case, a central edge vector is modified using six neighbouring vectors. The first method is *diffusion*, whereby a portion of the central edge vector is added to each neighbour vector. The second method is *infusion*, in which the central edge vector is replaced by the sum of itself and a weighted average of the neighbours. Which neighbouring vectors are used depends upon the direction of the edge vector to be modified. The central edge vector's direction is quantized to the nearest 45°, resulting in four possible neighbouring vector configurations (vector directions that differ by 180° use the same neighbourhoods):



The black pixels indicate which neighbouring vectors are involved in the smoothing operation.

After the smoothing operation has been iterated the required number of times, the next step is to perform the edge thinning. The thinning algorithm is based upon non-maxima absorption. Smaller edge vectors are absorbed by their larger neighbours. Each edge vector has a principal direction based upon which of the X and Y components is larger, giving four possible directions (i.e., up, down, left and right). There will be two neighbouring edge vectors at right angles to the principal direction (e.g., if the principal direction of the

edge vector at pixel (x,y) is up, the neighbouring vectors are at (x-1,y) and (x+1,y)). The central edge vector is compared with each of these neighbours only if they have the same principal edge direction. If this is indeed the case, and a neighbouring vector has a smaller component in the principal direction as compared to the central vector, then the central vector absorbs a portion of that neighbour. An example is shown in Fig. 3.13. Using an absorption factor of 0.5 ensures that a vector will immediately go to zero length if both of its neighbours are larger (because both neighbours will absorb half of each vector component).

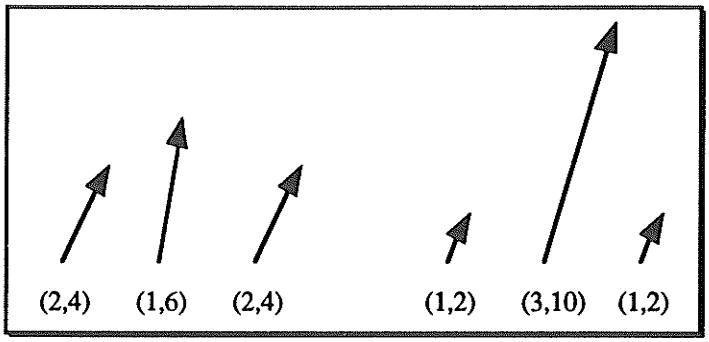


Fig. 3.13 The central vector has a principal direction in the vertical direction because the Y component is larger than the X component. Since the neighbouring vectors have the same principal direction but smaller Y components, they are absorbed by the central vector. In this case an absorption factor of 0.5 is used.

This iterative method is an improvement over the iterative averaging method. It has better noise immunity as well as better thinning abilities. For testing, the infusion smoothing method was used because diffusion produces much more ragged edges [EBER]. When applied to both test image 1 and the pre-smoothed version, it was found that increased iteration of the smoothing step produced much thicker edges (Fig. 3.14). The reason for this is that neighbouring groups of edge vectors have relatively the same direction and will therefore reinforce one another.

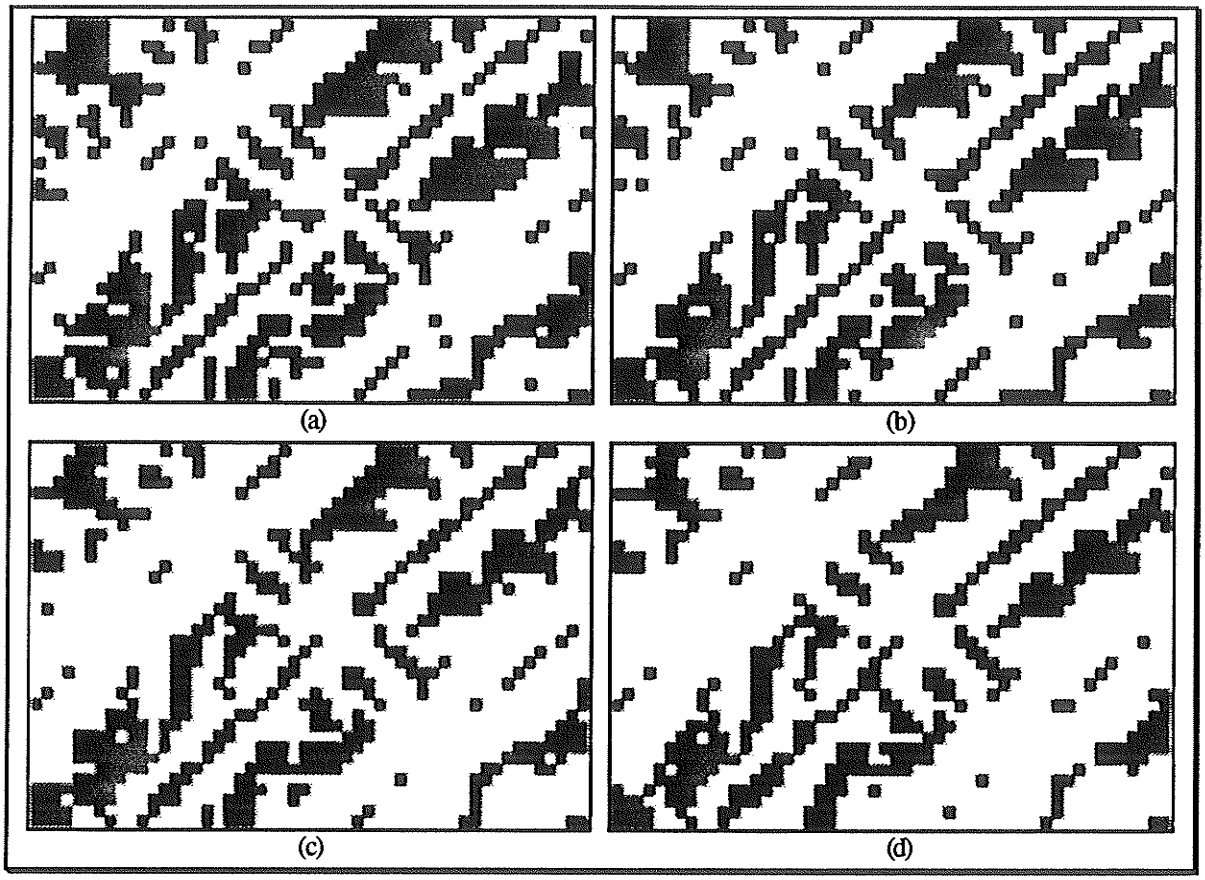


Fig. 3.14 Smoothed test image 1 with one iteration of thinning and one to four iterations of edge smoothing for (a)-(d), respectively (thresholded at 8).

Application to the smoothed test image resulted in less noise than the non-smoothed image, but in either case, the noise reduced the effectiveness of the thinning phase. Edges were thinned appropriately, but the noise interfered and caused false edges to appear. Fig. 3.15 illustrates the effects of increased thinning. The edges of both of the darker rings as well as the faint ring can be identified immediately. As the thinning proceeds, the presence of the false edges gives the image a fibrous appearance. This proves to be the drawback of this edge detection method. The ring edges are not separated from one another completely because of the false edges. This poses a problem when edge tracking is performed.

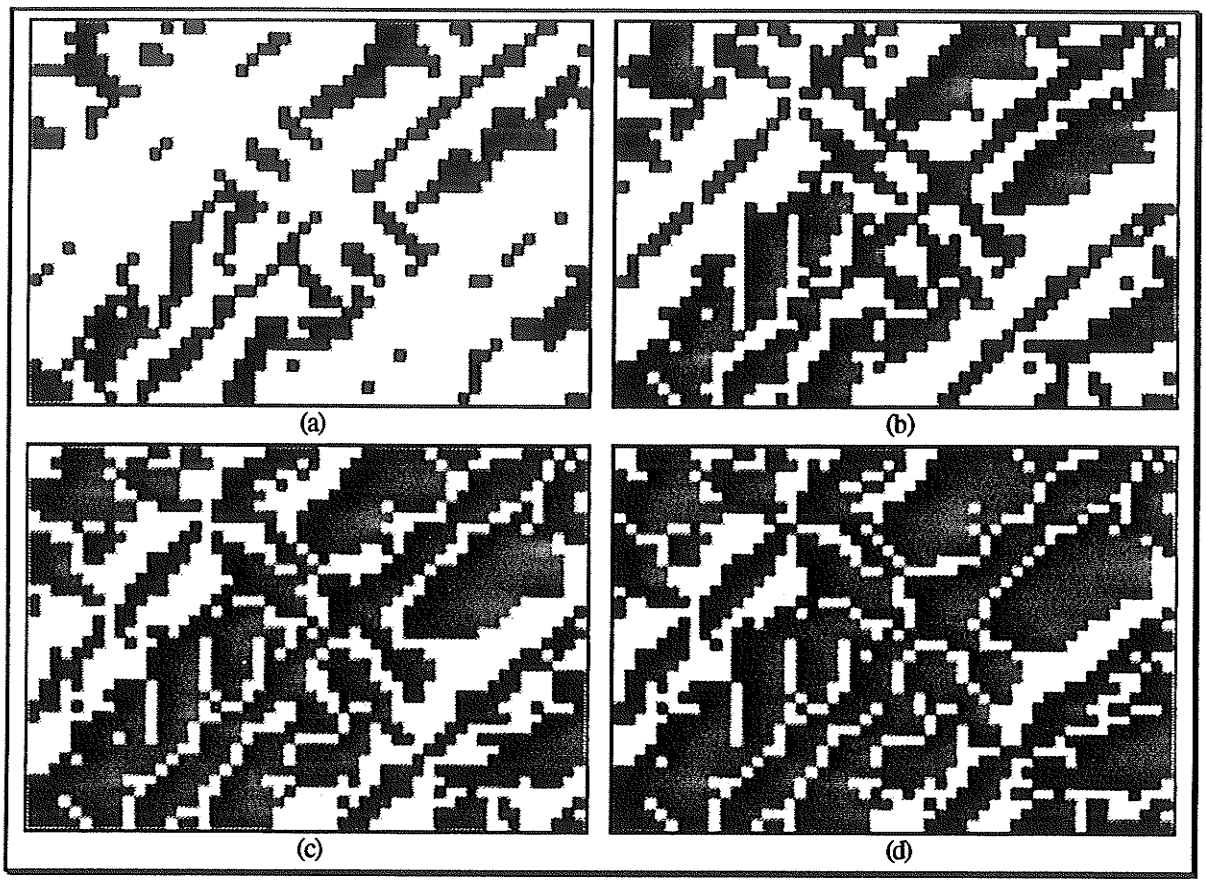


Fig. 3.15 Smoothed test image 1 with four iterations of edge smoothing and one to four iterations of thinning for (a)-(d), respectively (thresholded at 8).

3.2.4 Rule Based Multi-template

The previously described edge detection methods had a fixed resolution depending upon how large of an area was used. This initial condition would dictate what size of edge could be detected. By using different sized operators at each point in an image, and combining the results, one obtains an edge detector that is sensitive to edges of various sizes and works well in noisy areas. An edge detector of this type is described in [BRZA].

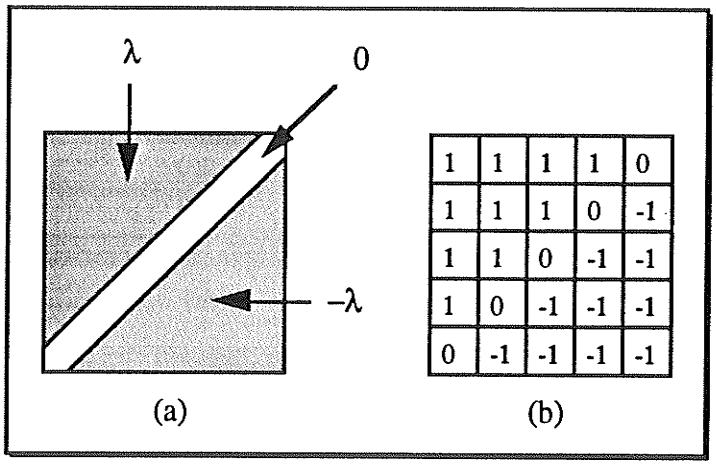
This method makes use of multiple templates of different sizes. Each template is a matrix of size NxN where N is an odd number. The smallest template size provides localization of edges, the intermediate sizes provide stabilization, and the largest size provides noise immunity. The templates are constructed to model ideal edges in different

orientations. As an example, to model edges at angles in 45° increments, eight templates for each size would be needed. If there were three different sizes, a total of 24 templates would be needed. If signed edge direction is not important, then only half the number of templates are needed due to the symmetry between edge templates that differ by 180°. A template that models a 45° edge is shown in Fig. 3.16(a). For simplified calculations, $\lambda = 1$ was chosen. Fig. 3.16(b) shows an example 5x5 template matrix.

Edge strength for a template is determined by the *cross-correlation coefficient.* For a template matrix T of size NxN centred upon a point (x,y) in a discrete image *f*, the cross-correlation coefficient $\gamma(x,y)$ for this point is computed with the following equation:

$$\gamma(x,y) = \sum_{-N/2}^{N/2} \sum_{-N/2}^{N/2} f(x+i, y+j)T(i,j).$$

Since the template matrices are of odd size, (i,j)=(0,0) corresponds to the centre of the template matrix T. The largest cross-correlation coefficient for a set of directional templates of a particular size, indicates the edge orientation. The cross-correlation coefficient should increase as the template size increases. However, in some situations this does not hold true, particularly at corners and closely spaced parallel edges. This does not pose a problem when edge detecting rings because such features either do not occur, or are not important.



1	1	1	1	0
1	1	1	0	-1
1	1	0	-1	-1
1	0	-1	-1	-1
0	-1	-1	-1	-1

Fig. 3.16 a) Ideal template modeling a 45° edge. b) 5x5 matrix form of template.

To determine whether a pixel is part of an edge, a rule-based method is used. Each point in the image is assigned a confidence value (C), an edge strength and an edge orientation. The confidence value indicates the likelihood of a pixel being part of an edge. The confidence values are assigned based on two sets of rules that utilize the cross-correlation coefficients obtained from the application of each template. The magnitude relationship ensures that the cross-correlation coefficient increases properly as the template size increases. The orientation relationship ensures consistency in the detected edge orientation between different template sizes. These two sets of rules are [BRZA]:

- Magnitude Relationship:

  Given: M template sizes, where the cross-correlation coefficient of a template $T_{k,i\pi/4}$ is $\gamma_{k,i\pi/4}$ for k = 1, ..., M and i = 1, ..., 8 (representing eight directional templates per size)

  - Rule 1: $\min(|\mu_1|, |\mu_2|, ..., |\mu_M|) > 0$, where $\mu_k = \max_i(\gamma_{k,i\pi/4})$, i = 1, 2, ..., 8
  - Rule 2: $\min(\mu_r - \mu_s) > T_e$, r = 1, ..., M, s = 1, ..., r-1

    If $\lambda = 1$, then $T_e = 7T_{min}$, where T_{min} is the minimum intensity change to be detected. This value was arrived at by considering the minimum increment in the cross-correlation coefficient from one template size to the next.

- Orientation Relationship:

  Given M template sizes, with D_j denoting the edge orientation determined by the j^{th} template size, a pixel is assigned confidence:

  - C_1 if $D_1 = D_2 = ... = D_M$
  - C_2 if $D_1 = D_2 = ... = D_{M-1}$ and $|D_1 - D_M| = 45°$, or if $D_2 = D_3 = ... = D_M$ and $|D_1 - D_M| = 45°$
  - C_3 if $D_1 = D_2 = ... = D_{M-1}$ and $|D_1 - D_M| = 90°$, or if $D_2 = D_3 = ... = D_M$ and $|D_1 - D_M| = 90°$
  - C_4 if $D_1 = D_2 = ... = D_k = D_{k+2} = ... = D_M$ and $|D_k - D_{k+1}| = 45°$
  - C_5 if $D_1 = D_2 = ... = D_k = D_{k+2} = ... = D_M$ and $|D_k - D_{k+1}| = 90°$

- C_6 if $D_1 = D_2 = ... = D_{M-2}$ and $|D_M - D_1| = 45°$ and $|D_{M-1} - D_1| = 45°$,

  or $D_3 = D_4 = ... = D_M$ and $|D_1 - D_M| = 45°$ and $|D_2 - D_M| = 45°$

- C_7 if $D_1 = D_2 = ... = D_{M-2}$ and $|D_M - D_1| = 90°$ and $|D_{M-1} - D_1| = 45°$,

  or $D_3 = D_4 = ... = D_M$ and $|D_1 - D_M| = 90°$ and $|D_2 - D_M| = 45°$

The confidences are related such that $C_{k-1} = C_k$ + constant, where $C_1 > C_2 > ... > C_7$.

The confidence assignments when using three template sizes are:

C_1	$D_1 = D_2 = D_3$
C_2	$D_1 = D_2$ and $\|D_1 - D_3\| = 45°$
C_2	$D_2 = D_3$ and $\|D_1 - D_3\| = 45°$
C_3	$D_1 = D_2$ and $\|D_1 - D_3\| = 90°$
C_3	$D_1 = D_2$ and $\|D_1 - D_3\| = 90°$
C_4	$D_2 = D_3$ and $\|D_1 - D_2\| = 45°$
C_5	$D_1 = D_3$ and $\|D_1 - D_2\| = 90°$
C_6	$D_1 \neq D_2 \neq D_3$ and $\|D_2 - D_1\| = 45°$, $\|D_3 - D_2\| = 45°$

The edge strength is computed by taking the average of the cross-correlation coefficients obtained from the templates of different sizes. So, for three template sizes, the edge strength at (x,y) would be

$$\frac{1}{3}[\gamma_1(x,y) + \gamma_2(x,y) + \gamma_3(x,y)].$$

This edge detection method works well in the presence of additive Gaussian noise. The directional templates can still detect edge locations and orientations correctly. Fig. 3.17 illustrates the results of applying the multi-template edge detection method to test image 1 as well as the pre-smoothed version. The pre-smoothed image has fewer jagged edges, and slightly less noise. In order to detect the edges of the faint rings, the edge detection threshold was set to detect low intensity changes. The images in Fig. 3.18 were generated by detecting intensity differences of at least 3 gray levels. By detecting low intensity changes, the larger intensity changes associated with darker rings were detected too well. What resulted was very thick ring edges that merged with neighbouring edges. The faint ring in Fig. 3.18(a) is clearly visible, but the rings in Fig. 3.18(b) are merged.

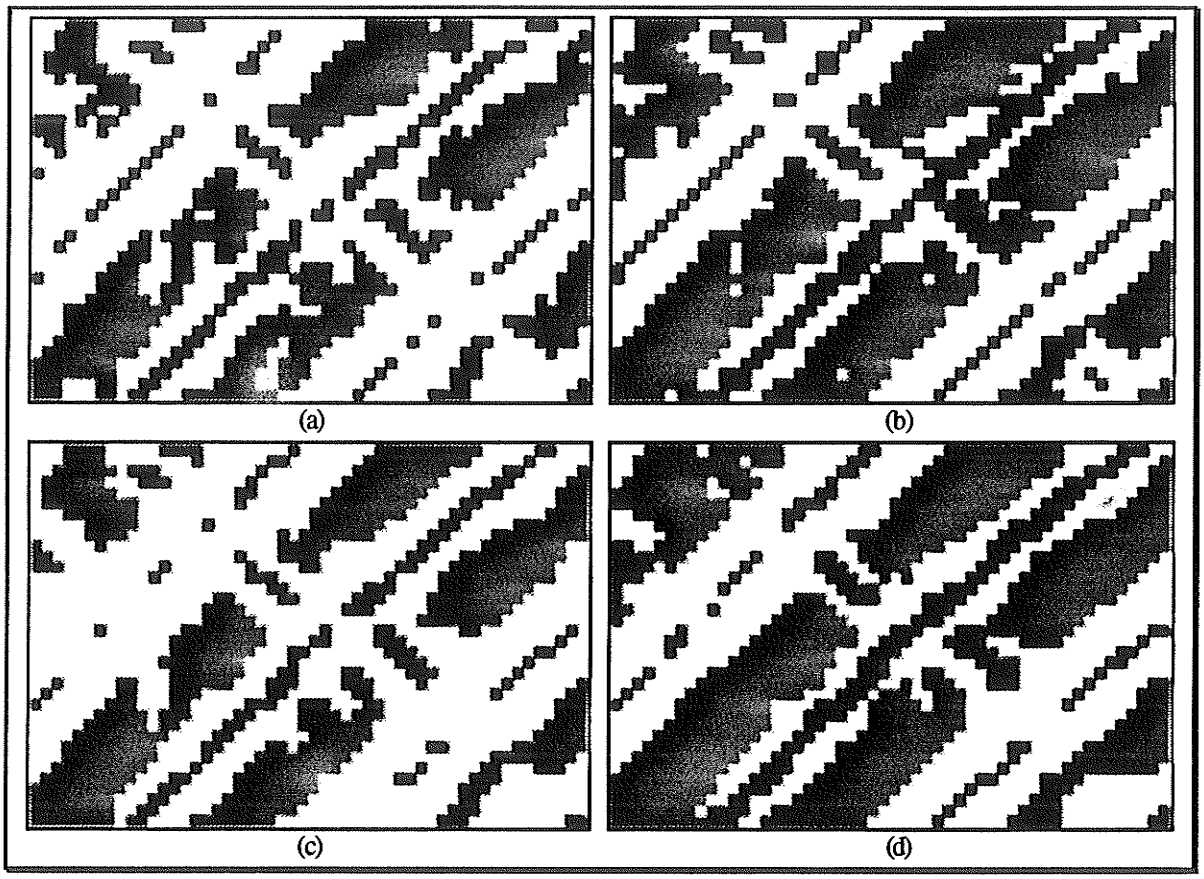


Fig. 3.17 In all images, white represents a non-zero confidence value. a) & b) Original test image 1: detection threshold of 3 and 4, respectively. c) & d) Smoothed test image 1: detection threshold of 3 and 4, respectively.

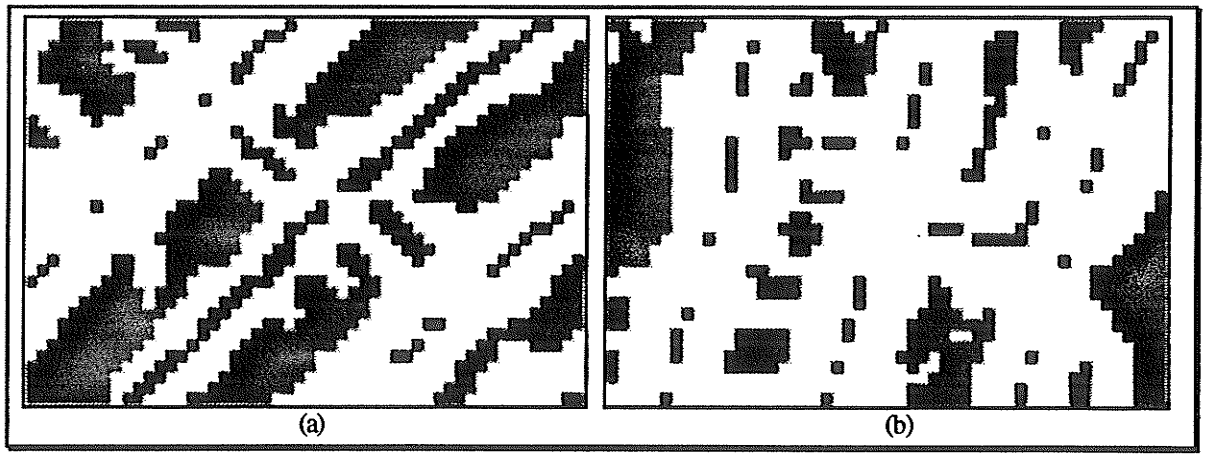


Fig. 3.18 Both images were pre-smoothed. Non-zero confidence values are white. Detection threshold set at 3. a) Test image 1 b) Test image 2.

The images in Figs. 3.19 and 3.20 were not thresholded based upon the confidence values. Instead, any pixel that was assigned a non-zero confidence value was given an intensity value of 255. The first two confidence values were the predominant ones in the image. Fig. 3.19 illustrates thresholding upon the confidence values. As can be seen, there is little change from Fig. 3.19(b) to Fig. 3.19(c). What this indicates is that the multi-template edge detector produces a large number of high confidence edge pixels and orientations. This is advantageous because the resulting information is very basic, and the thresholding process is really only needed to highlight edge pixels rather than discriminate amongst them.

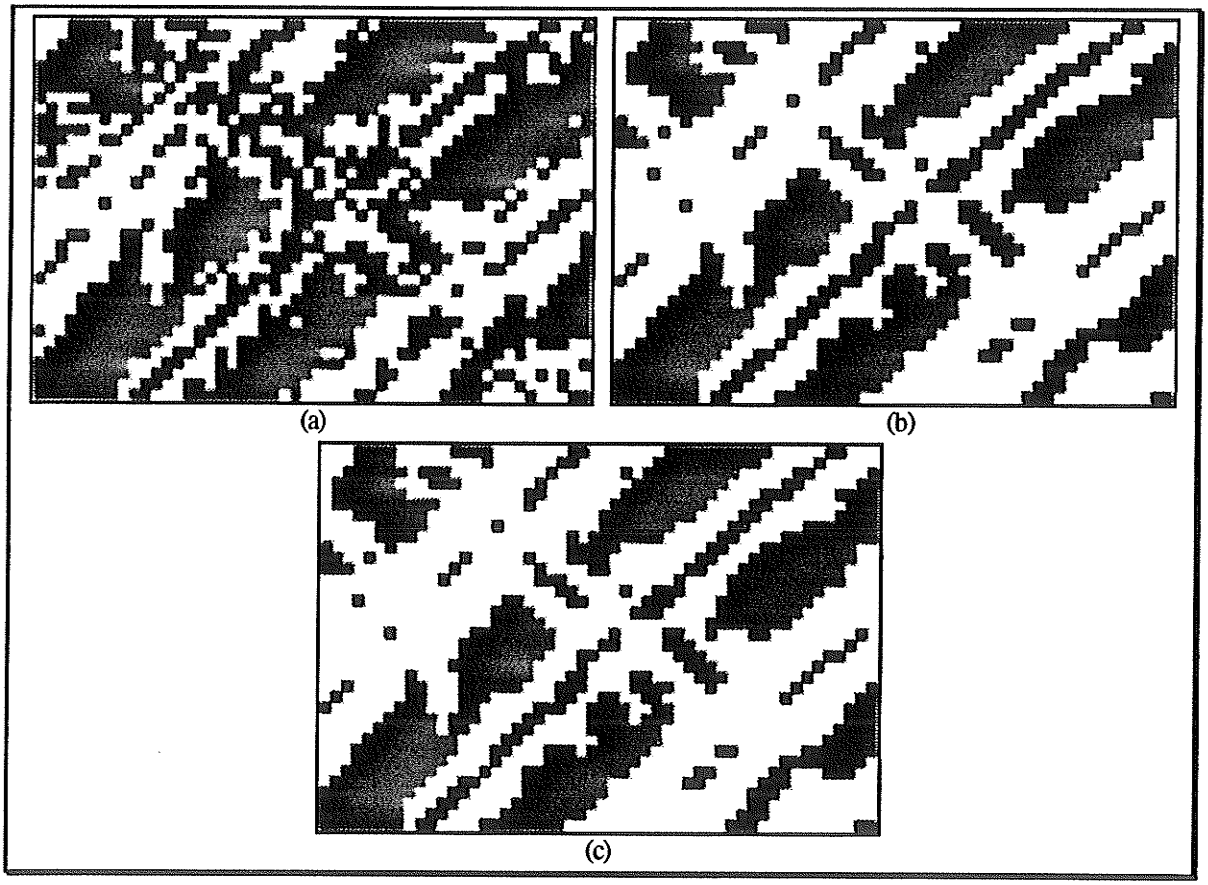


Fig. 3.19 Pre-smoothed test image 1 with the edge detection threshold set at 3. a) Only the highest confidence values. b) The first and second highest confidence values. c) The first, second and third highest confidence values.

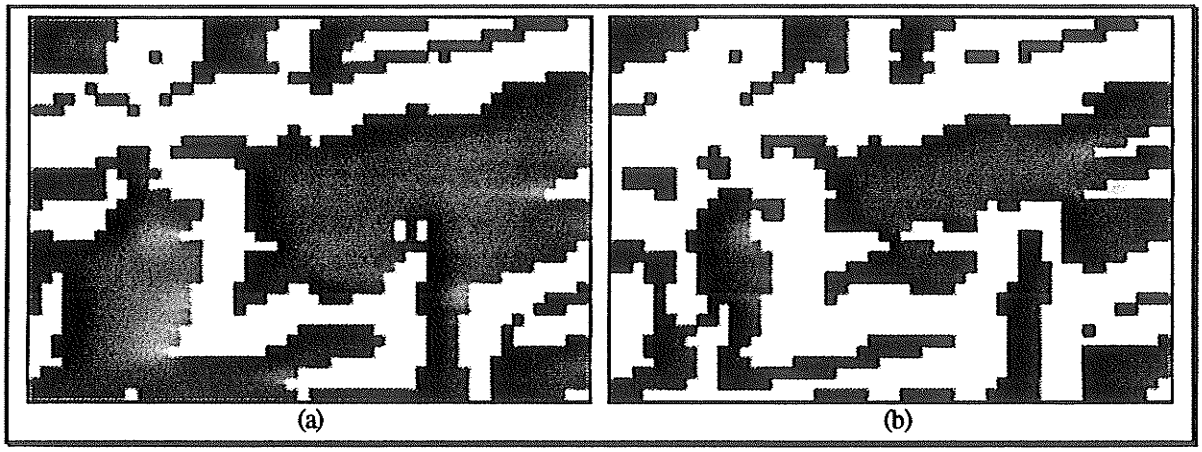


Fig. 3.20 Pre-smoothed test image 3. a) Detection threshold set at 4. b) Detection threshold set at 3.

Although the multi-template edge detection method works very well in detecting the ring edges of the first two test images, it does not perform as well on test image 3. An edge detection threshold of 4 does not detect the faint ring at all, while a threshold of 3 produces a large number of false edges and only partial detection of the middle ring (Fig. 3.20).

3.2.5 Median-Based Zero Crossing

There are many different ways of combining two separate filters to create an edge detector. As an example, the Prewitt operator mentioned above is the difference between two unweighted average filters. However, the regions covered by these two filters are disjoint. If these two regions overlapped exactly, the output would be zero.

One particular edge detector that makes use of overlapping regions is suggested in [DEFE]. These authors make use of the difference between a nonlinear filter and a linear filter. The operator is of the form

$$y = c\left(\mathrm{MED}(x_1,\ldots,x_m) - \sum_{i=1}^{m} a_i x_i \right) \text{ with the constraint } \sum_{i=1}^{m} a_i = 1$$

where MED denotes a median operation. A median filter was chosen because of its edge-preserving and noise-smoothing properties. The value c is a gain factor used to adjust the

height of the response. The linear filter used was simply an unweighted average since its output is close to that of a median filter for Gaussian white noise input.

Edges are detected by the presence of *zero-crossings*. If the first derivative of an image is taken, edges will occur at the peaks of the output because the slope of the image increases in the vicinity of an edge. If the second derivative is taken, then edges will occur at points where the output function crosses zero. Although it is not a second order differential, the linear median filter will produce zero-crossings in the same way. An example of applying the linear median filter to a 1-D edge can be seen in Fig. 3.21 [DEFE].

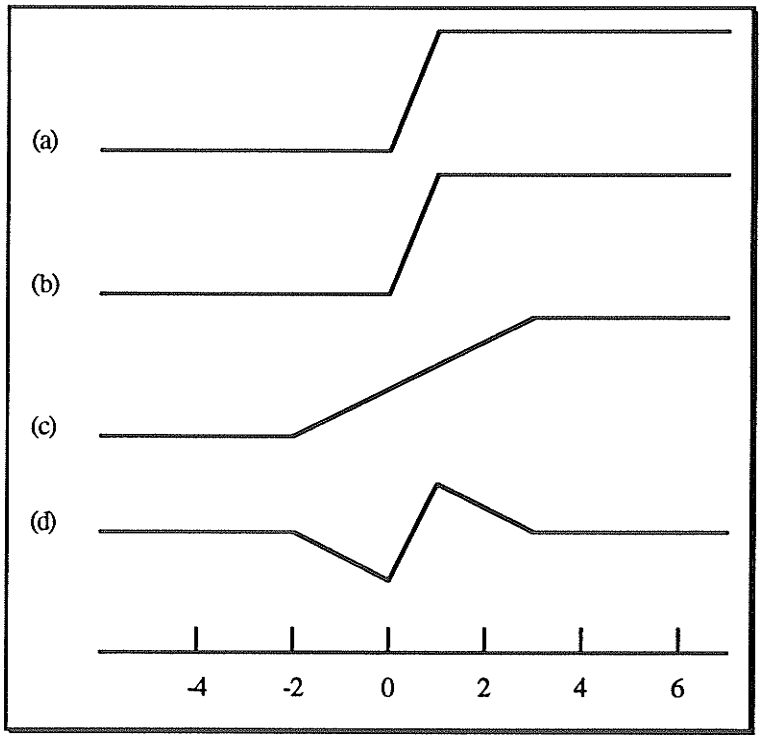


Fig. 3.21 Example of a nonlinear zero-crossing edge detector using the difference between a 5 point median and 5 point average filter: a) Input signal 1) b) Response to median filter c) Response to averaging filter d) Difference between median response and average response. Notice the zero-crossing.

This new filter performs better than linear methods under noisy conditions because image degradation is a non-linear function of noise.

The median filter supplies good localization because it senses an edge only when centred upon it. However, a 1-D median filter of width 2k+1 does not respond to a 1-D

edge less than k+1 in width. In the 2-D case this becomes more pronounced, so instead of a standard median filter, one can use a median filter specially designed to have high sensitivity to certain image details. An example of median filters attuned to certain edge orientations is shown in Fig. 3.22 [DEFE]. These filters can be combined into a tree structure resulting in a filter that is isotropic (i.e., insensitive to edge orientation).

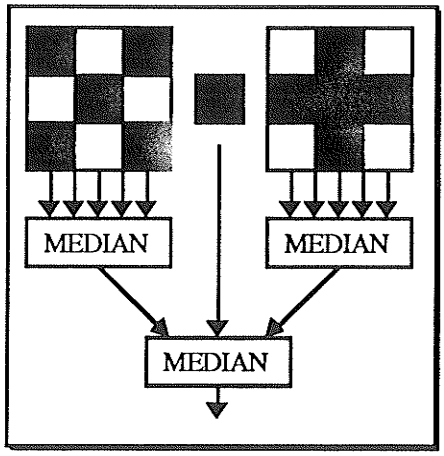


Fig. 3.22 Two 3x3 median filters sensitive to diagonal and vertical/horizontal lines respectively. The dark squares represent the pixels that are included in the operation. The results of the two filters are then fed into a three-point median filter along with the centre pixel.

Linear averaging substructures can be incorporated into the directional median filters resulting in linear-median hybrid filters. Fig. 3.23 shows an example of a tree structured linear-median hybrid filter with large averaging substructures [DEFE].

Defée et al [DEFE] tested the filters from Fig. 3.22 and Fig. 3.23, along with two other designs, for locality and noise immunity. As a comparison to another edge detector, they also tested a linear zero-crossing edge detector that was optimized for one-pixel wide edges (which is based on the Marr-Hildreth operator described below). Their conclusions indicate that the design in Fig. 3.23 produces the best results for detecting closely spaced edges.

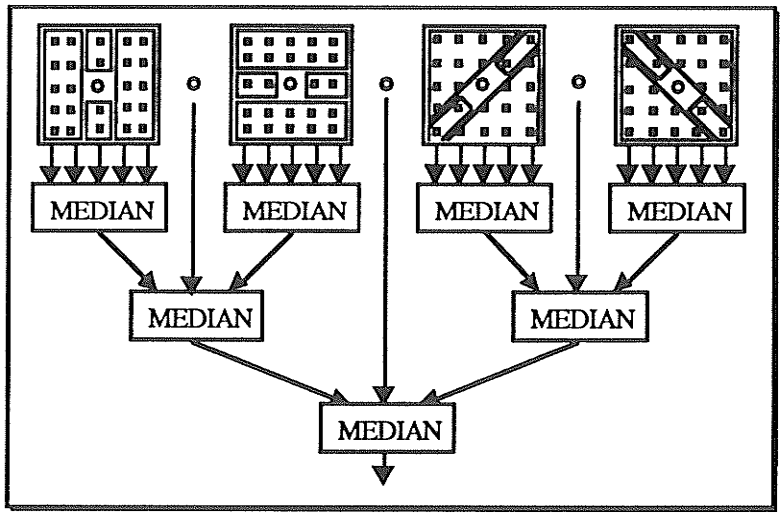


Fig. 3.23 A tree-structured linear-median hybrid filter using 5x5 directional filters. Grouped pixels are averaged.

This edge detection method was tested because it provides good locality with noise immunity. Its ability to detect closely spaced edges was not a deciding factor. At a 600 dpi scanning resolution, it is unlikely that the rings will be so close together that such a feature is needed.

Fig. 3.24 illustrates the results when applied to the test images and their smoothed versions. Even though the linear-median edge detection method already had a certain amount of noise immunity, it gave better results upon the smoothed image. This can be attributed to the K-nearest smoothing which enhances edge definition through the removal of noise. In the figure, a negative response is black, a zero response is gray, and a positive response is white. The rings produced a negative response because they are inverted roof edges. With test image 1, the middle ring was detected very well. In all images the edge detector showed excellent localization. However, there is still a large amount of noise, in the form of zero responses, which is apparent in Fig. 3.24(f). In this particular image, the detection of the middle ring was very poor.

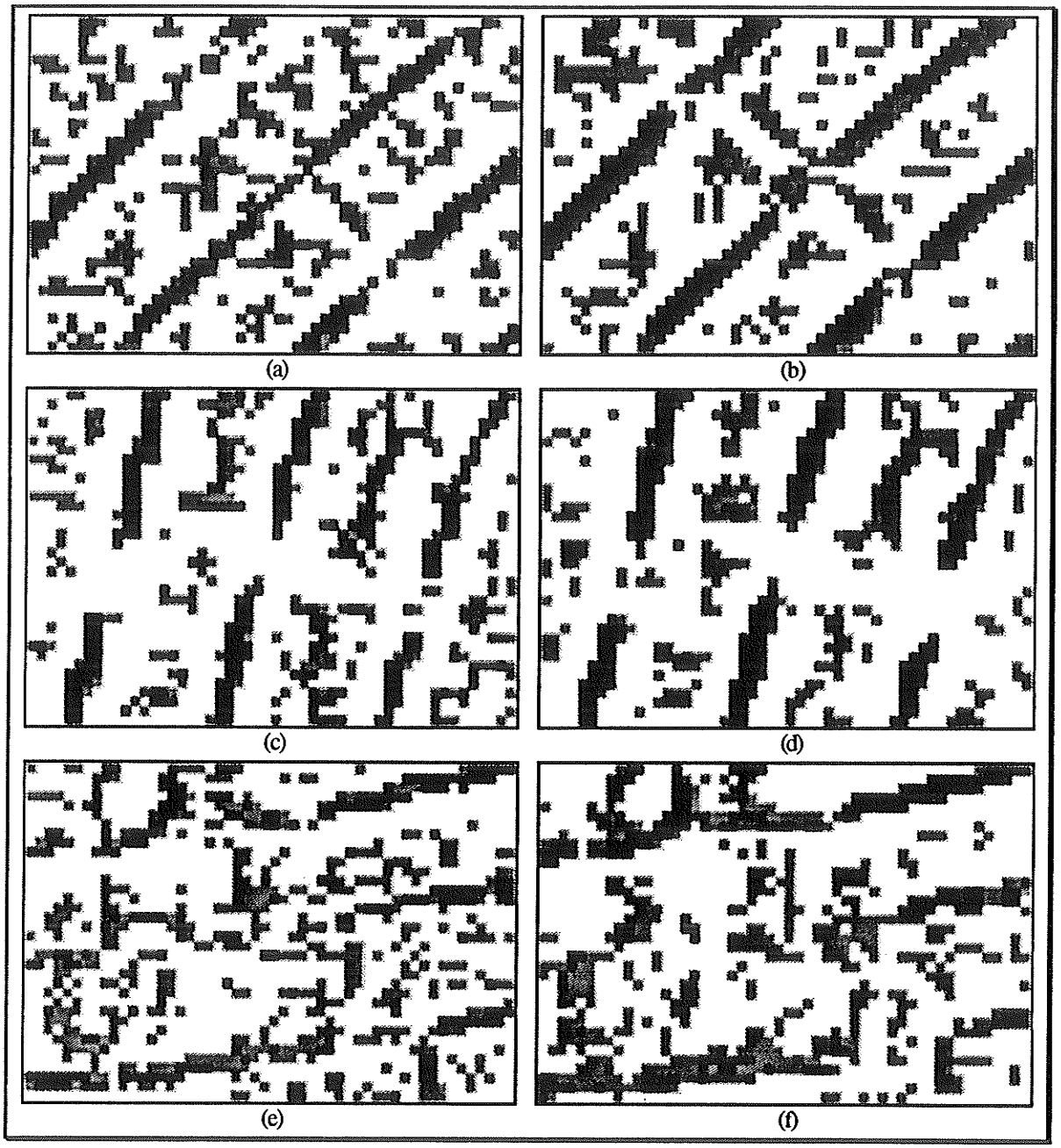


Fig. 3.24 Negative output shown in black, zero output in gray and positive output in white. (a), (c) & (e) Unsmoothed test images. (b), (d) & (f) Smoothed test images.

3.2.6 Entropy

A method for edge detection can be obtained through the use of *entropy* [SHIO]. Entropy refers to the amount of disorder present in a system. In the context of a digital image, the entropy will be large in the presence of smooth intensity variations (low spatial

frequency) and small in the presence of sharp intensity changes (high spatial frequency). Since the presence of sharp intensity changes is characteristic of edges, this method can be used to detect edges. It does not, however, provide edge direction information. This does not classify it as an edge enhancement method though, because it will produce an image containing only edges.

The entropy operator is applied to a finite region of an image centred upon some pixel with intensity I_0 and n neighbouring pixels with intensities I_1, ..., I_n. The entropy H of the brightness in the region is defined as:

$$H = -\sum_{i=0}^{n} \frac{p_i \log p_i}{\log(n+1)} \quad \text{where} \quad p_i = \frac{I_i}{\sum_{j=0}^{n} I_j}.$$

This equation is normalized so that $0 \leq H \leq 1$. The entropy values H are then thresholded to select edge pixel candidates. Low values of H will indicate the presence of an edge. This operator is sensitive to both the rate of change of brightness and the average brightness, which also makes it sensitive to noise. In order for the operator to be effective, the noise must be reduced beforehand. Any one of the edge preserving noise reduction methods mentioned in Chapter 2 can be used. The entropy operator is rotationally invariant which is advantageous due to its unbiased enhancement of edges in all directions, but cannot be used to obtain edge direction. Another method, such as the gradient operator, may be used for this purpose.

A problem that occurs with this edge detection method, is the choice of a thresholding value. The entropy levels that resulted were very close to 1.0. Thresholds of .999950 and larger were needed to highlight the detected edges. It should be noted that this is reverse thresholding. Values equal to and below the threshold prevail. A single threshold value did not enhance edges equally. As can be seen in Fig. 3.25, a threshold value that works well for one area of the image, may not work well for another. A higher threshold is needed to highlight faint rings, which results in over-enhancement and merging of darker rings. Fig.

3.25(d) shows test image 3 thresholded at .999990. Even at this threshold, the middle ring was not detected well.

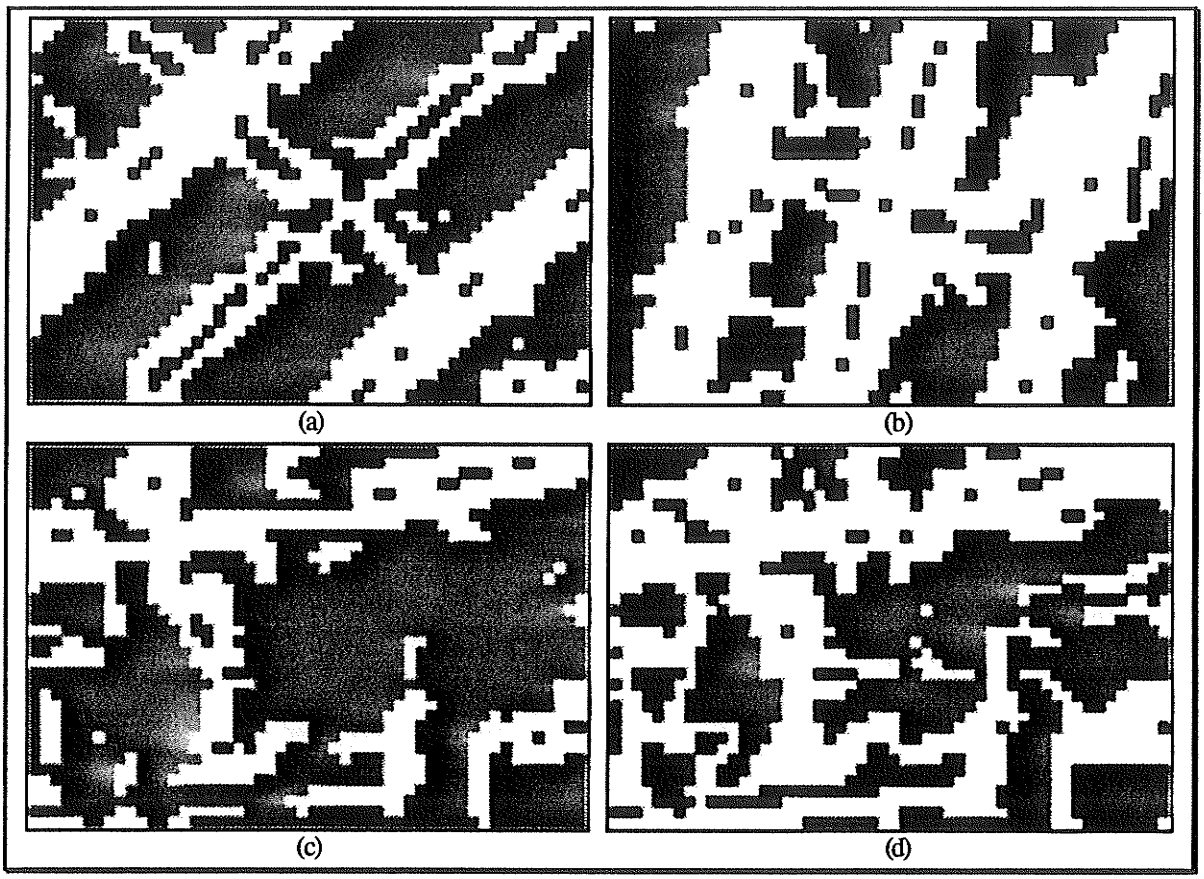


Fig. 3.25 All images were pre-smoothed. a) Test image 1 thresholded at .999980 b) Test image 2 at the same threshold c) Test image 3 at the same threshold d) Test image 3 thresholded at .999990.

3.2.7 Linear System Theory as a Basis for Edge Detection

3.2.7.1 Introduction to Linear System Theory

This section does not deal with edge detection directly. It is a preamble to the edge detection methods discussed in the following two sections. Some common terms are defined and a more precise definition of convolution is given.

With edge detection, one deals with the effects of filtering and spatial resolution. Filtering was discussed in Chapter 2. It is simply a process of letting specific information pass while discarding or altering the rest. An example is the low-pass filter which leaves

low spatial frequencies untouched while attenuating (filtering out) the higher spatial frequencies. Spatial resolution refers to the amount of detail examined by a particular filter. This is important in edge detection because the detected edges should not lie too far from the real edges.

Filtering processes and spatial resolution can be examined through the use of *linear system theory* which provides a strong mathematical basis [GASK]. A system is defined as anything that accepts an input and produces an output. A filtering process can be defined as a linear system. To relate the output to the input, a linear system can be expressed by the following equation (superposition integral):

$$h(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)d\alpha$$

where f is the input, g is the characteristic function of the system (or filtering function), and h is the output. This equation is referred to as the *convolution integral.* The presence of x-α is to impose a shift invariance upon the filtering function. This shift invariance implies that the filtering function can be applied upon any point x of the input function $f(x)$. Fig. 3.26 illustrates the concept of shift invariance and the convolution operation.

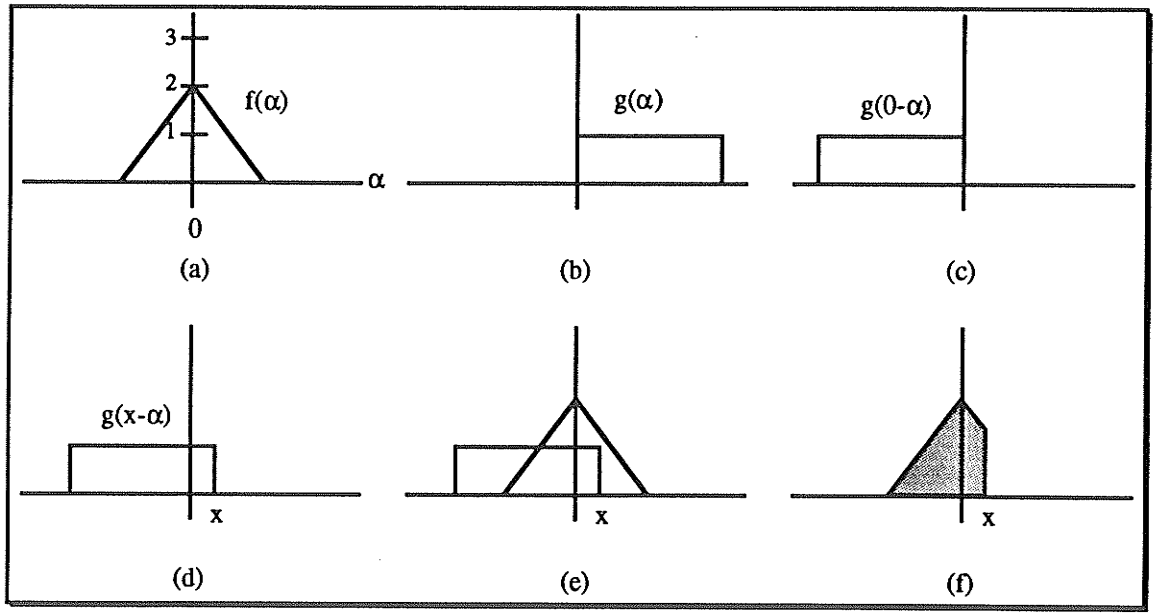


Fig. 3.26 The convolution process in one dimension. a) input function f(α) b) convolution function g(α) (i.e., characteristic function or filter function) c) g(α) reflected d) g(α) reflected and shifted e) functions superimposed f) product of functions.

Convolution as it pertains to image processing was discussed in Chapter 2. The above convolution integral generalizes to two dimensions as

$$h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u,v)g(x-u,y-v)dudv$$

The discrete form of this integral is

$$H(i,j) = \sum_m \sum_n F(m,n)G(i-m,j-n)$$

which can be recognized as the 2D digital convolution operation, where F is the input image, G is the convolution kernel and H is the output image.

To obtain the characteristic function of a linear system, an impulse function, the Dirac delta function, is introduced. It is defined by its integral property

$$\int_{-\infty}^{\infty} \delta(x)dx = \int_{-\varepsilon}^{\varepsilon} \delta(x)dx = 1 \qquad \delta(x) = 0 \text{ when } x \neq 0$$

where ε is an arbitrarily small number greater than 0. When the impulse function is used as input to a system, the output is the characteristic function of the system. This is illustrated by the following convolution operation:

$$\int_{-\infty}^{\infty} \delta(\alpha)f(x-\alpha)d\alpha = f(x-\alpha)\Big|_{\alpha=0} = f(x).$$

Since $\delta(\alpha) = 0$ when $\alpha \neq 0$, it follows that the integral is non-zero only at $\alpha = 0$. The impulse thus acts as the identity function. The characteristic function is referred to as the *impulse response* of the system because it is the system's response to an impulse at the input.

3.2.7.2 Optimal Recursive

Some of the edge detection methods introduced up to this point have not had very strong mathematical underpinnings. A more computational approach to the design of edge detectors was introduced in [CANN]. This method made use of certain predefined criteria that the edge detector must adhere to. The criteria stated by Canny are as follows:

1) Good detection. There should be a low probability of failing to mark real edge points, and low probability of falsely marking non-edge points. Since both these probabilities are monotonically decreasing functions of the output signal-to-noise ratio, this criterion corresponds to maximizing signal-to-noise ratio.
2) Good localization. The points marked as edge points by the operator should be as close as possible to the centre of the true edge.
3) Only one response to a single edge. This is implicitly captured in the first criterion since, when there are two responses to the same edge, one of them must be considered false. However, the mathematical form of the first criterion did not capture the multiple response requirement and it had to be made explicit.

Canny applied these criteria in the solution of an edge detector specifically designed for a step edge in Gaussian white noise. For consistency, the notation from [DER1] rather than [CANN], is used for the following equations. A step edge with input white noise variance ${n_0}^2$ can be modeled by the equation

$$I(x) = Au_{-1}(x) + n(x) \qquad \text{with } {n_0}^2 = n^2(x) \text{ for all } x$$

where $u_{-1}(x)$ is the step function, A is the step amplitude and $n(x)$ is the input noise [DER1]. The step function is defined as follows:

$$u_{-1}(x) = \begin{cases} 1 & x \le 0 \\ 0 & x > 0 \end{cases}$$

Canny's method of detecting edges was to convolve the input edge with a spatial antisymmetric function $f(x)$ (i.e., the impulse response) and mark edges at the maxima of the output $\theta(x_0)$ defined by

$$\theta(x_0) = \int_{-\infty}^{\infty} I(x) f(x_0 - x) dx.$$

The first criterion states that there should be good edge detection. Canny showed that this corresponds to maximizing the SNR, which is defined as the ratio of the edge detector

response to the step edge only, and the response to the square root of the mean-squared response to the input noise:

$$\text{SNR} = \frac{A}{n_0} \cdot \frac{\int_{-\infty}^{0} f(x)dx}{\left[\int_{-\infty}^{+\infty} f^2(x)dx\right]^{\frac{1}{2}}} = \frac{A}{n_0} \cdot \Sigma.$$

The range of the integral in the numerator is [-∞,0] because $u_{-1}(x) = 0$ for $x > 0$. Finding the impulse response $f(x)$ which maximizes Σ will correspond to finding the best operator for detection only [DER1].

The second criterion states that there should be good localization. Edges are marked at the local maximum of the operator response, and these marked edges should not lie too far from the actual edge. To achieve this, the variance σ^2 of the marked edge from the true edge, should be minimized. Canny showed that this corresponds to maximizing the localization criterion defined as

$$L = \frac{A}{n_0} \cdot \frac{|f'(0)|}{\left[\int_{-\infty}^{\infty} f'^2(x)dx\right]^{\frac{1}{2}}} = \frac{A}{n_0} \cdot \lambda.$$

This is the reciprocal of σ [CANN].

The third criterion stated that there should only be one response to an edge. The number of peaks in the response should be limited in order to reduce the probability of detecting more than one edge. In the absence of noise, only one edge will be marked. In the presence of noise, there may be many local maxima. The distance between peaks in the noise response of *f* denoted by x_{max}, is set to a fraction *k* of the operator width W [CANN]:

$$x_{\max} = k \cdot W = 2 \cdot \pi \cdot \left[\frac{\int_{-\infty}^{\infty} f'^2(x)dx}{\int_{-\infty}^{\infty} f''^2(x)dx}\right]^{\frac{1}{2}}.$$

Canny combined the detection and localization criterion equations and the resulting product, $\Sigma \cdot \lambda$, is maximized under the constraint of the third criterion. By expressing the criterion as a composite functional, he found that this leads to the solution

$$2f(x) - 2\lambda_1 f''(x) + 2\lambda_2 f''''(x) + \lambda_3 = 0.$$

The general solution in the range [0,W] can be written as

$$f(x) = a_1 e^{\alpha x} \sin \omega x + a_2 e^{\alpha x} \cos \omega x + a_3 e^{-\alpha x} \sin \omega x + a_4 e^{-\alpha x} \cos \omega x + C \qquad (2.1)$$

subject to the boundary conditions

$$f(0) = 0 \quad f(W) = 0 \quad f'(0) = S \quad f'(W) = 0$$

where S is an unknown constant equal to the slope of $f(x)$ at the origin.

The four constants a_1 through a_4 can be determined because of the four boundary conditions. Since $f(x)$ is antisymmetric (i.e., $f(x) = -f(-x)$) the solution can be extended to the range [-W,+W]. Canny found that the largest k that could be obtained was 0.58, corresponding to a performance index $\Sigma \cdot \lambda$ of 1.12. The shape of the resulting operator is approximately the first derivative of a Gaussian [CANN]:

$$f(x) = -\left(\frac{x}{\sigma^2}\right) e^{\frac{-x^2}{2\sigma^2}}$$

which has a performance index of 0.92 with k=.51. The first derivative of a Gaussian is thus about 20% worse than the optimal operator.

Canny's work was with a finite impulse response (FIR) filter $f(x) = Sxe^{-\alpha|x|}$. The range of this filter is [-W,+W]. If the operator covers an infinite range, [-∞,+∞], it is referred to as an infinite impulse response (IIR) filter. Canny's criteria was used by Deriche [DER1] to derive an edge detector based upon an IIR filter rather than a FIR filter.

The same differential equation is obtained, which leads to the same solution as (2.1). For an IIR filter, the boundary conditions become

$$f(0) = 0 \quad f(+\infty) = 0 \quad f'(0) = S \quad f'(+\infty) = 0,$$

and by applying these conditions to (2.1) the following solution is obtained:

$$f(x) = \frac{S}{\omega} e^{-\alpha|x|} \sin \omega x. \qquad (2.2)$$

The product $\Sigma \cdot \lambda$ is maximum for $\omega = 0$, and since $\sin(\omega x)/\omega \approx x$ for very small ω, equation 2.2 can be rewritten as

$$g(x) = Sxe^{-\alpha|x|}.$$

The variable α can be adjusted to give the desired localization or SNR. Decreasing α will produce a better SNR while reducing the localization, and vice versa.

Deriche derived this operator as a recursive filter in the following way. He defined $G(Z)$ as the Z-transform [OPPE] of the samples $g(n)$ of $g(x)$:

$$G(Z) = \sum_{n=-\infty}^{\infty} g(n) Z^{-n}$$

where Z is a complex number. He then split the discrete impulse response into two halves $g_-(n)$ and $g_+(n)$ such that

$$g(n) = g_-(n) + g_+(n) \quad \text{for } n = -\infty, \ldots, \infty$$

$$g_-(n) = \begin{cases} 0 & n \geq 0 \\ Sne^{\alpha n} & n < 0 \end{cases} \qquad g_+(n) = \begin{cases} Sne^{-\alpha n} & n \geq 0 \\ 0 & n < 0 \end{cases}$$

Using the Z-transform and simplifying he obtained:

$$G(Z) = G_-(Z) + G_+(Z^{-1}) \text{ where}$$

$$G_+(Z^{-1}) = \frac{aZ^{-1}}{1 + b_1 Z^{-1} + b_2 Z^{-2}} \qquad G_-(Z) = \frac{-aZ}{1 + b_1 Z + b_2 Z^2}$$

$$a = Se^{-\alpha} \quad b_1 = -2e^{-\alpha} \quad b_2 = e^{-2\alpha} \quad \text{with } S = \frac{\left(1 - e^{-\alpha}\right)^2}{e^{-\alpha}}.$$

Finally, he showed that G_+ can be represented by a filter recursing from left to right and G_- can be represented by a filter recursing from right to left as follows:

$y^+(m) = x(m - 1) - b_1 y^+(m - 1) - b_2 y^+(m - 2)$ for m = 1, ..., M

$y^-(m) = x(m + 1) - b_1 y^-(m + 1) - b_2 y^-(m + 2)$ for m = M, ..., 1

$y(m) = a(y^+(m) - y^-(m))$ for m = 1, ..., M

where x(m) is the input (i.e., grayscale values) and y(m) is the output.

By taking the integral of g(x), Deriche defined a smoothing filter, $h(x)$, as follows:

$$h(x) = k(\alpha|x|+1)e^{-\alpha|x|}.$$

This filter is designed to smooth the output produced by the above filter. Using the above procedure, he again derived two recursive functions:

$y^+(m) = a_0x(m) + a_1x(m - 1) - b_1y^+(m - 1) - b_2y^+(m - 2)$ for m = 1, ..., M

$y^-(m) = a_2x(m + 1) + a_3x(m + 2) - b_1y^-(m + 1) - b_2y^-(m + 2)$ for m = M, ..., 1

$y(m) = y^+(m) + y^-(m)$ for m = 1, ..., M

where the constants are defined as:

$$a_0 = k \quad a_1 = k(\alpha - 1)e^{-\alpha} \quad a_2 = a_1 - kb_1 \quad a_3 = -kb_2$$
$$b_1 = -2e^{-\alpha} \quad b_2 = e^{-2\alpha}$$
$$\text{with} \quad k = \frac{\left(1 - e^{-\alpha}\right)^2}{\left(1 + 2\alpha e^{-\alpha} - e^{-2\alpha}\right)}.$$

The final edge detection method is implemented as a combination of both of the previous filters. First, the image I(m,n) is filtered in the horizontal direction with g(x). The output of this step is then filtered in the vertical direction with h(x) producing an intermediate image $I_x(m,n)$. A second intermediate image, $I_y(m,n)$, is obtained in a similar way except that g(x) is applied vertically and h(x) is applied horizontally. The amplitude and direction of the edges are then obtained as follows:

$$A(m,n) = \left(I_x(m,n)^2 + I_y(m,n)^2\right)^{\frac{1}{2}}$$
$$D(m,n) = \arctan\frac{I_y(m,n)}{I_x(m,n)}.$$

Since this edge detection method has noise immunity incorporated into it, there is no need to smooth the image first. In fact, as shown in Fig. 3.27, it is apparent that there is more detail in the original test image as opposed to the smoothed test image. The adjustment of α is illustrated in Fig. 3.28. The increased noise immunity with decreasing α actually reduces the detail. This is inappropriate because the lighter features are not detected as well. Setting α to 1.0 seems to provide the best tradeoffs.

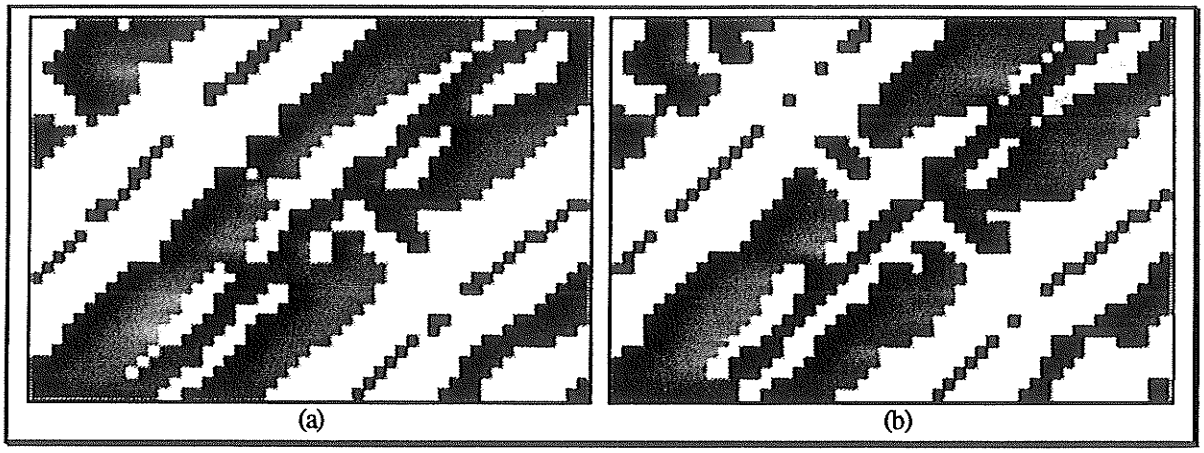


Fig. 3.27 Both images edge detected with α=1.0 and thresholded at 4: a) Pre-smoothed test image 1 b) Unsmoothed test image 1.

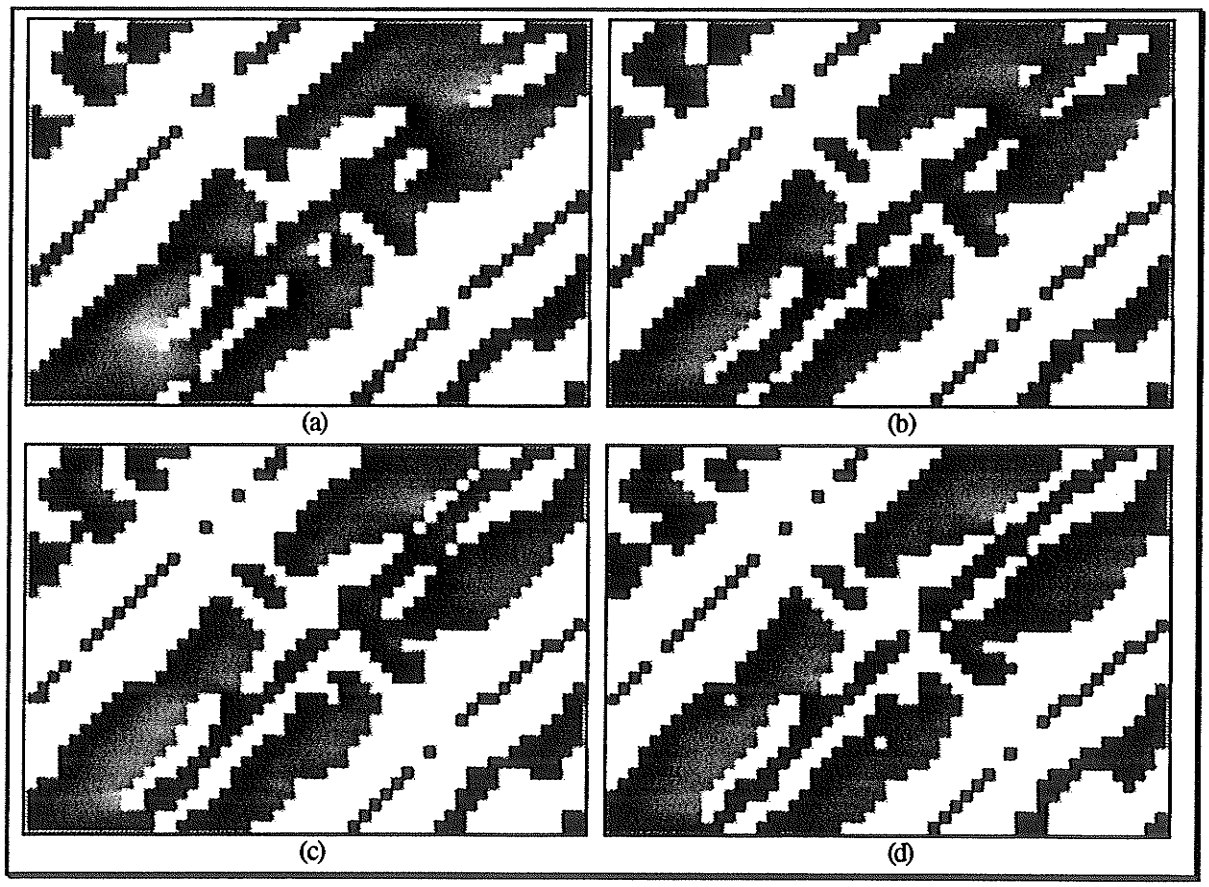


Fig. 3.28 Unsmoothed test image 1 thresholded at 4: a) α=0.8 b) α=0.9 c) α=1.0 d) α=1.1

This edge detection method performs very well in most situations. As can be seen in Fig. 3.28, both sides of the rings are detected. However, the edges are very thick, which

can cause problems during edge tracking because they lie much closer to neighbouring edges. This is most likely due to the reduced locality in order to achieve the proper noise immunity. Another factor is that the edge detector is optimally suited for step edges rather than ramp edges. As stated previously, ring edges tend to resemble ramp edges. Therefore, the edge detector is not optimally suited for rings.

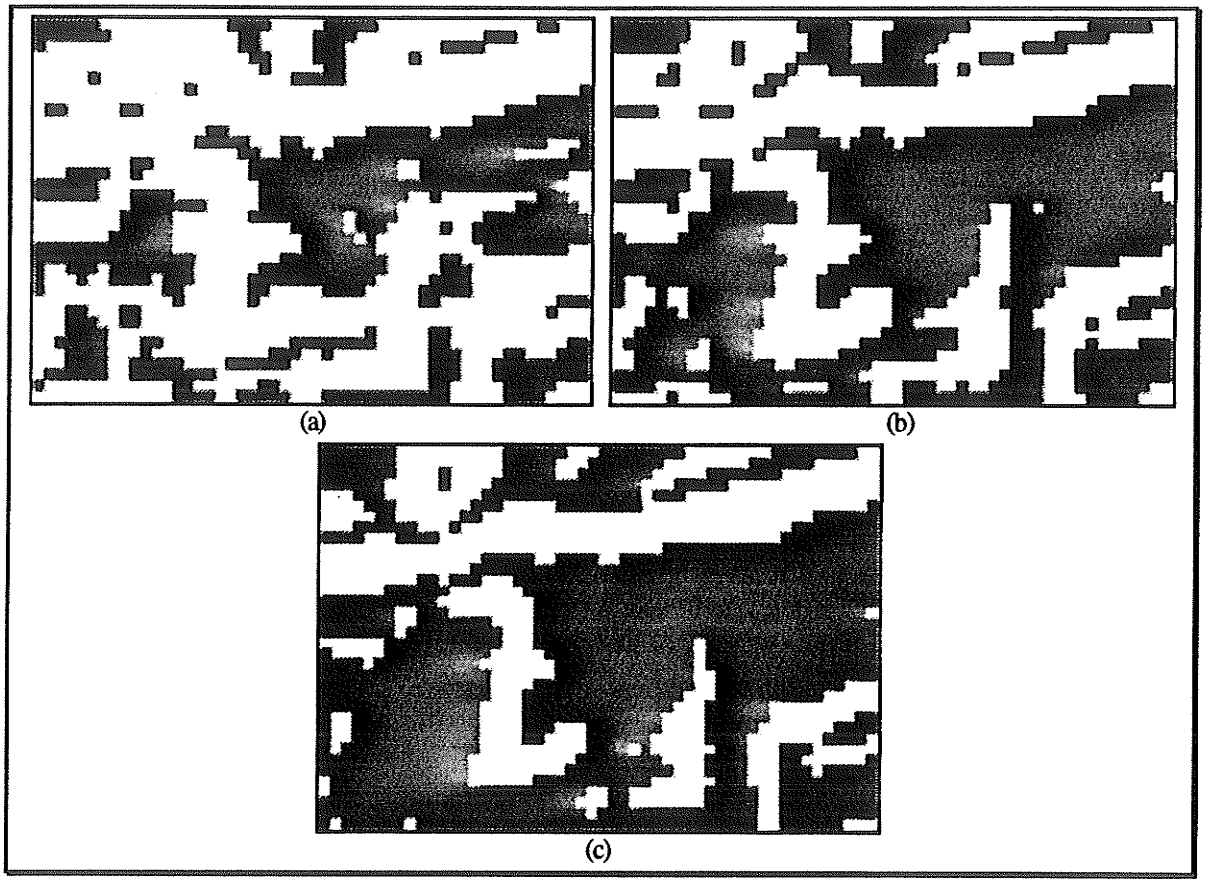


Fig. 3.29 Unsmoothed test image 3 with α=1.0: a) thresholded at 3 b) thresholded at 4 c) thresholded at 5.

Although this method works very well, there are cases where it does not provide enough edge information needed for edge tracking. An example of this is shown in Fig. 3.29. The upper ring is detected well and is visible in all the images. At a threshold of 5, the middle ring is not visible at all, and the lower ring is broken. At a threshold of 3, the middle ring is more visible, as is the lower ring, but other features are enhanced too much

and it is difficult to discern any ring structure. This ambiguity will hinder the edge tracking procedure.

3.2.7.3 Laplacian of the Gaussian

A different approach to edge detection centres upon human vision and the way the eye perceives intensity changes. Using biological data obtained from studies done on early human vision, Marr and Hildreth developed an operator that detects intensity changes [MARR, HILD]. This operator is the Laplacian of the Gaussian.

Intensity changes can occur at any resolution. The problem with most of the previously mentioned edge detectors, as well as this one, is that they detect features at a very limited scale of resolution. Small operators detect small features, but miss large ones, and vice versa. One operator cannot be optimal at all scales. To detect features at various scales, different sized operators should be used. This was the idea behind the multi-template edge detector.

The approach by Marr and Hildreth is to take local averages at various resolutions and then detect the intensity changes that occur at each one. To determine the optimal averaging, or smoothing, filter, they made two considerations. The first consideration is that of scale. The smoothing filter should reduce the range of scales over which intensity changes take place. Thus the filter should be smooth and roughly band-limited in the frequency domain. Therefore, its variance, $\Delta\omega$, should be small.

The second consideration is that of spatial localization. Intensity changes are generally localized, therefore each point in the filtered image should result from a smooth average of nearby points. The filter's spatial variance, Δx, should also be small.

These two requirements are conflicting, as stated by the uncertainty relation, $\Delta x \Delta\omega \leq 1/4\pi$ [BRAC]. The only distribution that optimizes this relation is the Gaussian [LEIP]:

$$G(x) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right) e^{\left(\frac{-x^2}{2\sigma^2}\right)}.$$

In two dimensions it is represented by

$$G(x,y)=\left(\frac{1}{2}\pi\sigma^2\right)e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}.$$

Edges can be detected as peaks in the first derivative of an image, or a zero-crossing in the second derivative of the image. The second derivative of the one dimensional Gaussian is

$$G'(x)=\left(\frac{-1}{\sigma^3\sqrt{2\pi}}\right)\left(\frac{1-x^2}{\sigma^2}\right)e^{\left(\frac{-x^2}{2\sigma^2}\right)}.$$

This operator is a directional derivative. A directional operator must be applied in various directions in order to find the maximum slope through the zero-crossing. Since this is costly, an orientation-independent operator is used instead, which only requires one application. The only orientation-independent second-order differential operator is the Laplacian. The Laplacian of the Gaussian (LoG) operator is defined as:

$$\nabla^2 G(x,y)=\frac{1}{2\pi\sigma^4}\left[2-\frac{x^2+y^2}{\sigma^2}\right]e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}.$$

An image convolved with this operator will contain zero-crossings at detected edge points (because it is a second order differential). Fig. 3.30 illustrates the one-dimensional Gaussian operator and the one-dimensional LoG operator, as well as a two-dimensional representation of the LoG operator [HUER]. The positive central region of the LoG operator is of width *w*. This width is related to the standard deviation σ of the Gaussian by the equation $w=2\sqrt{2}\sigma$ [HILD]. It was suggested in [HILD] that the size of the convolution matrix should be $4w$. However, Huertas and Medioni [HUER] found that $3w$ was sufficient.

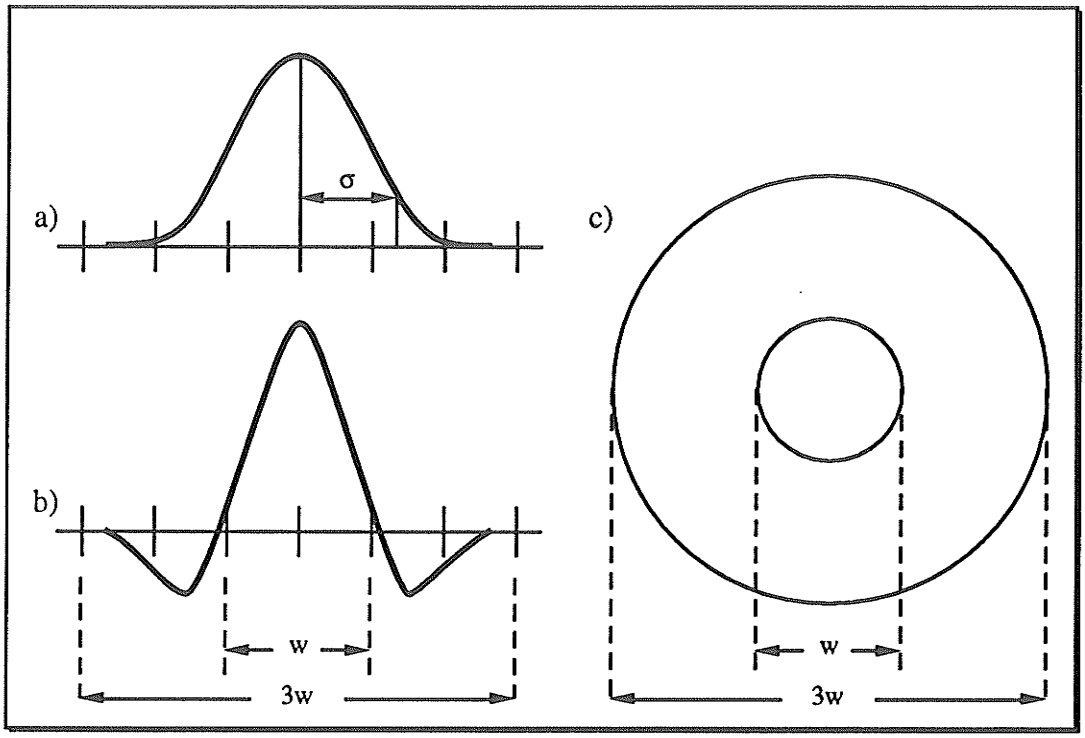


Fig. 3.30 a) 1D Gaussian operator b) 1D LoG c) 2D cross section of the LoG.

The LoG operator will produce a large positive output in response to a positive roof edge. In the sample biscuit image, the rings are negative, or inverted, roof edges. In order to produce a large positive output in response to inverted roof edges, an inverted LoG operator must be used:

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2}{\sigma^2} - 2\right] e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}.$$

This will not alter the positions of the zero-crossings. The constant scaling factor of $1/(2\pi\sigma^4)$ is removed so that the output is not too small. A convolution matrix is generated using the inverted LoG operator. The central matrix element is at (x,y)=(0,0) with increasing x to the right and increasing y down, although the sign of either x or y does not matter because the matrix is symmetric about the centre. This symmetry results because the operator is orientation independent. An example matrix with w=3 and a width of $3w$ is as follows:

$$\begin{bmatrix}
0.0000 & 0.0003 & 0.0022 & 0.0069 & 0.0100 & 0.0069 & 0.0022 & 0.0003 & 0.0000 \\
0.0003 & 0.0047 & 0.0296 & 0.0809 & 0.1099 & 0.0809 & 0.0296 & 0.0047 & 0.0003 \\
0.0022 & 0.0296 & 0.01460 & 0.2649 & 0.2629 & 0.2649 & 0.01460 & 0.0296 & 0.0022 \\
0.0069 & 0.0809 & 0.2649 & -0.0914 & -0.7124 & -0.0914 & 0.2649 & 0.0809 & 0.0069 \\
0.0100 & 0.1099 & 0.2629 & -0.7124 & -2.0000 & -0.7124 & 0.2629 & 0.1099 & 0.0100 \\
0.0069 & 0.0809 & 0.2649 & -0.0914 & -0.7124 & -0.0914 & 0.2649 & 0.0809 & 0.0069 \\
0.0022 & 0.0296 & 0.01460 & 0.2649 & 0.2629 & 0.2649 & 0.01460 & 0.0296 & 0.0022 \\
0.0003 & 0.0047 & 0.0296 & 0.0809 & 0.1099 & 0.0809 & 0.0296 & 0.0047 & 0.0003 \\
0.0000 & 0.0003 & 0.0022 & 0.0069 & 0.0100 & 0.0069 & 0.0022 & 0.0003 & 0.0000
\end{bmatrix}$$

The width of the positive central region is of importance when deciding what size of intensity change is to be detected. The output response of the LoG operator is based upon its width in relation to the edge width. This response will differ for various types of edges. The responses to spike, roof and ramp edges were examined by Huertas and Medioni [HUER].

For a spike edge of width d, and an LoG operator of width s and a positive central region width w the resulting zero-crossings will accurately reflect the location of the edges if $w<d<s$. If $d<w$ then the zero-crossings will be displaced by $(w-d)/2$. For a ramp edge of width d, a single zero-crossing will result if $d<s$. If $d>s$ then no zero-crossing results and the edge is not detected. Roof edges are similar to spike edges. If the ramp width $d<w$ then the zero-crossings are displaced by $(w-d)/2$, otherwise the zero-crossings accurately reflect the position of the edges.

The LoG operator provided excellent results when applied to the sample biscuit image. There was no need to smooth the image beforehand because the Gaussian part of the LoG operator provides the noise immunity. The choice of w is dictated by the average ring width. As stated before, the rings resemble roof edges. The operator width s cannot be smaller than the ramps of the roof edges otherwise some ring edges will not be detected. Also, the width of the positive central region should not be much larger than the ramp width of a roof edge, otherwise the zero-crossings will be displaced.

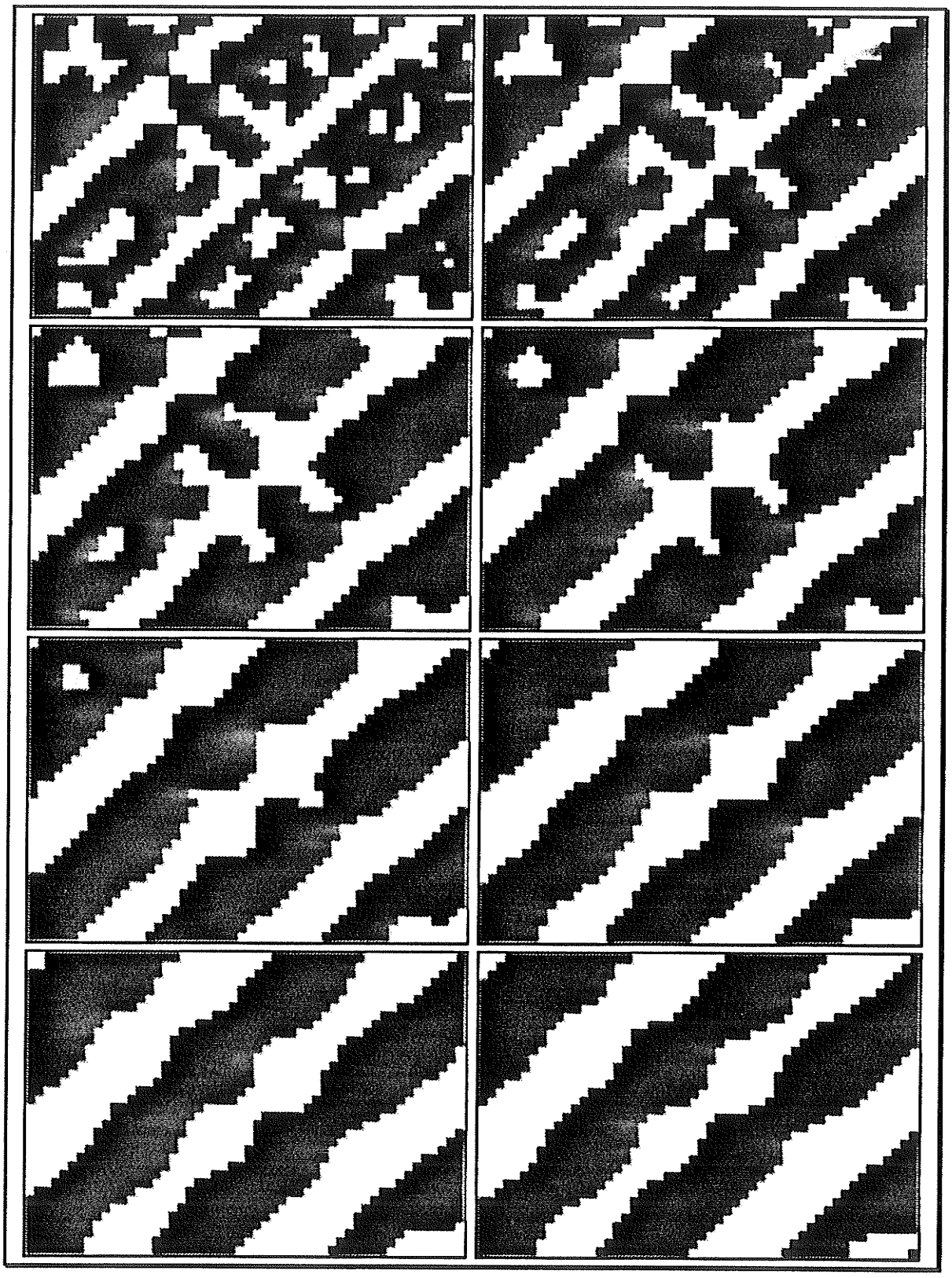


Fig. 3.31 LoG results on test image 1 for values of w=4 to 11 going left to right, top to bottom.

The varied sizes of the rings make the decision of w a trial and error approach. Fig. 3.31 illustrates the output of the LoG operator on test image 1 for various settings of w. The images were thresholded so that any positive output is displayed as white, and any negative output is displayed as black. They represent how the LoG operator detects intensity changes, not edges. The actual edge detection is achieved by locating the zero-crossings using a predicate matching algorithm described below.

As w increases there is less noise, but the rings are thicker because the detected edges are displaced. At about w=10, the middle ring begins to get thinner. This is because the operator is large enough to overlap more than one ring at a time. As a result, the outer rings effect the edge detection of the middle ring. An interesting advantage apparent with test image 1, is that the mark crossing the middle ring is not detected because it is a positive roof edge, and therefore produces a negative output. This allows the elimination of high intensity artifacts that would normally be detected as edges by other methods.

When applied to test image 2, the edge detection was not satisfactory until w=9 (Fig. 3.32). At smaller values, a large amount of noise was picked up, and it appears that there are extra rings. With w>9, the noise is not detected and the breaks in the rings are filled in. This illustrates another advantage of the LoG operator. Larger values of w will cause larger gaps to be closed. This can be attributed to the displacement of the zero-crossings as the rings get thicker. It is not of great concern that the precise location of the ring edges need to be found. Therefore, large values of w work very well, because they reduce a large amount of the noise and provide very good ring-edge detection.

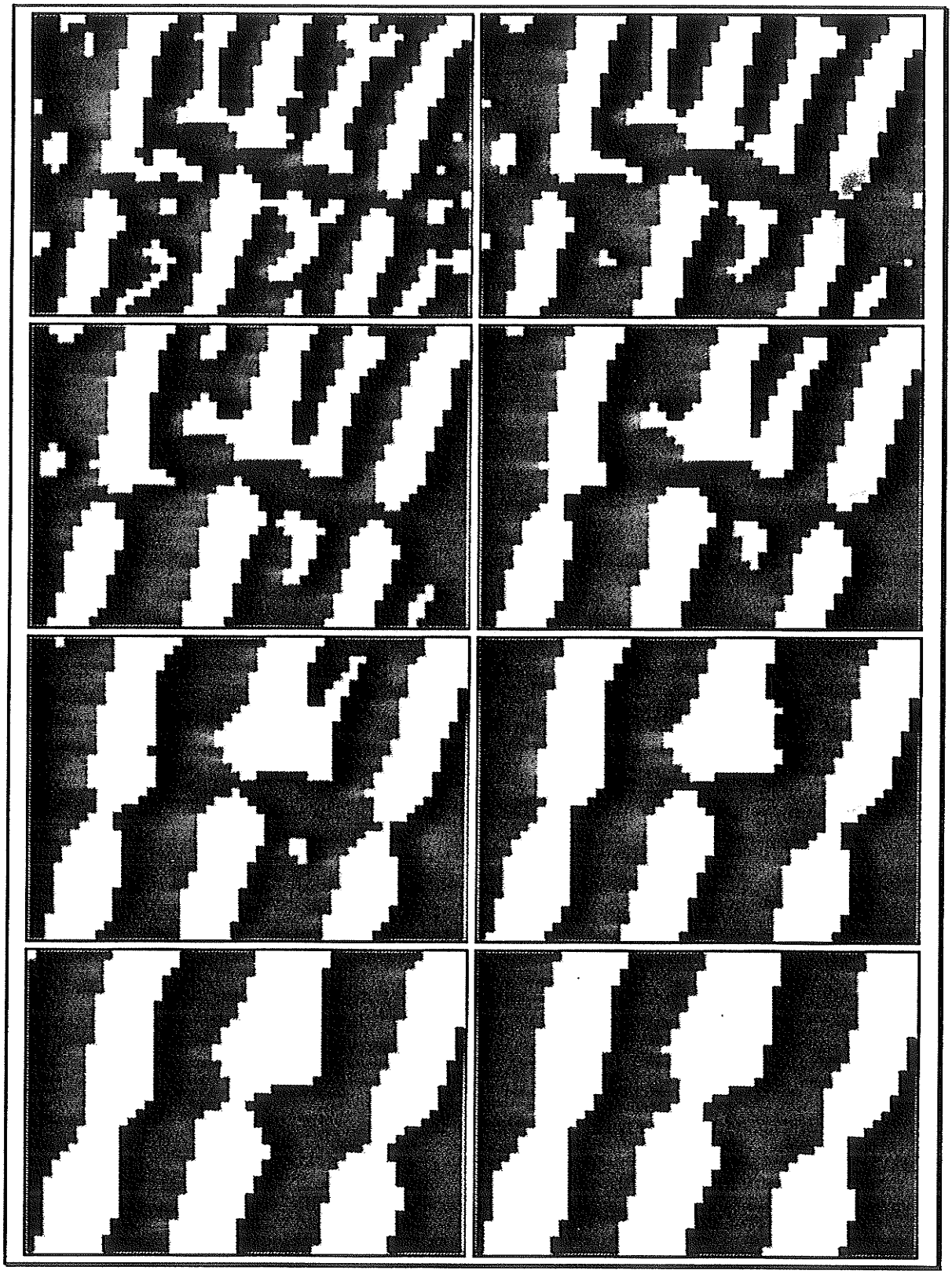


Fig. 3.32 LoG results on test image 2 for values of w=4 to 11 going left to right, top to bottom.

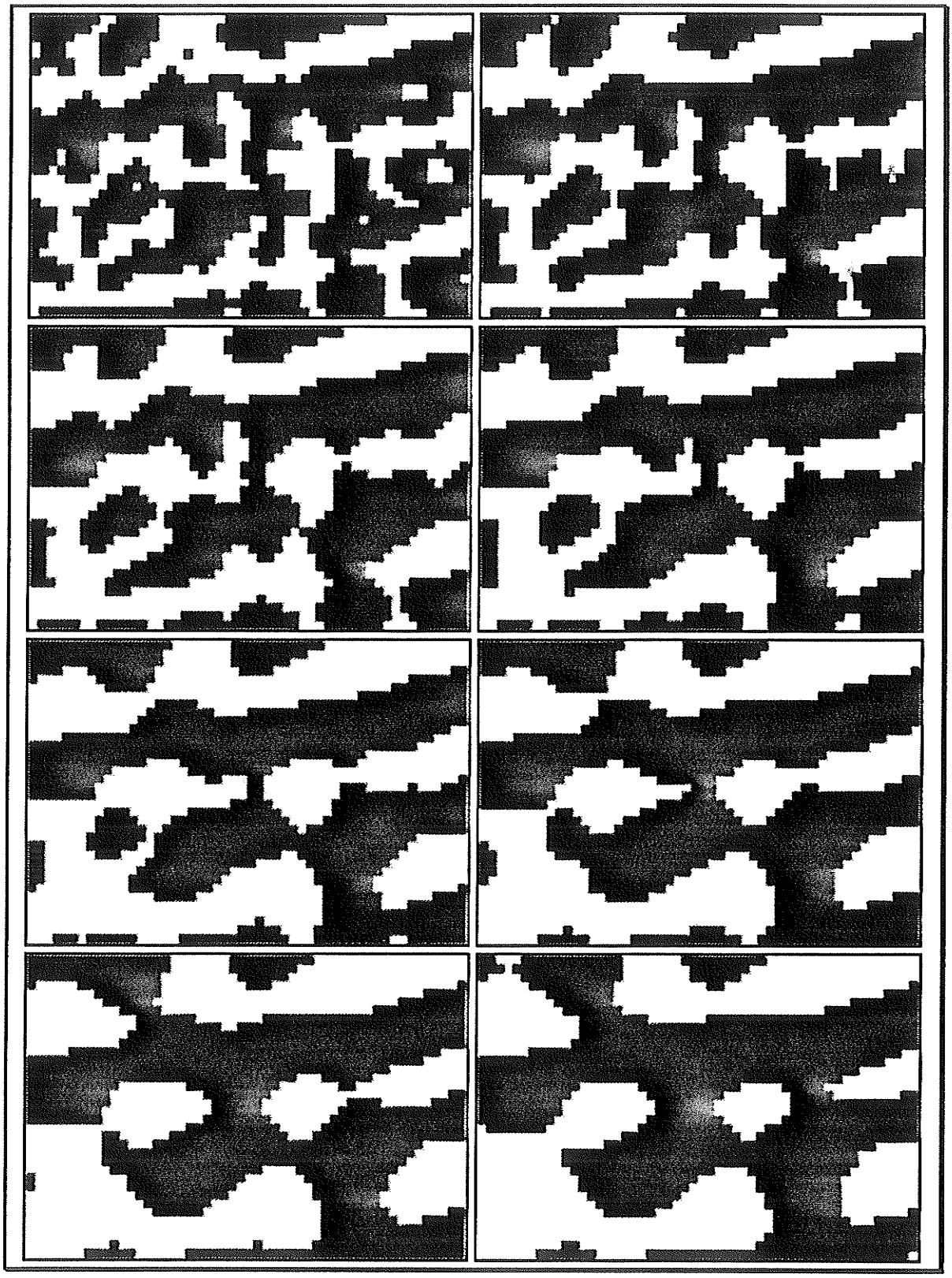


Fig. 3.33 LoG results on test image 3 for values of w=4 to 11 going left to right, top to bottom.

The results upon test image 3 demonstrate how well the LoG operator performs. At all values of w the rings are apparent (Fig. 3.33). Although noise is present at w=4, it is not dominant, and the middle ring can be seen. Once again, at approximately w=8 and w=9, the rings are well defined and there is no noise. However, in this case, the poor definition of the rings in the original image results in large breaks in the middle and lower rings, as well as the merging of the lower two rings along the left side of the image. This problem is enhanced even more above w=9. It is the merging of rings that causes the greatest problem when edge tracking is being performed, because it is very easy for the algorithm to divert from one ring to the next. However, the benefits of the LoG operator's superior edge detection of the rings compared to others, makes it a preferred method in spite of this.

As previously stated, the examples given represent the location of intensity changes, not edges. The edges are located by finding the zero-crossings. A zero-crossing will occur between any pair of adjacent positive and negative values, at a subpixelic resolution. Zero values represent true zero-crossings occurring precisely at the centres of pixels.

A method to find the zero-crossings was presented in [HUER]. Each 3x3 area of the convolved image is matched with eleven zero-crossing predicates. An order is imposed on the predicates so that once a match is found, no other predicates are checked. If a match is not found, then no pixel is marked as an edge point, and the next 3x3 area is checked. For each successful match, subcases are tested to determine which pixel is an edge point. The direction of the edge at that point is also obtained depending upon which subtest was successful. Since there is no need to reproduce every predicate, only the first one is shown for an example (Fig. 3.34). The example predicate as well as the second, third and fourth predicates check for zero-crossings based upon the presence of a zero value. The others rely solely upon the arrangement of adjacent positive and negative values. Fig. 3.35 illustrates the results of applying the predicate matching algorithm to test image 1 convolved with an LoG operator that has a positive central region width of 9 and a total width of 27. The edges have been reduced to single pixel wide lines. These zero-crossing lines are

almost continuous with only a few missing pixels. The discontinuity proves to be a problem for edge tracking if the edge points are to be linked together.

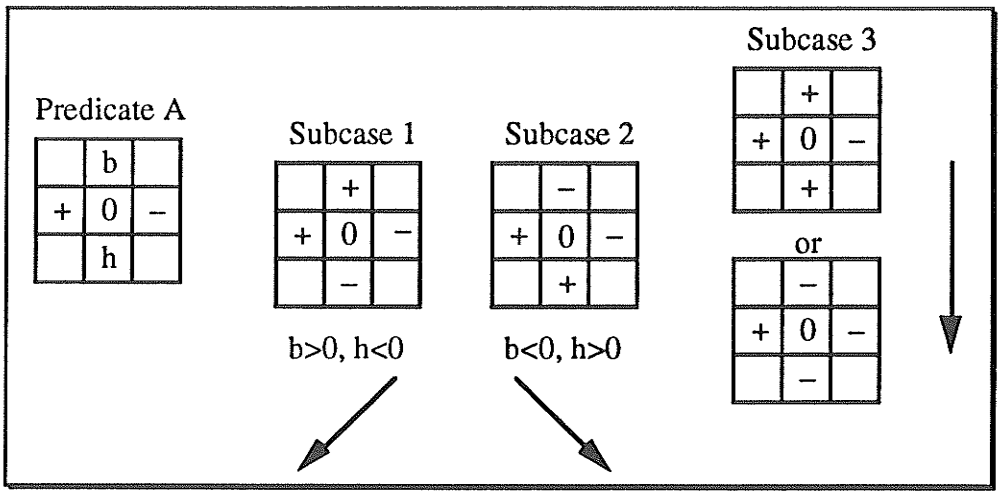


Fig. 3.34 Assume that the 3x3 area matches predicate A. First subcase 1 is checked, and if successful, the edge direction is southwest. If subcase 1 fails, then subcase 2 is checked, which, if successful, gives an edge direction of southeast. If both subcases fail, then the area must match one of the templates in subcase 3, and an edge direction of south is given. In all subcases, the centre pixel is marked as the edge point.

Marr and Hildreth [MARR,HILD] suggested that multiple LoG operators of varied size should be used and the resulting images combined in some way to detect edges. It was noted however, that there is a problem in combining the zero-crossing information from different sized operators, because the positions of the zero-crossings will differ. This is exactly what occurred when the sample biscuit image was edge detected. Therefore this idea was not investigated.

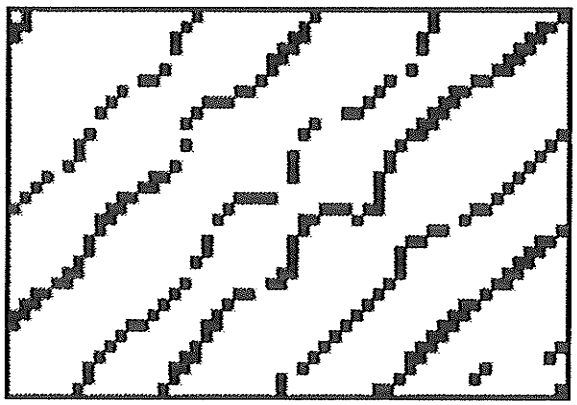


Fig. 3.35 Zero-crossings found with predicate matching algorithm on test image 1 convolved with LoG operator with w=9 and width $3w$.

3.3 Colour Edge Detection

The advantage of colour edge detection over grayscale edge detection is that, depending on the method used, differences in hue can be detected. When a colour image is converted to grayscale, hue information is lost. A grayscale image will only retain intensity information. It may be possible that an edge occurs between two different colours that have identical intensities. Such an edge will be lost when the image is converted to grayscale.

A colour image is composed of three separate images representing the red, green, and blue components. The most common configuration is to have 8 bits per component. Thus, each component can range from 0 to 255 (zero intensity to maximum intensity). The combination of the three components at a particular pixel gives it its colour. Colour edge detection is usually achieved by performing similar operations upon each separate image and then combining the results with some sort of metric.

3.3.1 Colour Gradient

The gradient can be applied to colour images, but not in a trivial manner. A first approach would be to take the gradient of each colour component separately and then combine them, using the sum, RMS, or maximum of the absolute values. These methods may work well with a grayscale image, but can cause problems in a colour image. For example, consider a point in an image where the blue band is constant and the red band increases in intensity from left to right while the green band decreases. The gradients of the red and green bands will be of opposite sign. If they are equal in magnitude, the values will cancel, giving an edge strength of zero. The RMS will give a proper edge strength in this case, but can fail in others. Consider two points where the gradient magnitudes are equal, for all components. The gradients of the first point have identical directions, while the second point has one gradient in an orthogonal direction to the other two. The RMS would give equal weights for both, but this is incorrect. More weight should be assigned to the first because the gradient directions have the least variance.

A way to combine the gradients of the colour components, without these problems, was given by Di Zenzo [ZENZ]. The three component colour image is extended to the idea of a *multi-image* which can have *m* components, with *m* not necessarily equal to 3. As with a grayscale image, which can be considered a two dimensional function $f(x,y)$, a multi-image can be treated as *m* functions $(f^1(x,y), ..., f^m(x,y))$. A point (x,y) in a multi-image with $m=3$ can be represented as the vector (R(x,y),G(x,y),B(x,y)) corresponding to the red, green and blue components.

By treating a multi-image as a vector field, Di Zenzo obtained a formula for computing the angle of maximum contrast. This angle is then used to compute the maximum contrast (i.e., edge strength). The two equations are as follows:

$$\theta = \frac{1}{2}\arctan\left(\frac{2g_{xy}}{g_{xx} - g_{yy}}\right)$$

$$\text{Edge strength } = F(\theta)^{\frac{1}{2}} = \left(g_{xx}\cos^2\theta + 2g_{xy}\cos\theta\sin\theta + g_{22}\sin^2\theta\right)^{\frac{1}{2}}$$

The gradients along the X and Y directions are written as the following two vectors:

$$\mathbf{u} = \frac{\partial R}{\partial x}\mathbf{r} + \frac{\partial G}{\partial x}\mathbf{g} + \frac{\partial B}{\partial x}\mathbf{b}$$

$$\mathbf{v} = \frac{\partial R}{\partial y}\mathbf{r} + \frac{\partial G}{\partial y}\mathbf{g} + \frac{\partial B}{\partial y}\mathbf{b}$$

where **r**, **g**, and **b** are the unit vectors corresponding to the R, G, and B axes. By combining these two vectors via the dot product, the variables g_{xx}, g_{yy}, and g_{xy} are obtained:

$$g_{xx} = \mathbf{u}\bullet\mathbf{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2$$

$$g_{yy} = \mathbf{v}\bullet\mathbf{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2$$

$$g_{xy} = \mathbf{u}\bullet\mathbf{v} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y}.$$

The gradients for the individual components can be computed using any of the methods discussed in section 3.2.1. When computing the maximum rate of change, θ and $\theta+\pi/2$ must be checked. One value will correspond to the minimum rate of change, and the other will correspond to the maximum rate of change. The resulting edge strengths can then be thresholded.

Application to a colour version of test image 1 did not produce acceptable results. As can be seen in Fig. 3.36, there is simply too much noise. The lack of a properly smoothed colour image meant that a proper examination could not be done.

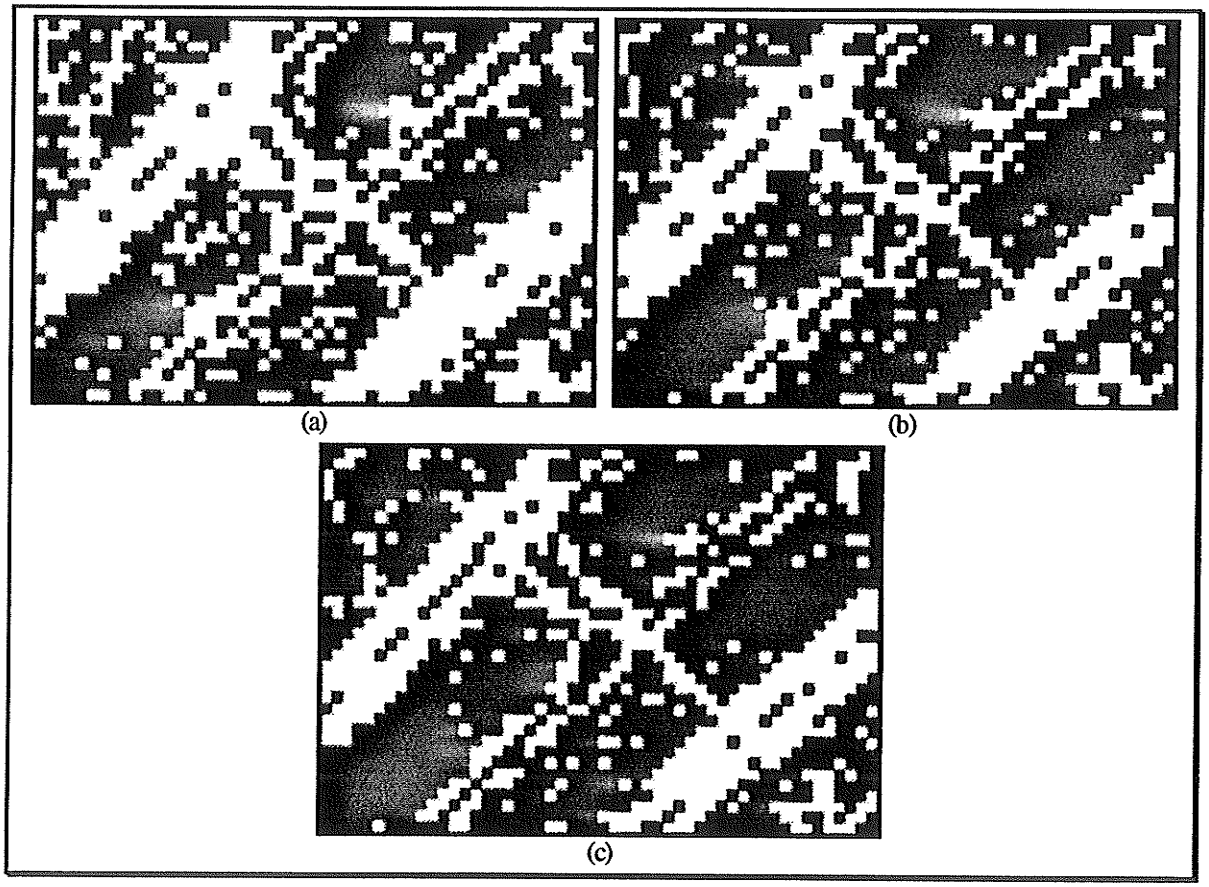


Fig. 3.36 a) Edge detected image; threshold at 8 b) threshold at 9 c) threshold at 10.

A similar approach in computing the colour gradient was taken by Cumani [CUMA]. However, special cases are taken into account, and the resulting algorithm is different. The maximal contrast λ (i.e., edge strength) is computed as follows:

$$\lambda = \frac{g_{xx} + g_{yy} + \sqrt{(g_{xx} - g_{yy})^2 + 4g_{xy}^2}}{2}.$$

The angle of maximum contrast is computed in the same way, resulting in the vector

$\mathbf{n} = (\cos\theta, \sin\theta)$

which points in the direction of maximum contrast. If $(g_{xx} - g_{yy}) = 2g_{xy} = 0$ then **n** is undefined. This occurs when the contrast is the same in all directions. The algorithm for detecting edges is as follows:

1) For each pixel compute λ and **n**. Discard pixels where **n** is undefined or λ lies below a threshold.
2) At each remaining pixel, fit a least-squares quadric to the values of λ. The pixel is marked as an edge point if the resulting direction is within a certain threshold of **n**.

Fitting a least-squares quadric is achieved by convolving a point with the two matrices

$$\mathbf{A} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

If the two results from the convolutions are a and b respectively, then the angle is

$$\theta = \arctan\frac{b}{a}.$$

Once again, the lack of a properly smoothed image degraded the results. The choice of an angle threshold in the second part of the algorithm had a noticeable effect on the edge detection. The difference between thresholding with $\pi/4$ and $\pi/8$ can clearly be seen in Fig. 3.37. Allowing a larger angular difference between **n** and the angle computed with the least-squares method, gave better edge detection. However, the large amount of noise is unsatisfactory.

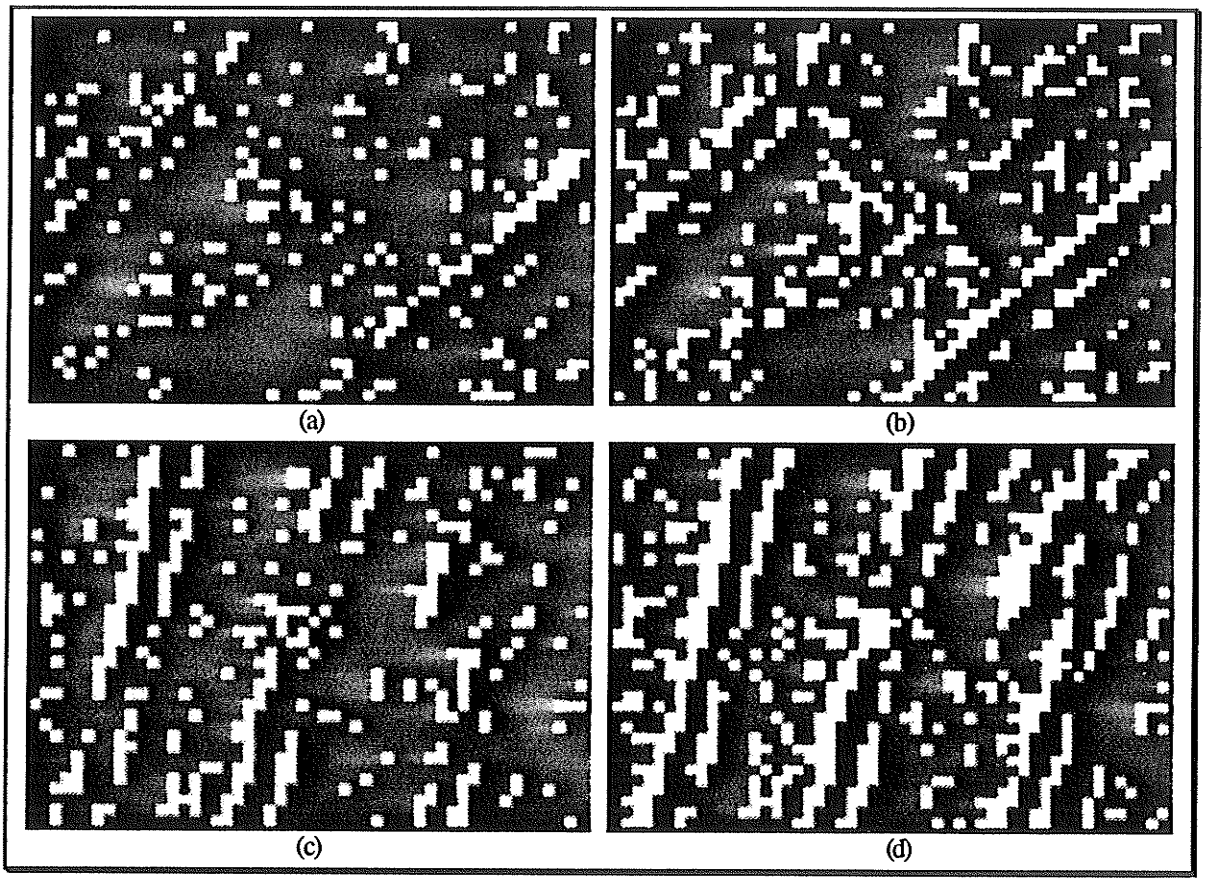


Fig. 3.37 The output from step 1 of the algorithm was not thresholded, in order to obtain as much information as possible. a) & b) Colour test image 1 with angle threshold of $\pi/8$ and $\pi/4$, respectively. c) & d) Colour test image 2 with angle threshold of $\pi/8$ and $\pi/4$, respectively.

3.3.2 Colour Entropy

A simple extension can be applied to the entropy operator so that it can be used for colour edge detection [SHIO]. By applying the operator to the separate red, green, and blue components of an image, and then combining them into a single entropy operator, changes in brightness and hue can be detected.

Let H_R, H_G, and H_B be the entropies of the red, green and blue components respectively. Let a_R, a_G, and a_B be the actual red, green and blue components of the image at the centre pixel. The colour entropy is then defined as:

$$H = q_R H_R + q_G H_G + q_B H_B$$

where

$q_R = a_R / (a_R + a_G + a_B)$

$q_G = a_G / (a_R + a_G + a_B)$

$q_B = a_B / (a_R + a_G + a_B)$.

The colour entropy operator has the same properties as the grayscale entropy operator. It is small when the change in brightness is severe and small when the change in hue is severe. Once again, a reverse thresholding is used, since lower values indicate the presence of an edge.

Even without the benefit of a pre-smoothed test image, the colour entropy edge detection method did not detect the noise as much as the gradient methods. There are, however, problems that are identical to the grayscale entropy operator. Certain thresholds needed to detect faint rings are too high for darker rings, resulting in thick edges that merge together (Fig. 3.38). The edges also tend to be rather rough, and noise is still a problem at higher thresholds.

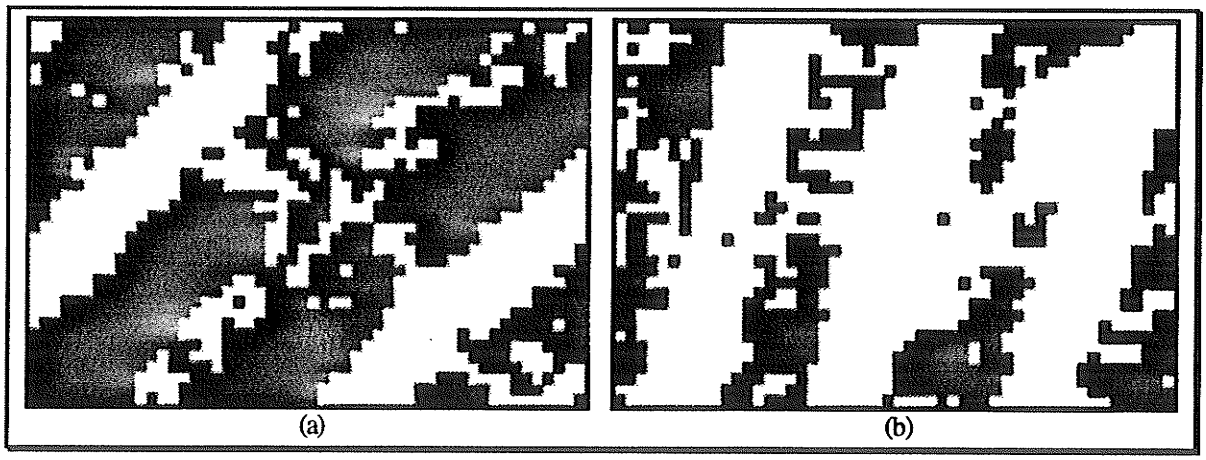


Fig. 3.38 Both images were pre-smoothed. a) Test image 1 thresholded at .999850 b) Test image 2 thresholded at .999850.

3.4 Conclusions

The gradient edge detection methods, while simple to implement, are too sensitive to noise. They are however efficient and offer excellent edge localization. The Prewitt and

Sobel operators offer better noise immunity, while maintaining excellent localization. The weighted averaging used by the Sobel operator provided better edge detection than the Prewitt operator at the cost of a slightly higher noise response. While the gradient methods had much better performance on the smoothed test images, the Prewitt and Sobel operators did not perform as well.

The two iterative edge detection methods suffered many problems. The method proposed in [KASV] showed poor edge detection as well as poor edge thinning. It was very sensitive to noise, and produced a very small output that required a threshold of 1 to enhance. The method proposed in [EBER] performed much better. It was not as sensitive to noise, because of the edge smoothing phase, and the thinning phase worked well to thin the edges. Neither method is acceptable though, because the noise sensitivity is too great and the edge detection capabilities are unsatisfactory.

The multi-template edge detection method performed well. It provides adequate noise immunity and localization. Its drawback lies in choosing a detection threshold that works well for most cases. It performed well upon both test image 1 and test image 2, but produced unsatisfactory edge detection on test image 3.

The median based zero-crossing edge detection method provides excellent localization with good noise immunity. Its performance was satisfactory upon both test image 1 and 2. It did not perform well upon test image 3, however, and in each case there was an excess of noise in the form of zero responses, which could only be removed at the cost of reduced edge detection.

The entropy edge detection performed reasonably well but has drawbacks. A given threshold does not work well for all cases, and is difficult to choose because the output of the operator is between 0.0 and 1.0. It is very sensitive to noise, so the image must be smoothed beforehand. Also, there is the absence of edge direction information, which must be obtained in another way.

The optimal recursive edge detection method has excellent noise immunity, and therefore produced excellent results upon the unsmoothed test images. The ability to adjust a single variable to obtain a desired tradeoff of noise immunity and edge localization is an advantage. However, the need for better noise immunity in the test images resulted in poor localization and therefore thicker edges. Its performance on test image 3 was also unsatisfactory.

The colour edge detection methods were investigated to find out whether or not more information could be obtained from the colour image as opposed to the grayscale image. However, the lack of a properly smoothed test image resulted in poor performances from the colour gradient methods. The colour entropy operator exhibited a reasonable amount of noise immunity, but suffers from the same problems as the grayscale entropy operator.

The Laplacian of the Gaussian operator provided the best results overall. It has excellent noise immunity and the ability to adjust the edge localization by changing the size of the operator. In the test images, a large operator with a positive central region width of at least 9 was needed to reduce the noise response, and this resulted in poor localization, but, the excellent edge detection far outweighs the localization problem. Even though test image 3 represented a worse case scenario, the LoG operator still managed to detect the rings with only a few minor flaws.

Chapter 4
Edge Tracking

4.1 Introduction

To the human eye, an edge detected image can provide a wealth of information. For a computer to make use of the information within the image, the edges must be transformed into something it can understand. This is usually done by linking together edge elements, or edgels, into chains. The structure of the chains can then be examined in whichever way is desired.

The features of interest within the sample biscuit image are the rings. In many cases, the ring edgels that result will be sparse and disconnected. Simply linking them with their neighbours will produce many short segments. It is more desirable to extract a complete ring, rather than small sections of it. By implementing an edge tracking method, the ring can be traced out in a sequential fashion. This will produce an ordered set of data points which can then be linked together conveniently.

The same coordinate system from Chapter 3 is used for the examples. References to a particular direction are made using standard map directions (i.e., north, south, east, west etc.) North refers to a decreasing Y direction (up the page), and west refers to a decreasing X direction (to the left side of the page). The other directions are intuitively obvious.

4.2 Subpixelic Edge Tracking

The first edge tracking method employed was proposed in [AVRA]. This method makes use of a finite-state automaton to keep track of the location of an edge. In a grayscale image, an edge will occur at the boundary of two regions that have a significant difference in intensity. At any point along the edge, there will be a "light" pixel and a "dark" pixel on either side. The positions of two pixels are maintained at all times, one light and one dark. The automaton advances the pixels based upon certain neighbouring pixels. The goal is to maximize the difference between the light and dark pixels.

Which neighbours are examined depends upon the state of the automaton. There are eight possible states (Fig. 4.1). Each state corresponds to a possible direction automaton. In the example shown, the dark region is always to the right of the direction of motion.

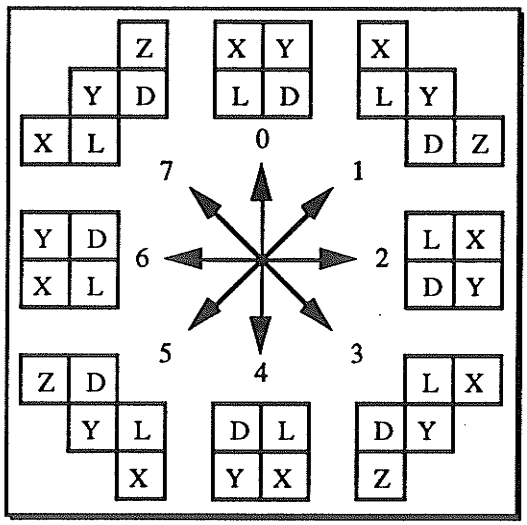


Fig. 4.1 The eight possible states of the automaton. D is the dark pixel, and L is the light pixel. X, Y and Z are the neighbouring pixels involved in the test.

When the automaton is in a vertical or horizontal state (0,2,4,6), three pairs of pixels are tested: (D,X), (L,Y) and (X,Y). For example, assume the automaton is in state 2. There are three possible state changes:

1) (D,X) has maximum intensity difference: L is advanced to X; state becomes 3

2) (L,Y) has maximum intensity difference: D is advanced to Y; state becomes 1

3) (X,Y) has maximum intensity difference: L is advanced to X, and D is advanced to Y; state stays at 2

If the automaton is in one of the diagonal states (1,3,5,7), four pairs of pixels are tested: (D,Y), (L,Y), (X,Y), and (Y,Z). For example, assume the automaton is in state 1. There are four possible state changes:

1) (D,Y) has maximum intensity difference: L is advanced to Y; state becomes 2
2) (L,Y) has maximum intensity difference: D is advanced to Y; state becomes 0
3) (X,Y) has maximum intensity difference: D is advanced to Y, and L is advanced to X; state stays at 1
4) (Y,Z) has maximum intensity difference: L is advanced to Y, and D is advanced to Z; state stays at 1

This method is typically used to perform edge detection of continuous edges. Avrahami and Pratt used it to follow the edges defined by glyphs scanned in at a high resolution. In such a situation, noise is not a major concern, and the algorithm works well.

The algorithm can be used to follow ring edges, since there is an intensity difference between a ring and the surrounding wood. For it to work well, the noise has to be reduced. It can also be used to track edges that have already been detected by another algorithm. In particular, it is well adapted to track the edges detected by the LoG algorithm.

By using only single pixel differences, noise can very easily cause the algorithm to diverge from its intended path. To reduce tracking errors, differences of averages can be used. Instead of taking the gray level at a pixel, an average of the local area, centred on the pixel, can be taken. This will avoid diversions from the edge due to a single pixel that has an abnormally low or high intensity. The size of the averaging area must be small enough so that it won't extend to other rings.

An example of how the subpixelic edge tracking algorithm works is shown in Fig. 4.2. The image is an enlargement of a ring that has been edge detected with the LoG

operator using a positive central region width of 9. The algorithm was set up so that the light pixel is always on the left and the dark pixel is always on the right. Notice how it can be used to track the side of a ring. This can very easily be modified so that the dark pixel is on the left and the light pixel is on the right. In this way the inner side of the ring could be tracked instead.

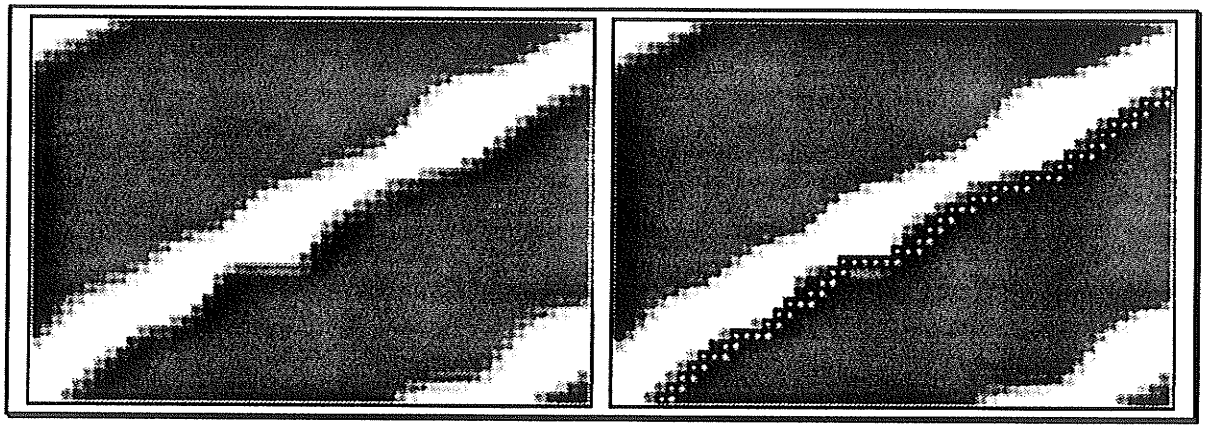


Fig. 4.2 An example of how the subpixelic method tracks an edge. The marked points represent the darker pixel at each step in the algorithm.

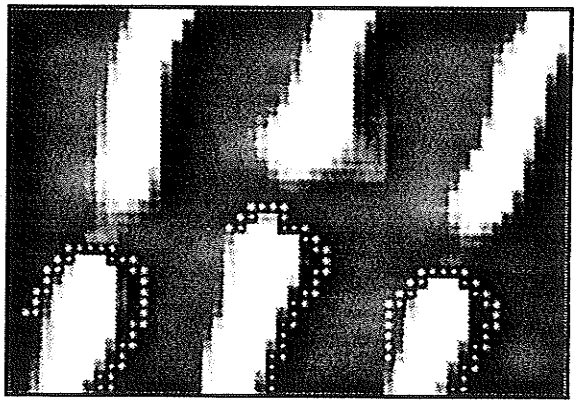


Fig. 4.3 Three cases of where the subpixelic edge tracking method fails to cross a ring break.

Although it will work very well in situations as that shown above, the subpixelic method is very susceptible to changes in intensity. Problems that occur because of this susceptibility can be circumvented by tracking the ring edges detected by the LoG operator. Such rings will not, in most cases, contain noisy areas. However, the occurrence of ring breaks will pose many problems with this method. Without the enhancements that will be

described later, the algorithm will simply turn around at the break, and continue in the opposite direction. An example of this is shown in Fig. 4.3. The image used was test image 2 from Chapter 3. Again, the LoG operator was used to detect the edges.

4.3 Minimum Intensity Edge Tracking

Another tracking method can be designed to exploit the intensity difference of the tree rings from the surrounding wood. In the sample biscuit image, the rings are distinguished by their lower intensity. If the algorithm is made to specifically follow lower intensities, then it can track a ring. However, this sort of idea will only work with narrow spike or roof edges. This is appropriate because the rings are actually roof edges. This tracking method is also well suited for the output produced by the non-inverted LoG operator. Applying this operator to the sample biscuit image without finding the zero-crossings will produce well defined rings that have a very low intensity compared to their surroundings.

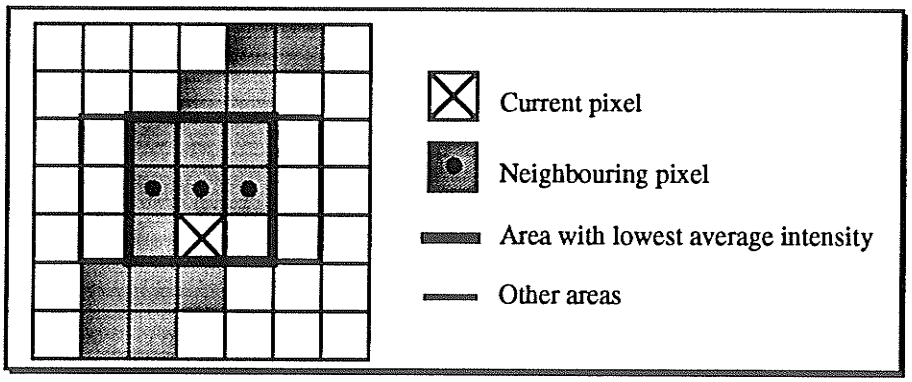


Fig. 4.4 An example of edge tracking using local averages. The current direction of movement is north. If the current pixel is at (x,y) then areas to be examined are centred on pixels (x-1,y-1), (x, y-1), and (x+1,y-1). Since the area centred at pixel (x,y-1) has the lowest local average, the current pixel is moved to position (x,y-1) and the current direction stays the same.

A first attempt at designing a minimum intensity tracking algorithm, was to track single pixels based upon square local averages. As with the algorithm in [AVRA], there are eight possible states, corresponding to eight possible directions of movement. Each state has three possible directions in which to advance: straight ahead, or 45° to the left and right.

The current pixel can therefore be advanced to three possible neighbours. A local average is taken around each of these three neighbours, and the current pixel is advanced in the direction of lowest intensity, with the state changing appropriately. Fig. 4.4 illustrates an example of pixel movement. Directions at 90° to the left and right are not checked because they may force the edge tracker to reverse its path too quickly. A generally forward direction of motion should be maintained.

The fact that the areas overlap makes this edge tracking method a bad choice. Using larger areas to balance increased noise will result in further overlap. Each area should be disjoint from the others so that the averages are not influenced by a common set of pixels. Having common pixels means that the averages will be closer to one another. If there is a proper direction to be taken, it may not be given enough weight to be chosen above the other two directions.

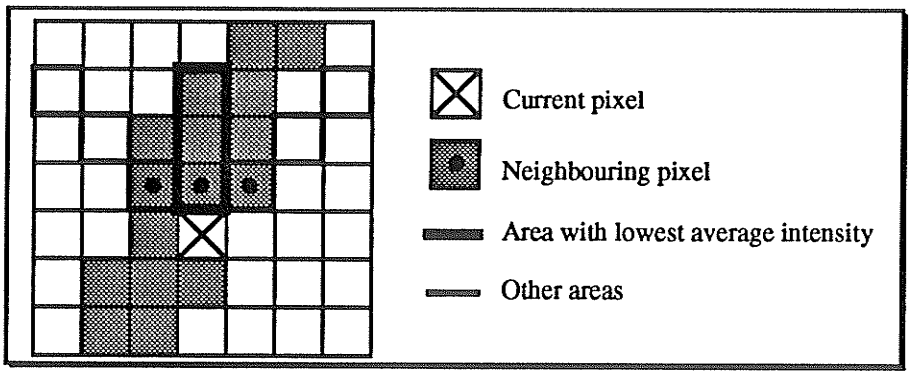


Fig. 4.5 An example of edge tracking using line averages of length 3. The current direction of movement is north. If the current pixel is at (x,y) then the neighbouring pixels to be examined are at (x-1,y-1), (x, y-1), and (x+1,y-1). The vertical area has the lowest average intensity, so the current pixel is moved to position (x,y-1) and the direction of motion stays the same.

Since the structure of a tree ring is known beforehand to be rather circular, this method can be modified to take advantage of this. Averages can be taken along lines of direction rather than around a particular pixel. The tracking is still carried out in the same way, except that the averages are computed used disjoint sets of pixels. Fig. 4.5 illustrates an example of how the averages might be taken. At any given time in the tracking, there will be a high probability that two of the line averages will include many pixels off of the ring,

as long as they are made long enough. Again, as with the other method, the line lengths must not be made so long that they intersect with other rings.

This method has the added advantage that ring gaps shorter than the line length can be crossed (see Fig. 4.6). It does have a drawback, however, because it will track the inside of a ring rather than its edges. This is not a problem if the ring is thin enough, since tracking the centre will eventually result in a good representation of the ring.

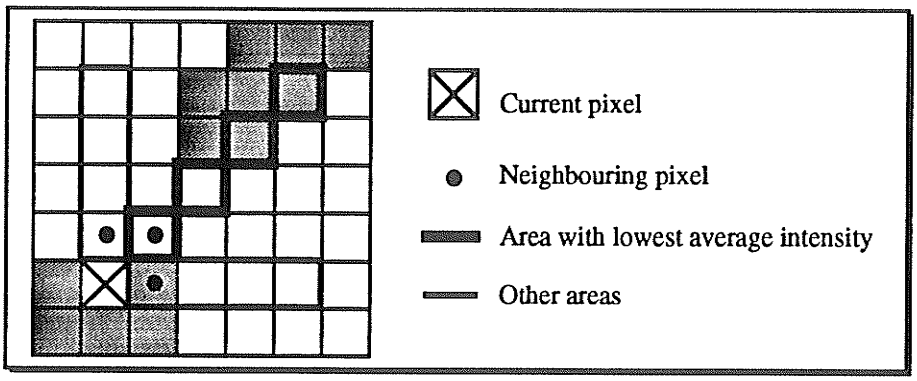


Fig. 4.6 Current direction is Northeast. Notice that if the line crosses the gap and intersects the ring, the edge tracker will not follow an incorrect path.

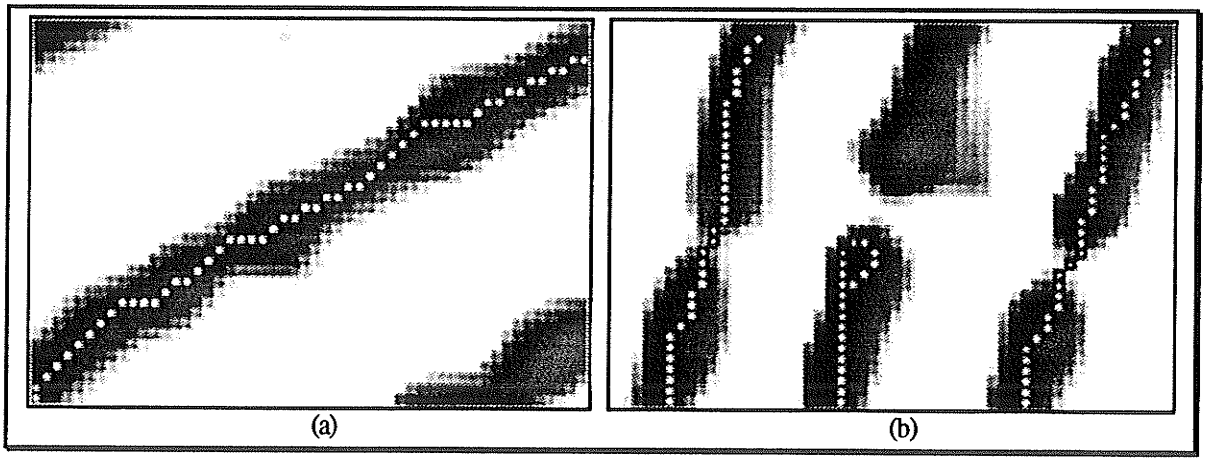


Fig. 4.7 Examples of the minimum intensity edge tracking algorithm.

To illustrate how the algorithm will track a ring, the same images from the previous section are used. They are inverted so that the ring is of a lower intensity, since the algorithm was designed to follow low intensities (Fig. 4.7(a)). Notice how it stays in the

middle of the ring. Fig. 4.7(b) shows how the algorithm can successfully cross some ring breaks. The break in the middle ring was too large, so the algorithm reversed its path.

4.4 Enhancements

In an ideal environment, the two edge tracking methods mentioned above would execute without error. Such an ideal environment would not contain any noise, and the edges would be complete, without breaks in them. However, in practice, noise is very likely to occur and many of the rings may have breaks in them. The natural intensity difference of a ring is not guaranteed to be homogeneous along the entire circumference, and, when edge detection is performed, various natural flaws may result in spurious edges that join adjacent ring edges. Any one of these features can force the edge tracker off of the proper path.

Fortunately, a tree ring has a particular structure that can be exploited. The curvature does not change dramatically within any short distance along its length. This knowledge can be used to modify the edge tracking algorithms so that they perform better.

The rather constant curvature means that any abrupt changes in direction during the tracking procedure should not occur. For example, if the current tracking direction is north, one should not expect the direction to change to south within a certain pixel distance. If this were to occur, then the tracking algorithm would have followed a very high curvature path.

The algorithm is modified to compare the current direction to the directions that were taken at each of the n previous pixels. If an opposite direction is encountered then the algorithm must backtrack to that pixel and move in a different direction than originally taken. An example of how this is done is shown in Fig. 4.8. Assuming the algorithm is at position 1 in the figure, it would continue as follows:

1) The current direction is north at pixel (x,y), and there are three possible directions to move, corresponding to pixels (x-1,y-1), (x,y-1) and (x+1,y-1) which are

shown in bold (Fig. 4.8(a)). The algorithm has chosen to move northeast, so the current pixel is now at (x+1,y-1) and the current direction is northeast.

2) The algorithm proceeds along an incorrect path to position 2. Assume that the algorithm checks the 10 previously tracked pixels. At position 1 (7 pixels back), the direction was north, which is opposite to the current direction of south. This indicates that the path just traversed has a high curvature. Thus, the algorithm backtracks to position 1.

3) At position 1, the algorithm moves once in the current direction of north, to position 3 at (x,y-1), and keeps the current direction the same (Fig. 4.8(b)). The neighbouring pixels to be checked are now different and the error can be avoided.

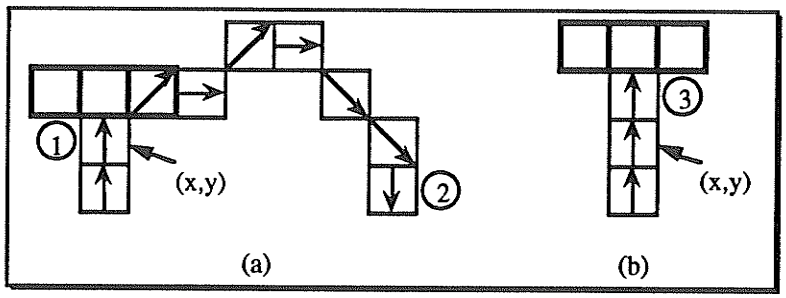


Fig. 4.8 a) The algorithm has following an incorrect path. b) The algorithm has backtracked and is now following a new path.

To detect the path change earlier, instead of just checking for an opposite pixel direction than the current one, the algorithm can check for directions that differ by 90° or more. So, in the above example, different directions from that at position 1 would be south, southeast and southwest. Each one differs from north by 90°. If these extra directions were checked then the incorrect path would be detected two pixels earlier than position 2, where the pixel direction is southeast.

An example of where this backtracking method improves the edge tracking algorithm is shown in Fig. 4.9. The image is a clasp of a ring that was edge detected with the LoG operator. The figure illustrates how the minimum intensity edge tracking algorithm can successfully traverse a ring break without reversing its path. It uses a centre averaging line

that is 5 pixels long with the other two at 3 pixels in length. Having a longer centre averaging line increases the chance of crossing short ring gaps. Fig. 4.9(a) is the path taken by the algorithm without backtracking. Fig. 4.9(b-l) shows the path the algorithm takes using backtracking. At each point, the 7 previous pixel directions are checked.

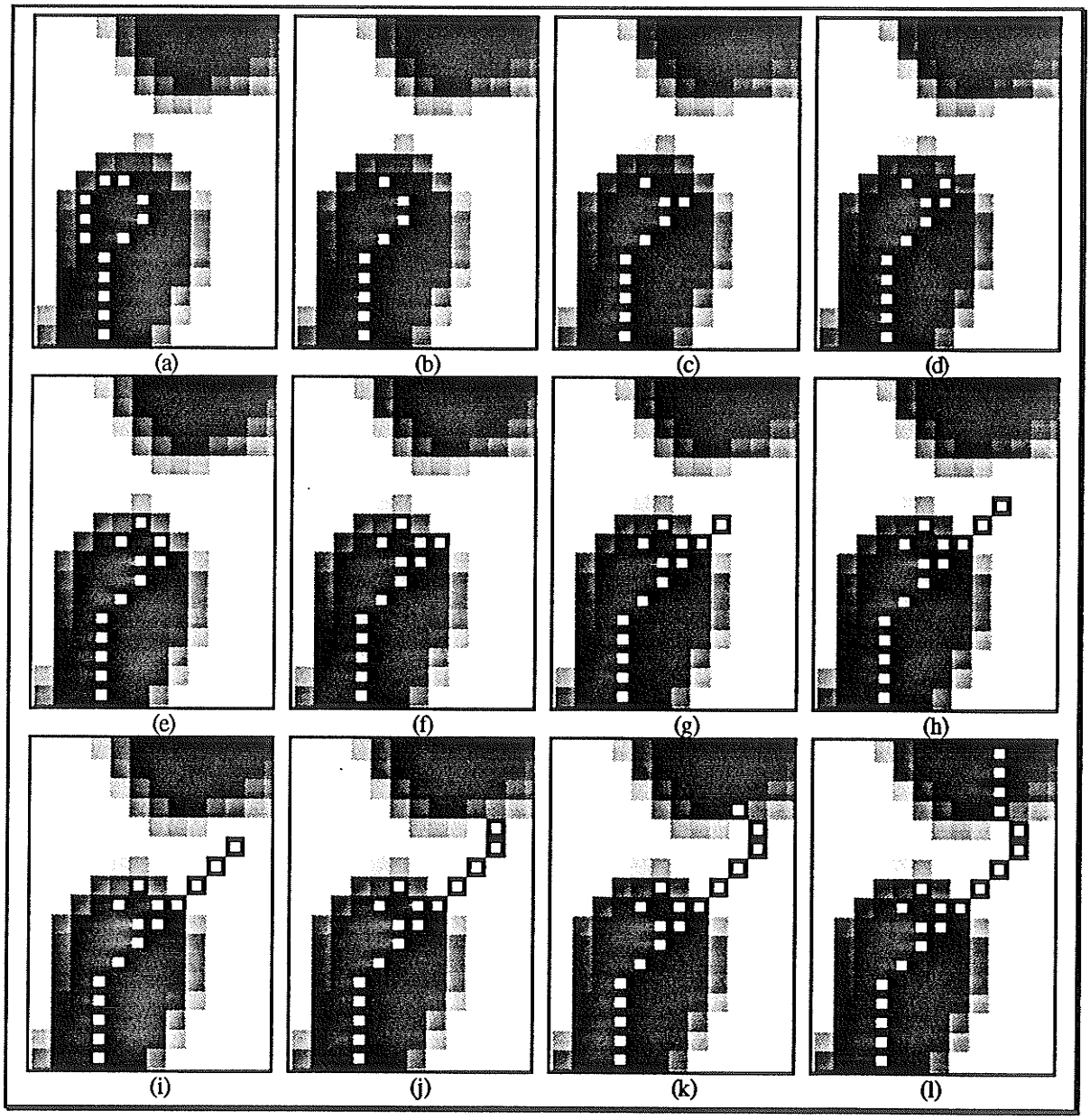


Fig. 4.9 An example of the minimum intensity edge tracking algorithm correctly traversing a break in a ring.

In Fig. 4.9(b), the algorithm has reached a point where the next pixel has a different direction (west) from the pixel 2 steps back (northeast) (the pixel is not shown because it is detected before the algorithm actually moves to it). In Fig. 4.9(c), the algorithm has backtracked 2 pixels and stepped once in a northeast direction. At this point, it starts to turn left (Fig. 4.9(d,e)). Once again it detects a different direction 2 pixels back, backtracks to that point, and steps once in a northeast direction (Fig. 4.9(f)). From there it continues forward, eventually connecting with the ring again (Fig. 4.9(g-k)). Fig. 4.9(l) shows the algorithm a few more steps along from Fig. 4.9(k), where it is correctly following the ring again.

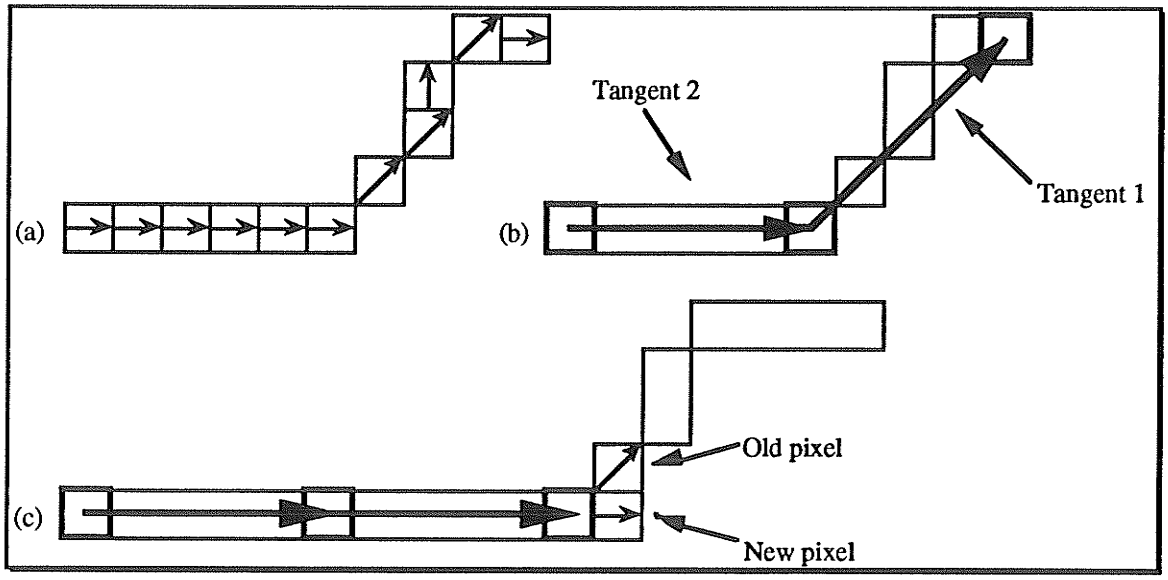


Fig. 4.10 a) The path taken by the algorithm as it shifts over to another ring. b) Assume that the angular difference between the two tangents shown has exceeded the first threshold (the tangents are taken at 5 pixel intervals, starting at the current pixel and working backward) c) The algorithm has backtracked to a point where the angular difference between the tangents is below the second threshold. The algorithm then advances once in the direction of the pixel that was backtracked to.

Although this improvement will prevent the tracking algorithm from turning back on itself in a very short distance, it won't prevent it from jumping across to an adjacent ring. Consider the example in Fig. 4.10. Assume that the tracking algorithm was originally following one ring, and has moved upwards, across the ring gap, continuing on an

adjacent ring. There are no different pixel directions that have occurred, so the algorithm does not know that it has made a sharp turn upwards.

To detect this sort of motion, the local curvature of the path can be checked. Within a digital image, the curvature is calculated as shown in Fig. 4.10. Two approximate tangents are taken, and the difference in their angle is the curvature [ROSE2]. If this curvature passes a certain threshold, then the algorithm must back up to a point where the curvature is less than a second threshold. Once the algorithm has backtracked, it can proceed by moving in a different direction than chosen before.

The first threshold is the minimum curvature which indicates the algorithm has deviated from its proper path. The second threshold is the maximum curvature that is acceptable during normal operation. Simply backing up until the curvature is less than the first threshold may not be far enough, because the curvature could still be too great. By using two thresholds, there is a grace area where the algorithm can operate without causing backtracks. Only severe changes in curvature are detected.

Another way of measuring the curvature can be achieved by comparing the angles of two lines as shown in Fig. 4.11. Rather than having both lines the same length, one is allowed to be longer. If the curvature is checked in this way then the threshold will have to be smaller because the angular difference will be smaller. This can be verified by comparing the angle between the tangents in Fig. 4.10 to that of Fig. 4.11. Either method is appropriate.

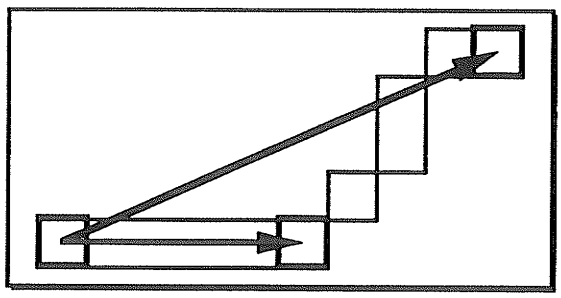


Fig. 4.11 A different way of measuring the curvature.

A different approach to measuring the curvature was investigated. This other method makes use of the fact that, given three points, there is a unique circle that can be drawn such that all three points lie on the circle. One can find the centre of this circle from the following two equations:

$$x = \frac{((2b_x + 2b_y - 2a_x - 2a_y)(2b_y - 2c_y) - (2b_x + 2b_y - 2c_x - 2c_y)(2b_y - 2a_y))}{((2b_x - 2a_x)(2b_y - 2c_y) - (2b_x - 2c_x)(2b_y - 2a_y))}$$

$$y = \frac{((2b_x + 2b_y - 2c_x - 2c_y)(2b_x - 2a_x) - (2b_x + 2b_y - 2a_x - 2a_y)(2b_x - 2c_x))}{((2b_x - 2a_x)(2b_y - 2c_y) - (2b_x - 2c_x)(2b_y - 2a_y))}$$

where (a_x,a_y), (b_x,b_y), and (c_x,c_y) are the three points. At each step in the tracking procedure, the current point as well as two previous points are used to find the centre of the unique circle that passes through them. If this circle has a small radius, then the curvature is high, and vice versa. It does not work well in practice though. To detect a high curvature, the points have to be relatively close. This can give rise to an extremely large radius for most situations (e.g., if the points are collinear, the radius will be infinite). The choice of a threshold is then made very difficult. In addition, the digital nature of the curve on which the points lie can introduce error.

Once the high curvature has been detected and the algorithm has backtracked, it can continue in one of three ways. The first is as described earlier. Simply move once in the direction of the new current pixel (as shown in Fig. 4.10). The other two ways make use of the tangent and the curvature respectively.

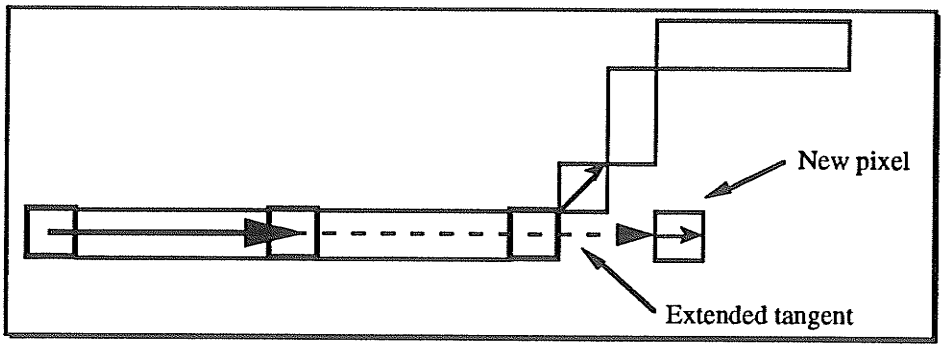


Fig. 4.12 Tangent 1 is extended a certain distance, and the new pixel position is taken at the end.

The purpose of backtracking was so that a possible error could be corrected. If the algorithm proceeds once along a new direction by only one pixel, there is a possibility that the same error could occur. It may be better to make a much longer jump, thereby crossing the area that caused the error. This can be done by proceeding a certain length along the tangent, and continuing from there. Fig. 4.12 shows an example of this using the same situation as in Fig. 4.10.

Another method of continuing uses the curvature that initiated the backtrack. The algorithm simply moves in a direction opposite the curvature that resulted in the error. In Fig. 4.10(b), the angle of tangent 1 minus the angle tangent 2 is positive, indicating a positive curvature. After backtracking, the algorithm can then proceed in a direction opposite the curvature. An example is shown in Fig. 4.13. The problem area can then be avoided.

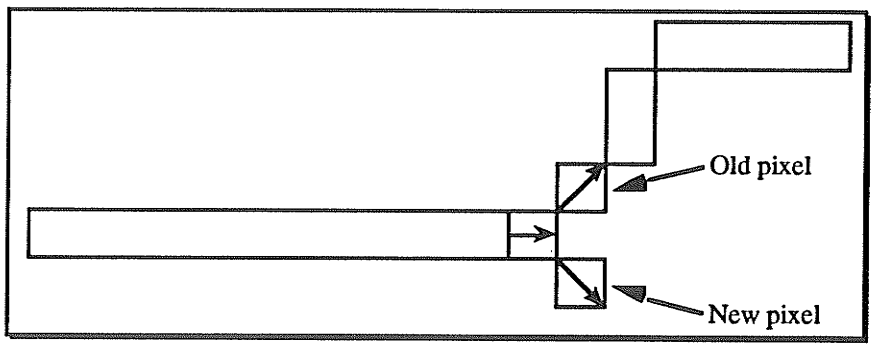


Fig. 4.13 After backtracking, the algorithm moves along a negative curvature path rather than a positive one.

An example of where the detection of a large curvature helps avoid error is shown in Fig. 4.14. The minimum intensity tracking method is used with a centre line length of 5 pixels and side line lengths of 3 pixels. The tangents are computed at 15 pixel intervals, starting at the current pixel position and working backward. If the current pixel is n, then the first tangent is computed using the slope of the line between pixel n and n-15. The second tangent is computed using pixels n-15 and n-30. The minimum threshold before a gross curvature is detected is 30 degrees. Once this threshold is exceeded, the algorithm

will back up one pixel at a time until the curvature is less than 6 degrees. These numbers are arbitrary and were chosen because they represent reasonable thresholds for the particular resolution the rings were scanned in at.

In Fig. 4.14(a), the algorithm has detected a reversed direction. After backtracking and continuing it once again detects a reversed direction (Fig. 4.14(b)). For a third time it detects a reversed direction and backtracks again (Fig. 4.14(c)). This time however, when the algorithm continues, it follows a straight line path to the upper left. If it were left to continue it would eventually connect with an adjacent ring. However, once it has reached the stage shown in Fig. 4.14(d), the computed curvature has exceeded the threshold of 30 degrees. Therefore it backtracks until the curvature is below 6 degrees. The method for continuation in this case, is to extend the tangent and start at the end. The lower circled pixel in Fig. 4.14(d) is where the algorithm has backtracked to. The upper circled pixel is where the algorithm has jumped. The new pixel was arrived at by multiplying the length of the tangent line by 1.2. In relative coordinates, the start of the tangent line is at (0,0) (15 pixels back, but not visible in the image) and the end at (-1,15). Multiplying both Δx and Δy by 1.2 gives the new pixel coordinate at (-1,18). The new pixel happens to coincide with a pixel that was already tracked. However, the tracking directions are different. The new pixel direction is set to that of the backtracked pixel, which happens to be northwest. At the previous time that the pixel was encountered, the direction of movement was northeast. With the new direction of movement, the algorithm correctly crosses the ring gap. Fig. 4.14(e) illustrates the algorithm two steps along from Fig. 4.14(d). Notice the newly marked pixel. From there the algorithm proceeds in a predominantly northern direction as shown in Fig. 4.14(f).

These enhancements are more suited to the minimum intensity tracking method. Each time either algorithm encounters an error, it must backtrack and continue in another direction. To continue properly, the subpixelic method must reconnect with the edge of the ring. The minimum intensity algorithm can continue successfully if reconnected to any part

of the ring. This larger area results in a higher probability that the algorithm will successfully continue on the same ring.

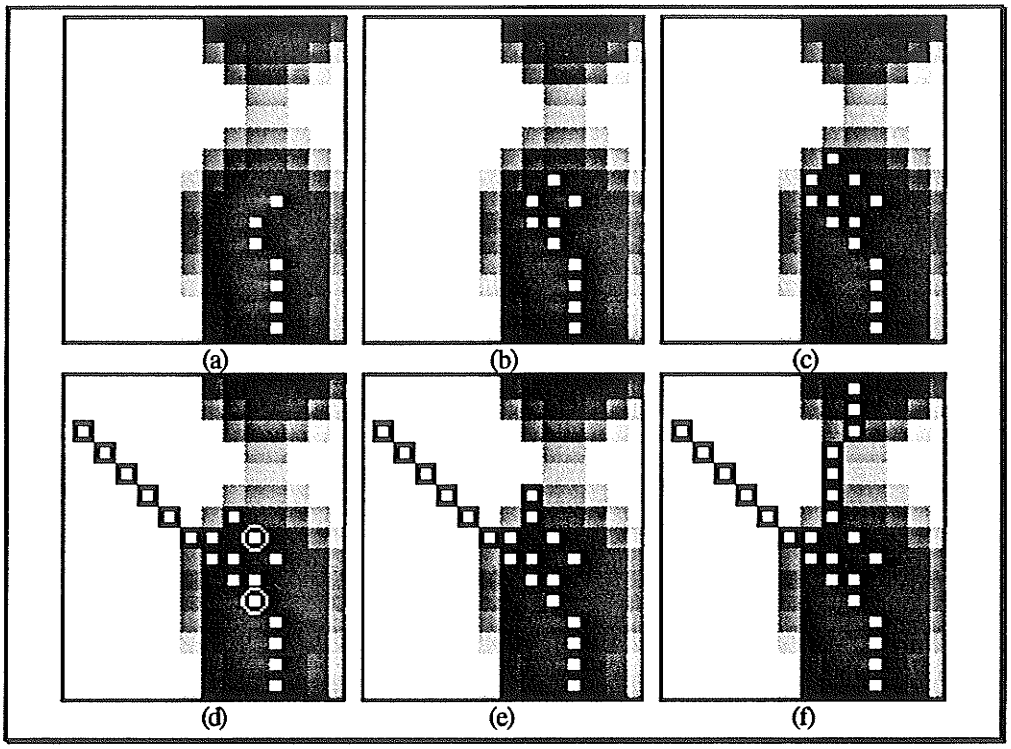


Fig. 4.14 (a)-(c) Detection of reversed direction three times. d) Large curvature detected and algorithm has backtracked and jumped forward using the tangent. e) The algorithm has corrected its path. f) The algorithm has connected with the other side of the ring break.

4.5 Linking Edge Pixels

It was mentioned in the introduction that the linking of edge pixels is not as desirable as edge tracking. The absence of many edge pixels does not allow one to obtain a continuous chain that traverses the entire circumference of a ring. It is possible to obtain many small segments of a ring, but they would then have to be connected manually to obtain a complete chain.

In an attempt to overcome this problem, the edges can be extended with the anticipation that some of the shorter segments will become attached to others. Consider the output from the LoG operator with zero-crossings detected. Each edge pixel will be

associated with one of eight edge directions. To extend the edges, the direction of each pixel as well as its neighbours must be examined. Which neighbours are examined depends upon the edge direction of the current pixel. If there isn't a neighbour lying one pixel ahead in this direction as well as one pixel 45 degrees to the left and right, then a new edge point is placed at the neighbouring pixel corresponding to the current edge direction. The edge direction at this new edge pixel is set to the same direction as the pixel being examined. An example of this is shown in Fig. 4.15. If there is a neighbour in any of these three directions, then no new edge pixel is added.

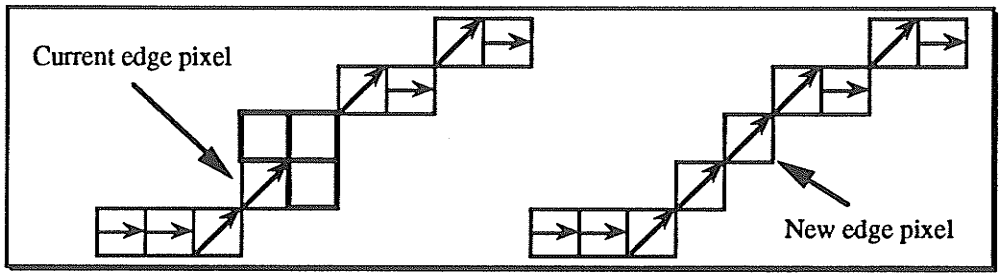


Fig. 4.15 The directions of the edge at each pixel are shown as arrows. The neighbours to be checked at the current pixel are shown in bold. Since there aren't any neighbouring pixels, a new one is added in the direction of the edge at the current pixel. The two segments are now connected.

This process can be iterated a number of times. The number of iterations needed corresponds to how large the gaps are between edge segments. If there is a five pixel wide gap, then five iterations would be needed. Errors can occur though, when the ends of edge segments are extended but never connect to a neighbouring segment. In this case it is possible that the segment may be extended into a segment that is part of an adjacent ring. This would severely affect a tracking algorithm. An example of extending edge segments is shown in Fig. 4.16. Notice that while some of the gaps are closed, others are not. The instances where the gaps aren't closed is due to the direction of the edge at those points. The edge points are extended past the neighbouring edge pixels, without ever connecting. If left to continue in this manner, they would eventually connect with incorrect segments. This could be corrected by searching longer distances for neighbouring edge pixels.

However, the larger the area that is searched, the higher the probability that an edge segment could be extended into an incorrect one.

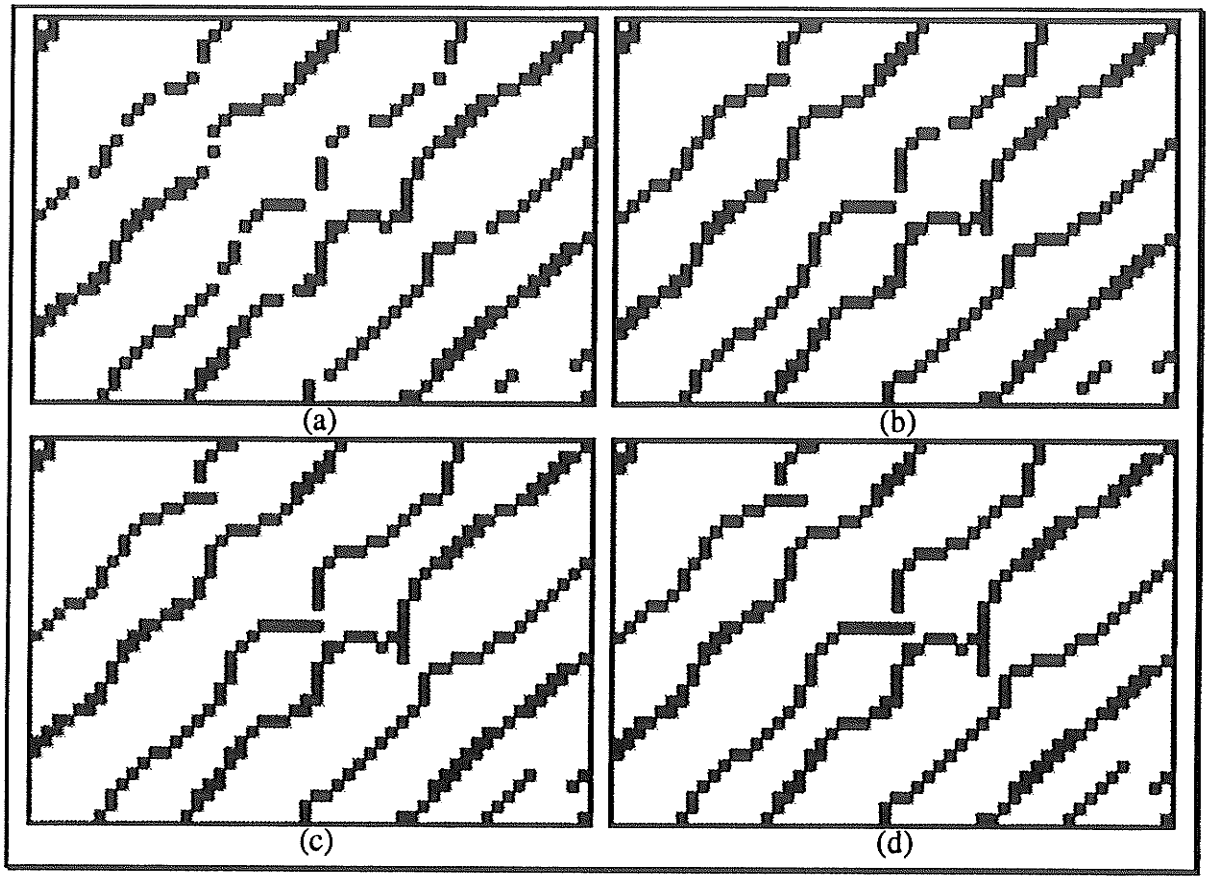


Fig. 4.16 a) Test image 1 edge detected with the LoG operator (w=9) and zero crossings located. (b)-(d) One, two, and three iterations of edge pixel extension, respectively.

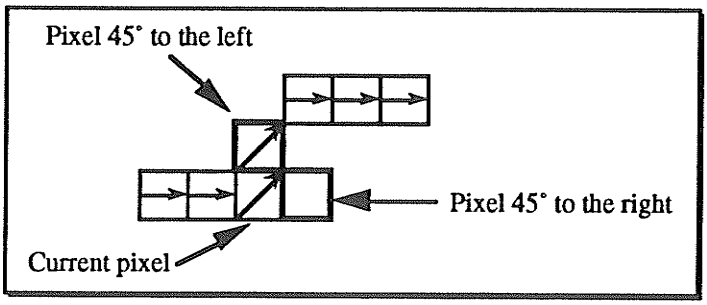


Fig. 4.17 The direction of the edge at the current pixel is northeast. An edge pixel does not lie in that direction, therefore the pixels 45° to the left and right must be checked as well. The pixel 45° to the left is an edge pixel, so the algorithm can continue.

To actually track the ring edges using only the edge pixel and edge direction information is a simple matter. Once a starting point is given, the algorithm will follow a

path based upon the edge direction at each pixel. There are cases where the current edge direction will not point towards the next pixel, in which case the two pixels 45 degrees to the right and left of the current edge direction must be checked as well (see Fig. 4.17).

Unfortunately, there are going to be gaps in the edges even if the segments are extended. In Fig. 4.16, there are two single pixel wide gaps that are never closed. The algorithm will stop when it reaches these gaps. This is shown in Fig. 4.18. The edge directions will point in one direction on one side of the ring, and in the reversed direction on the other side. This is due to the way the predicate matching algorithm assigns directions. At each zero crossing, the positive side will always be on the right of the direction of the edge, and the negative side on the left. In Fig. 4.18(a), the algorithm was started at the bottom of the image, on the left edge of the middle ring. Once it reached the gap, the algorithm stopped and had to be restarted in order to continue tracking the ring edge (Fig. 4.18(b)). A different problem occurs on the right ring edge. The small extension near the middle that was added when the segments were extended (as can be verified in Fig. 4.16) caused the algorithm to veer off its intended course (Fig. 4.18(c)). Once again the algorithm had to be restarted. Note that the direction of tracking is opposite to that of the other ring edge.

The advantages of tracking a ring edge in this way are the simplicity as well as the accuracy, due to the single pixel wide edges. However, the narrow path that is tracked can cause many problems, such as those mentioned above. It is certainly possible to track individual segments with the algorithm, and then combine them at a later time, but this would require excess manual control. Unfortunately, the enhancements that are useful to the other two tracking methods, do not work well for this approach. It is much more difficult to continue along a proper path after an error has been detected, because there are so few edge points.

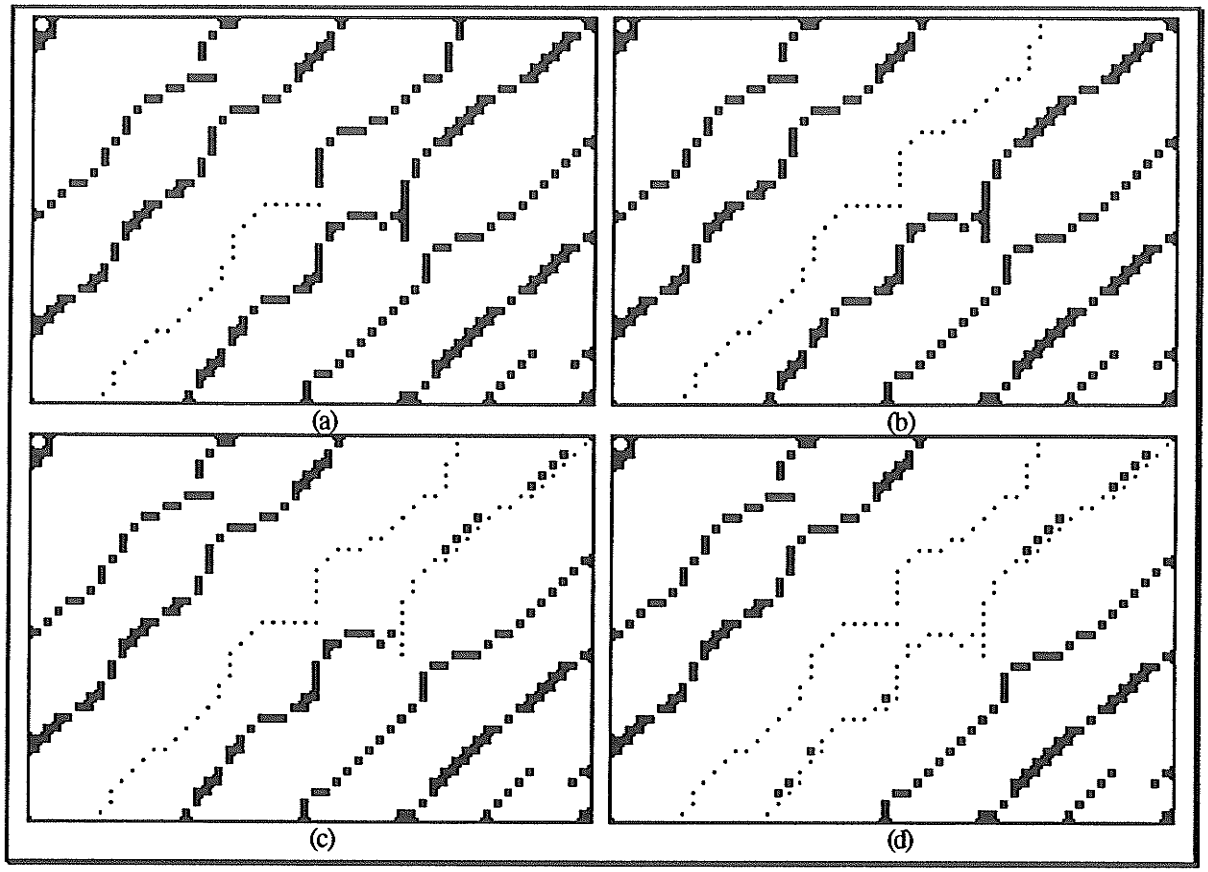


Fig. 4.18 a) The algorithm has stopped because of a gap. b) The algorithm was restarted. c) The opposite ring edge was tracked, but the algorithm stopped because of the small extension. d) The algorithm was restarted.

4.6 Handling Incomplete Rings

In many cases, it may not be possible to track a ring completely. The algorithm may track a large portion of the ring with only a small section missing. If a function is derived from the points, then the missing portion can be approximated. Splines are suitable for such cases.

Periodic cubic splines were used. The periodicity is natural to a tree ring. The choice of a cubic spline allows both C^1 and C^2 continuity. This will ensure a smooth curve around the entire circumference of the ring. The accuracy of the approximation to the missing portion will depend upon its length in relation to the diameter of the ring. A relatively large missing portion such as that demonstrated in Fig. 4.19(b), will result in a poor

approximation, while a smaller missing section will result in a good approximation (Fig. 4.19(d)).

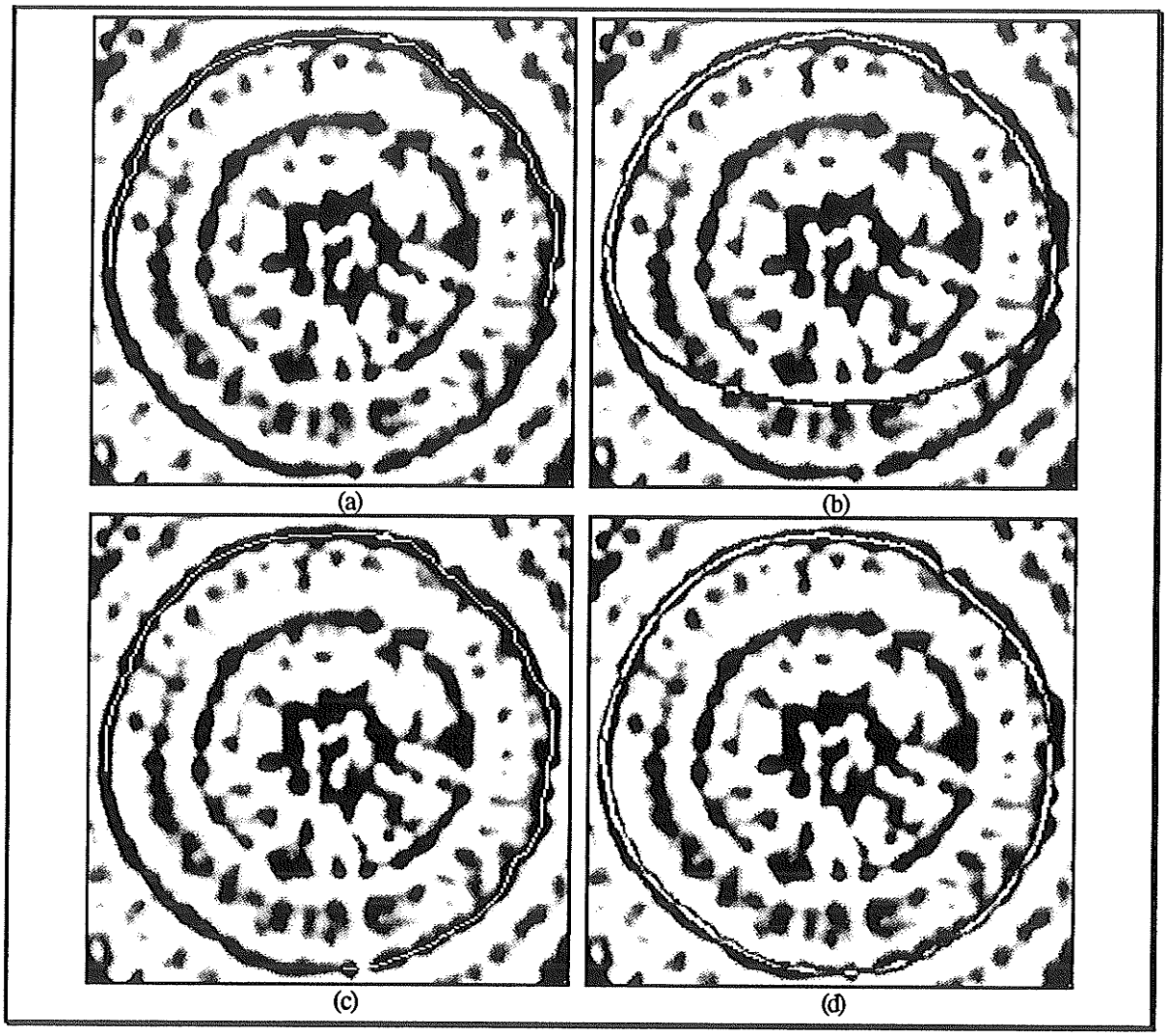


Fig. 4.19 a) A ring that was only tracked half way around. b) The resulting spline (a poor approximation of the missing section). c) The algorithm is allowed to track more of the ring. d) The resulting spline (a good approximation of the missing section).

There may be cases where more than two disjoint segments of a ring can be tracked successfully, but not the entire ring. This would be a frequent situation if edge tracking was based upon edge direction information (because of the large number of edge gaps that occur). It can also occur with the other methods, as will be seen in Chapter 5, but are less frequent (depending upon the clarity of the biscuit image). Since splines are ultimately

going to be used to interpolate the entire ring, the points obtained from separately tracking the disjoint sections can be combined to form a periodic spline.

4.7 Minimum Radial Inertia

This section is concerned with a less conventional algorithm to detect contours. It is based upon the concept of *radial inertia* [GRAT]. Given a curve that approximates the contour and is defined by some parameters, the radial inertia function can be used to obtain these parameters. It works primarily with radially connected contours, which is ideal for tree rings. A radial line drawn through such a contour will cross it only once.

The algorithm works with a bounded sub-domain D of a function $f(x,y)$. Each point (x,y) in D can be expressed in polar coordinates as follows

$$x = \rho\cos(\theta) + x_0$$
$$y = \rho\sin(\theta) + y_0$$

with the pole situated at (x_0, y_0). The curve $c(\theta)$ being derived can be expressed as

$$x = c(\theta)\cos(\theta) + x_0$$
$$y = c(\theta)\sin(\theta) + y_0$$

With respect to the curve $c(\theta)$ and pole (x_0, y_0), the radial inertia of $f(x,y)$ taken over the sub-domain D is defined as [GRAT]

$$\int\int_D f[\cos(\theta) + x_0, \sin(\theta) + y_0][\rho - c(\theta)]^2 \rho d\rho d\theta. \qquad (1.1)$$

In discrete form it is defined as

$$\sum_{i \in D} f_i[\rho_i - c(\theta_i)]^2$$

where f_i is a pixel intensity, ρ_i is the pixel's radial distance from the pole, and θ_i is its angle relative to the pole. The domain D is simply the set of pixels that define the contour of interest. Equation 1.1 is related to the *moment of inertia*. The moment of inertia of a function $f(x)$ about the y axis is defined as follows [PURC]:

$$\int_a^b x^2 f(x) dx$$

The radially connected curve used in [GRAT] is a periodic cubic B-spline. Given m knots, the curve is defined as

$$c(\theta) = \sum_{i=1}^{m} r_i b_i(\theta) = \mathbf{r}^{\mathrm{T}}\mathbf{b}(\theta)$$

where r_i is the parameter corresponding to the spline b_i. The right hand side is in vector form, with the vector lengths equal to the number of knots m.

With the curve defined in this way, the radial inertia can be written as

$$q_0 - 2\mathbf{q}^{\mathrm{T}}\mathbf{r} + \mathbf{r}^{\mathrm{T}}\mathbf{Q}\mathbf{r} \qquad (1.1)$$

where

$$q_0 = \sum_{i \in D} f_i \rho_i^2$$

$$\mathbf{q} = \sum_{i \in D} f_i \rho_i \mathbf{b}(\theta_i)$$

$$\mathbf{Q} = \sum_{i \in D} f_i \mathbf{b}(\theta_i)\left[\mathbf{b}(\theta_i)\right]^{\mathrm{T}}.$$

To find the minimum radial inertia, the first partial differences with respect to r_i (i=1...m) are equated to zero. This results in the following equation:

$$\mathbf{r}_{\mathrm{opt}} = \mathbf{Q}^{-1}\mathbf{q}.$$

Using the algorithm to obtain B-splines for the tree rings is a straightforward approach. The image must first be edge detected. The best method for achieving this is with the LoG operator. When working on a particular ring, the sub-domain used must not include pixels from other rings. To achieve this, the ring can be bounded on the inside and outside by two polygons. Thus only those pixels lying inside the larger polygon, but outside the smaller one, are used. However, this would require careful placement of the polygons. In the sample biscuit image, the rings are rather circular, so a circular bound can be used. This only requires the placement of the centre of the circle and then a specification of the radius.

To simplify the matter, the algorithm can be applied to the centre ring first, and then work outwards. A circular bound is placed upon the centre ring, the ring is processed and then all pixels lying within the bounds are discarded. A circular bound is placed around the next ring, which is then processed, and again the pixels within the bounds are discarded. This ensures that the only pixels being processed at each stage are those that belong to the current ring.

If a ring is not circular, then a polygonal bound is needed. Due to the similar structure that rings exhibit, it may be possible to simply enlarge the initial polygonal bound (by moving each vertex away from the centroid) in order to enclose succeeding rings, but this idea was not implemented.

For an example, consider the steps taken in Fig. 4.20. To begin, a circular bound is placed around the centre ring. The algorithm is applied, resulting in a B-spline. This area is then erased, and a circular bound is placed around the next ring. The algorithm is applied again to produce another B-spline, and the procedure continues as before. For efficiency reasons, two circular areas can be used to bound the inside and outside of the ring. This reduces the number of pixels needed in the computation.

Although there was a fair amount of manual control involved in the example, the results are very promising. The B-splines accurately approximate the rings. This sort of method can be used in situations where the tracking methods cannot successfully track enough of a ring to derive a properly interpolating B-spline. The drawback tends to be the trouble in properly bounding the rings so that neighbouring rings do not affect the resulting B-spline. Extra pixels either belonging to other rings or occurring because of noise can skew the B-spline. It was suggested in [GRAT] that the procedure can be iterated using restricted sub domains based upon the B-spline from the previous iteration. The new sub-domain can consist of those pixels lying a certain distance from the B-spline in the previous step. This lessens the effect of noise. As can be seen in Fig. 4.20, excellent approximations can be achieved without iteration.

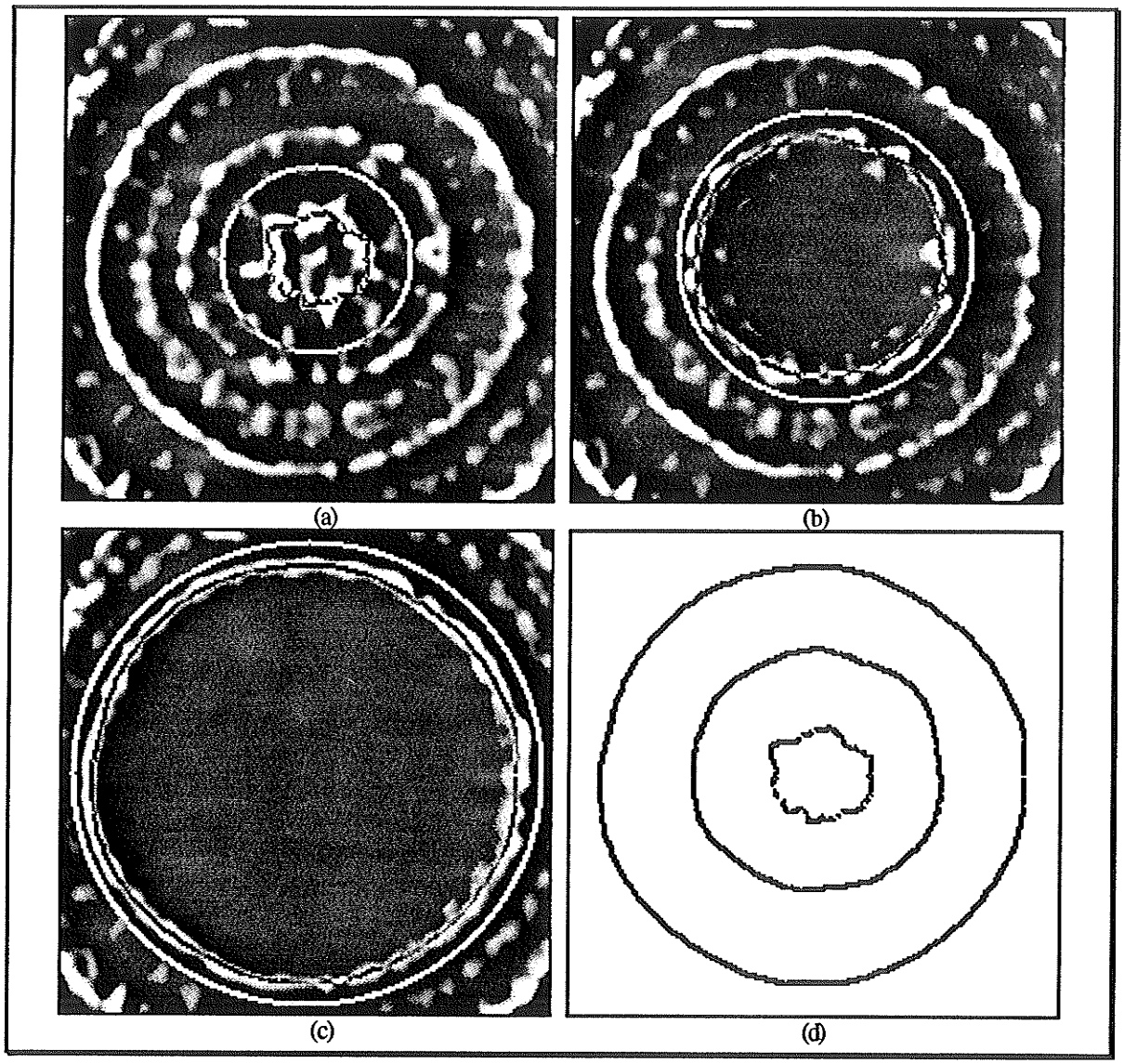


Fig. 4.20 The B-splines were drawn in exclusive-or mode to make them more visible a) A bounding circle has been placed around the centre knot and the B-spline was computed. b) The bounding area from (a) was erased, a new bounding circle was placed around the first ring, and the B-spline was computed. c) The same procedure was repeated for the second ring. In this case the noisy data between the first and second rings was also erased by defining a circular bound just inside the third ring. d) The three B-splines that were computed.

4.8 Conclusions

The minimum intensity algorithm out-performs the other two methods. It can traverse small ring breaks and is not effected by noise as much as the subpixelic method. Its

drawback lies in the fact that it can only traverse the centre of a ring which means that data on the ring's width cannot be obtained.

The subpixelic tracking method can be used to track both sides of a ring, making it possible to obtain data on the ring's width. However, it is very sensitive to noise and cannot traverse ring breaks as easily. It is also not very suited to the enhancements described.

Tracking based upon edge direction information is the simplest method, but it must have complete ring edges if it is going to successfully traverse the entire circumference of a ring. The problems in obtaining complete ring edges make this tracking method undesirable.

Chapter 5
Applications

5.1 Introduction

This chapter ties together the investigations undertaken in the previous chapters. In section 5.2, a description of the software that was used to test all of the algorithms is described. Section 5.2 is a case study which outlines the procedures that can be used to extract tree rings. The final two sections describe two applications of this research. The first describes a procedure for counting rings algorithmically and the second on how to obtain relative ring areas.

5.2 Software

In order to test the image enhancement, edge detection and edge tracking techniques, an application was developed. It incorporates all of the image processing and ring analysis techniques discussed in Chapters 2, 3, and 4. Extra features were added in order to simplify the study of these methods.

For the LoG edge detection method, both a serial and parallel version were implemented. For the parallel version, a T805-30 transputer was used. The motivation for implementing a parallel version was the large amount of processing time required by the LoG operator. A 35x35 operator requires 1225 multiplications for each pixel. The sample biscuit image's dimensions are 1296x1161 (1,504,656 pixels). Thus, 1,843,203,600

multiplications would be required. On a SPARCstation 1, this equates to over an hour of processing time.

With most image processing techniques, including LoG edge detection, parallelization is straightforward. If there are n nodes being used on the transputer, then the image is simply split up into n blocks, one for each node. The host processor sends the blocks to each of the nodes via a message passing protocol. Each node processes its respective sub-image and sends the data back to the host processor which combines the processed blocks into the final image.

The main application was developed under the UNIX operating system on a Sun SPARCstation using the C programming language. The interface was created with the XView 2.0 libraries and runs under the X Window System. All images were stored using the Tagged Image File Format (TIFF).

The main part of the interface consists of a single control window (Fig. 5.1). The control window contains the menus and various buttons and input areas. The image lies within the image window which has sliders for viewing different sections of the image (assuming the image is larger than the image window). There are three other windows that can be displayed or hidden under menu control. The first is the enlarge window which allows the user to display an enlarged portion of the main image. This window is also used to initiate edge tracking. The larger pixels make it easier for selecting a starting point. By clicking on one of the pixels in the enlarge window, the currently selected edge tracking algorithm will be initiated. The second window is for displaying a ring cross-section selected by the user. The third window is simply for displaying a histogram of the colours in the image. The cross-section and histogram windows have horizontal sliders that are used to view different sections of the data that may lie outside the window area. Each of the three windows can be resized, and the respective data that they display will be rescaled or redrawn, where appropriate.

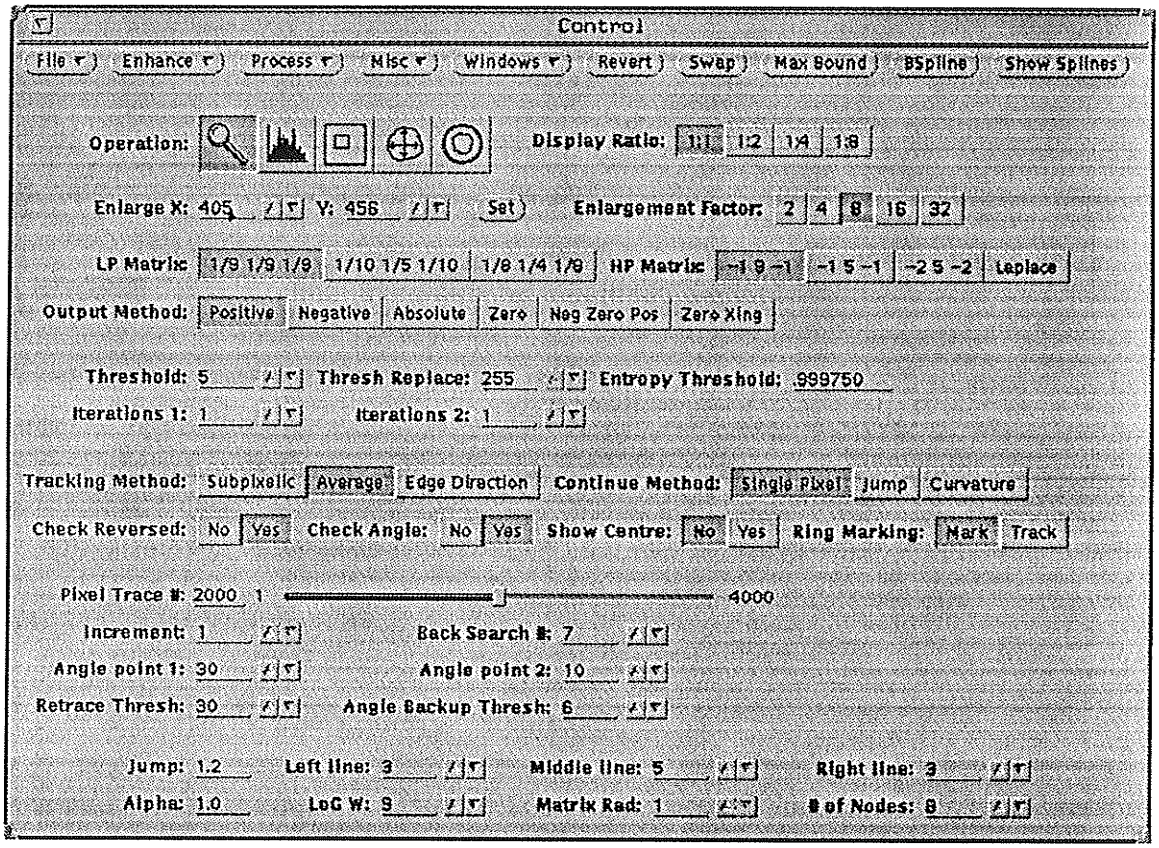


Fig. 5.1 Application control window.

The menu bar in the control window contains the following menus and functions:

File: Loading and saving of TIFF files as well as quitting the program.

Enhance: Execution of the image enhancement techniques discussed in Chapter 2.

Process: Execution of the edge detection techniques discussed in Chapter 3.

Misc: Miscellaneous functions: computing histogram, extending edge pixels, reversing the image, and marking rings based upon a ring cross section (explained below).

Windows: Opening or closing the enlarge, histogram and cross-section windows.

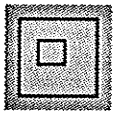
There is a backup image retained at all times for undoing any image processing operation. There are two buttons in the menu bar that make use of this. The revert button is for loading the backup image into the image window. The swap button is for replacing the

backup image with the image that is currently in the image window. This is used if one does not want to undo back to the original image after many operations have been performed. The other buttons in the menu bar will be discussed where appropriate.

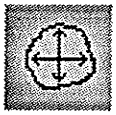
Below the menu items are various user selectable items. Each one is described in turn, beginning with the first row. The first item, **Operation**, has four buttons that are used to select which operation can be performed in the image window with the mouse. These operations are indicated by the following icons:

Enlarge: Clicking on a point will display an enlarged version of the area in the enlarge window.

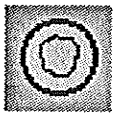
Cross-section: A line can be drawn anywhere in the image, and the gray levels of the pixels lying on the line are used to draw a graph in the cross-section window.



Bounds: A rectangle can be drawn to select a sub-area of the image for applying operations to, rather than the whole image.



Area: The inside of a pre-computed spline can be selected in order to count the number of pixels within its bounds using a flood-fill algorithm.



Circular Bounds: A circle can be drawn to select a sub-area of the image for applying the minimum radial inertia algorithm.

Each operation is activated by clicking the left mouse button in the image window. For obtaining a cross-section, a click and drag method is used to place the line. The same method is used for selecting the processing and radial inertia bounds, except that a rectangle and a circle are displayed, respectively.

The display ratio can be changed by selecting one of the buttons of the **Display Ratio** item. Either the whole image is displayed in the image window (1:1), or reduced versions can be displayed corresponding to 1/4th, 1/16th and 1/64th of the size of the original. The **Enlarge X** and **Y** buttons are used for manually entering an enlargement

point, which is then displayed in the enlarge window by clicking the **Set** button. Next to the **Set** button is the **Enlargement Factor** item which has five buttons for controlling the enlargement scale. Enlargements of 2x, 4x, 8x, 16x and 32x can be selected.

The **LP Matrix** and **HP Matrix** items are for selecting the different low-pass and high-pass convolution kernels discussed in Chapter 2. The **Output Method** item is used for changing the output method for those edge detection algorithms that produce both positive and negative values. The first button is for outputting only positive values. Negative values are set to a zero gray level. The second button is for outputting the absolute value of negative values, and setting positive values to a zero gray level. The third button selects absolute output, showing both positive and negative values. The fourth button selects zero value output. Zeros are set to maximum gray level (255) while positive and negative values are set to gray level zero. The **Neg Zero Pos** button will result in negative values being set to a gray level of zero, zero values set to a gray level of 128, and positive values set to a gray level of 255. The **Zero Xing** button is used only for the LoG edge detection method. If selected, the zero-crossings will be found using the predicate matching algorithm, and output at gray level 255.

The **Threshold** item selects the value for thresholding, and the **Thresh Replace** item indicates the gray level that those pixels lying below the threshold are set to. The **Entropy Threshold** item is used for both the grayscale and colour entropy edge detection methods. On the next line are two items for setting the number of iterations used by the iterative edge detection methods. The first iteration number is for the edge detection phase and the second one is for the edge thinning phase.

Below these items are the ones used for the edge tracking process. Most of these items are supplied so that the user can change values used during this process. In cases where the algorithm fails to traverse the entire circumference of a ring, some of these values can be modified to possibly enable the algorithm to succeed. However, being able to change these values does not guarantee that the tracking algorithm will always succeed.

The first item is for selecting which tracking method is to be used. This can be either the subpixelic method, the minimum intensity method, or tracking based upon edge direction information obtained during the edge detection process. How the tracking algorithm proceeds after backtracking is selected with the **Continuation Method** item.

The tracking enhancements discussed in Chapter 4 can be turned on or off with the **Check Reversed** and **Check Curvature** items. These control the tests for reversed direction and large curvature, respectively. The **Show Centre** item is simply for displaying the approximate centre of the ring. In Chapter 4, a method for checking the curvature was discussed that made use of three points to compute a circle. If the option is on, a pixel will be drawn at the centre of this circle at each stage in the tracking process. This does not affect the tracking algorithm, and is included for display purposes only.

Once a cross-section line has been drawn on the image, the ring marking algorithm can be run to mark on this line where it thinks there are rings. Normally, any points that are detected as rings, are marked with an X. Another option is to deploy the ring tracking algorithm at each one of these points, so that multiple rings can be tracked with one operation. These two choices are controlled by the **Ring Marking** item.

The next group of items are used for changing the parameters affecting the current tracking algorithm. The number of pixels that are tracked is set with the **Pixel Trace #** slider item. If the tracking method successfully tracks the entire ring, then it will stop when it has reached the starting point, even if the number of pixels tracked has not reached the number input into the item.

Normally, the algorithm will move in single pixel increments. For experimental purposes only, this increment can be changed with the **Increment** item. The number of previously traced pixels that are checked to detect a reversed direction, is set with the **Back Search #** item. Setting this to higher values will allow the algorithm to detect a reversed direction that occurs over a much longer path.

The **Angle point 1** and **Angle point 2** items are used for adjusting the length of the tangents used in detecting a large curvature. If equal length tangents are being used, then only the first item is needed. The lengths of the tangents will be half of this value. Thus, if it is set to 30, then the first tangent will be computed with the current pixel n and the pixel n-15. The second tangent will be computed using pixels n-15 and n-30. If the unequal length tangent method is being used, then both items must be set and the numbers indicate the backward distance at which the two tangents are taken. If the first item is set to 30, and the second one set to 10, the first tangent will be computed using the current pixel n and the pixel n-*30* steps back. The second tangent will be computed using pixel n-10 and pixel n-30.

There are two thresholds used in detecting a large curvature. The first is to detect when the curvature has become too great, and the second one indicates how far back the algorithm must go before continuing. The **Retrace Thresh** and **Angle Backup Thresh** items are for setting these two thresholds. The numbers represent angles from 0 to 360 degrees.

The final group of items are a collection of miscellaneous parameters. When the jump continuation method is selected, the algorithm will jump forward a certain distance after it has backtracked. This distance will be a multiple of the length of the line used to compute the local tangent. This multiple is set with the **Jump** item. The next three items, **Left line**, **Middle line,** and **Right line**, are used to set the lengths of the line averages used for the minimum intensity edge tracking method.

The **Alpha** item is used to set the α parameter used for the optimal recursive edge detection method. The width of the positive central region of the LoG operator is set with the **LoG W** item. The **Matrix Rad** item is used to set the radius of the averaging area used in obtaining the ring cross-section. The final item, **# of Nodes,** is used to set the number of nodes on the transputer that are used in the parallel implementation of LoG edge detection.

5.3 Case Study

This section deals primarily with the procedure that was used to obtain spline approximations to the rings in the sample biscuit image. To begin, the image was edge detected with the LoG operator with a positive central region width of 9. This decision was based upon conclusions provided in Chapter 3. The minimum intensity edge tracking method was used because it has the highest success rate of all three tracking methods. There are two situations where the minimum radial inertia contour detection algorithm was used due to an extremely poor definition of the rings.

Beginning with the rings of the main trunk, the initial parameter settings for the edge tracking were as follows (using the item names from the Control window):

Increment:	1	**Back Search #:**	7
Angle point 1:	30	**Angle point 2:**	10
Retrace Thresh:	30	**Angle Backup Thresh:**	6

Jump: 1.2 **Left line:** 3 **Middle line:** 5 **Right line:** 3

The checks for a reversed direction or a large curvature were enabled. The **Continuation Method** item was initially set to **Single Pixel.**

The tracking algorithm was set to start tracking upwards. An item could have been added to allow this to be changed interactively, but it was not of importance. In most cases, the tracking algorithm was initiated on the left side of the ring, at approximately the 9 o'clock position. With the tracking direction set to north, this results in a clockwise traversal of a ring. In some cases it was necessary to start the algorithm on a different part of a ring to avoid a noisy area that it could not traverse. In these situations, the spline was used to approximate the missing portion.

Starting with the centre ring and working outwards one ring at a time, the procedure was as follows (settings remain the same from one ring to the next unless changed as indicated):

Ring 1: radial inertia method used due to noise

Ring 2: successfully tracked

Ring 3: unsuccessfully tracked when starting from left side of ring, but successfully tracked when starting from right side

Ring 4: successfully tracked; Jump continuation method used

Ring 5,6: successfully tracked

Ring 7: radial inertia method used due to poor definition of ring

Ring 8: successfully tracked

Ring 9: radial inertia method used due to poor definition of ring

Ring 10: successfully tracked; Curvature continuation method used

Ring 11: successfully tracked; Jump continuation method used

Ring 12: successfully tracked

Ring 13: due to poor definition of ring near the bottom, the tracking algorithm was started closer to the problem area; Curvature continuation method used

Ring 14: again, the ring was poorly defined near the bottom so the tracking algorithm was started near the problem area

Ring 15: successfully tracked

Ring 16: algorithm started near problem area; Jump continuation method used; setting the Jump distance to 1.3 allowed algorithm to track further into the problem area

Ring 17-22: successfully tracked; Jump distance set to 1.2

Ring 23,24: algorithm started on right side of ring to avoid problem area

The remaining rings would have to be tracked in more than one section due to increased noisy areas. Many of the external rings are simply too poorly defined to be tracked successfully. The radial inertia algorithm could be used for these rings, but it would be difficult to define the proper boundaries. The resulting splines are shown in Fig. 5.2.

In basically all cases, excellent approximations to the rings were obtained. One particular error can be seen on the ninth spline (starting at the centre spline) near the bottom. The interpolation of a noisy area resulted in a small "glitch" where the ninth spline touches with eighth spline.

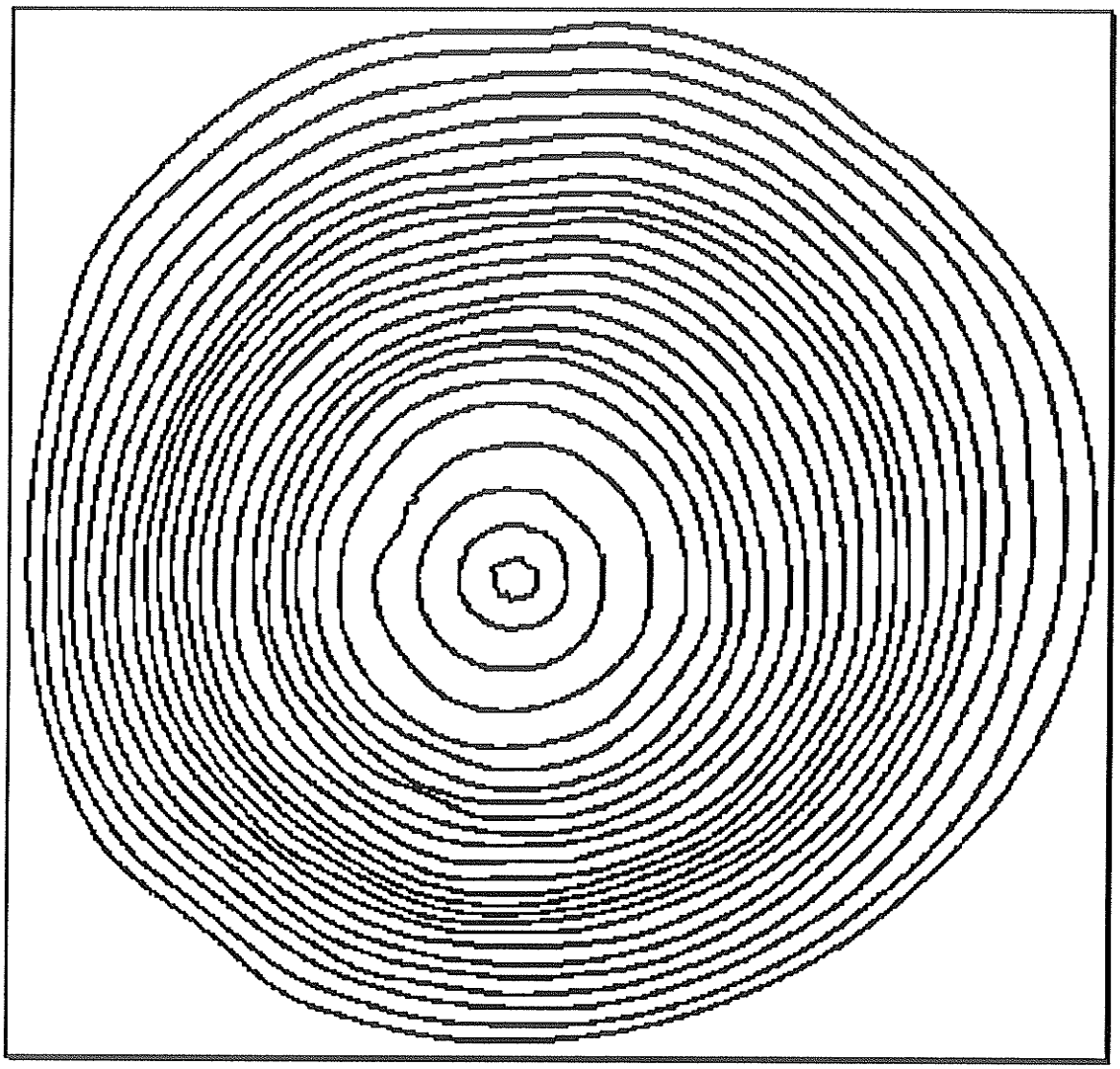


Fig. 5.2 The splines obtained by edge tracking the rings. The centre spline is not a ring, but rather the centre of the tree.

For the mistletoe growth, the centre knot and the innermost ring were not tracked because of the large low-intensity artifact that encompasses them. The initial tracking settings were identical to those above except for four changes. The **Retrace Thresh** item was set to 40 rather than 30 because of the tighter curvatures occurring on the second innermost ring. The **Left line**, **Middle line**, and **Right line** items were set to 4, 6 and 4, respectively. The longer line lengths reduce the chance that the algorithm will reverse because of the generally thicker rings.

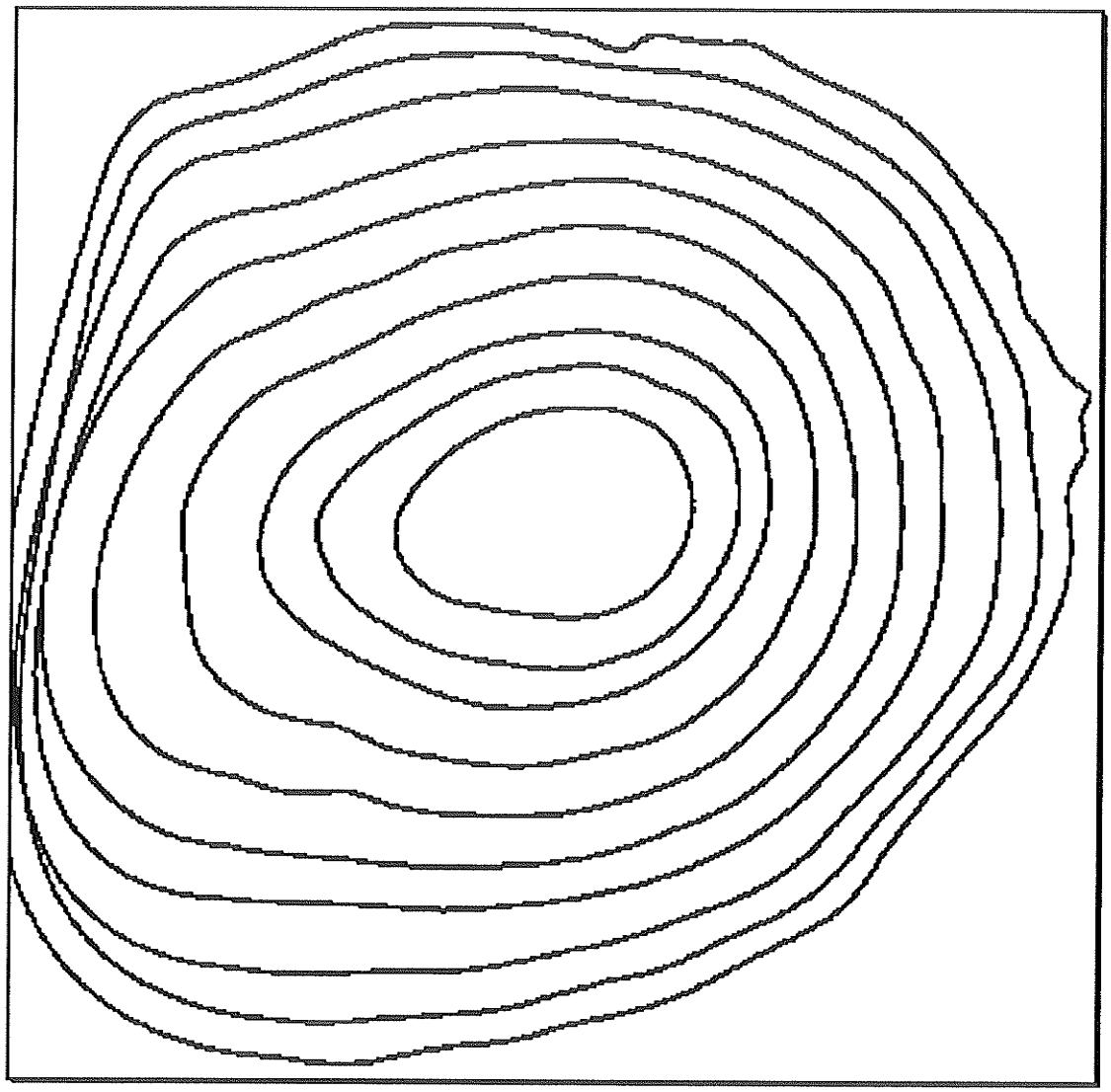


Fig. 5.3 The splines obtained by edge tracking the rings. The missing portion on the left side of the outermost spline occurred because the interpolated section lied outside of the image bounds.

Starting with the second innermost ring, the procedure was as follows (settings remain the same from one ring to the next unless changed as indicated):

Ring 2: algorithm started on the lower right to properly traverse the large gap in the ring at the upper right; Jump continuation method used

Ring 3: successfully tracked; **Retrace Thresh** set to 30 due to smaller curvature of remaining rings

Ring 4,5: successfully tracked; line lengths set to 6, 8, and 6 for left, middle and right, respectively, because rings are thicker

Ring 6: algorithm started on upper left of ring due to noisy area on the left; spline interpolates missing portion

Ring 7-11: algorithm started on upper left of rings due to noisy area; splines interpolate missing portions; line lengths set to 7, 9, and 7 because rings are thicker

Due to the absence of any ring definition on the left of side of the mistletoe, the tracking algorithm could not traverse the rings completely. For rings 6 to 11, the missing sections had to be interpolated with the splines. As can be seen in Fig. 5.3, these interpolations were not very accurate. Again, as with the other half of the biscuit, the outer rings could not be tracked in one section due to a lack of definition. The effects of the increased noise at the outer rings can be seen in the poor spline interpolation of the outermost ring.

5.4 Ring Counting

When studying a biscuit, it is useful to know how many rings there are. This can be accomplished by hand, but if one already has a digital image of the biscuit, it is more efficient to do it algorithmically. A description of such an algorithm follows.

In chapter 3, it was shown that within a cross sectional image, the tree rings resemble inverted roof edges. This cross section was obtained by drawing a radial line across the rings such that it crossed each ring only once. Obtaining the gray levels of each pixel that lies on the line, from start to end, produces an image such as that shown in Fig. 5.4(a).

By traversing the line and counting the number of troughs that occur, an estimate of the number of rings can be obtained. However, a cross section of noisy rings can produce random shallow troughs. This can be seen in Fig 5.4(a). There are only three rings in the image (the deeper troughs) but there are two shallow troughs that occur because of noise. Pre-smoothing the image to remove the noise will improve the definition of the troughs, but extraneous troughs can still occur. Fig. 5.4(b) illustrates the same line cross section, but the biscuit image was smoothed first, using 4 iterations of the K-near noise reduction method, with K=6. There is still one trough that does not correspond to a ring.

Instead of taking the gray level at each pixel on the line, a local average can be used instead. This will reduce the effects of the noise and will produce much smoother varying troughs. The size of the averaging area is important. Fig. 5.4(c) illustrates the same cross-section on the original image, but using a 3x3 local average at each point on the line. Notice that there is still one extra trough. Using a 5x5 local average finally results in only three troughs.

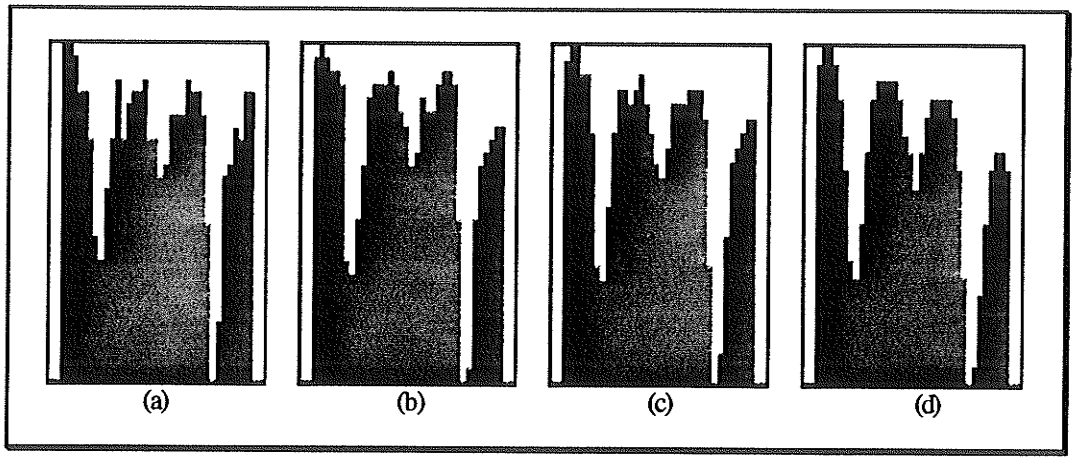


Fig. 5.4 Example cross sections of rings a) Original biscuit image b) Image smoothed with the K-near noise reduction method c) Original, but using a 3x3 average at each point on the line d) 5x5 averaging.

It appears that the use of a local average at each point reduces errors. It is possible that even better results can be achieved by using it upon a pre-smoothed image. However, it would be advantageous to have only the rings present in the image. This can be accomplished through edge detection. The most effective edge detection scheme presented in Chapter 3 was the LoG operator. It can provide an image that consists solely of the rings. There is an added advantage in using this method because the rings will also be very smoothly varying troughs, which is due to the way the LoG operator responds to intensity changes. Other edge detection methods could be used, such as the multi-template edge detector, but it is not appropriate in this case to have both sides of the rings detected because there would be extra troughs.

It is a simple matter to count the troughs within a given cross section. A binary automaton can be used. One state corresponds to the algorithm moving downwards into a trough, and the other for moving upwards out of a trough. If the state changes from up to down, then a peak was encountered. If the state changes from down to up then a trough was encountered. The gray levels corresponding to the previous peak and the previous trough are stored. If a trough is encountered and the difference between the trough gray level and the gray level of the previous peak is above a certain threshold, then the trough is marked as a ring. A threshold is used so that shallow troughs that occur are not considered tree rings.

This algorithm has a low error rate when used on an image that was edge detected with the LoG operator. The positioning of the cross-section line is done manually to allow some discretion in choosing a properly defined set of rings. Fig. 5.5(b) shows an example set of rings with marks on each point that the ring counting algorithm recognized as a ring. The direction of the cross-section line is implicit in the position of the marks. The rings in Fig. 5.5(a) were detected with the LoG edge detection operator using a positive central region width of 9 and a total width of 27. Fig. 5.5(c) shows the cross-section of the rings. The graph was generated by using a 3x3 local average at each point on the cross-section line.

The high accuracy in counting the rings can be attributed to the excellent edge detection abilities of the LoG operator. The faint rings along the left side in Fig 5.5(a) are detected very well and are clearly visible in Fig. 5.5(b). The use of averaging to take points off of the cross-section line allows for the line to cross through a break in a ring and still have it marked. This will occur as long as the averaging area is wide enough to bridge the ring break. An example of this can be seen in Fig 5.5(b). The cross-section line intersected with a very high intensity part of the third line from the left, but a trough was still generated and the ring was marked.

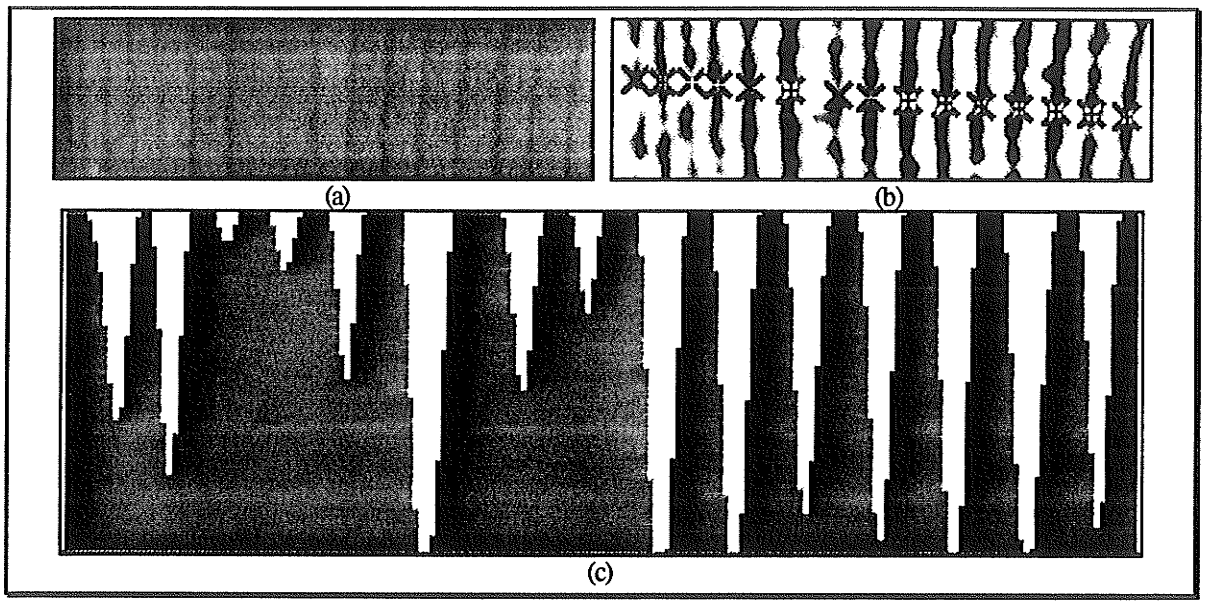


Fig. 5.5 a) Original test area b) Edge detected with LoG operator (w=9) and showing marked rings. c)The cross section used to count the troughs.

Although the accuracy is high, there are problems that can occur. If the cross-section line crosses over a spot edge that was the result of noise, it can result in a spurious trough in the cross-section. This trough will be incorrectly marked as a ring if it is deeper than the threshold. If the cross-section line crosses through a ring break it may be possible that a trough will not be generated and the ring will not get marked. However, a quick visual inspection can verify if such errors have occurred.

5.5 Ring Area

Once a ring has been tracked, and a spline that interpolates this ring is computed, it is possible to obtain an approximation of the ring's area. Each pixel will represent some fraction of an inch, depending upon the scanning resolution used to digitize the biscuit. The sample biscuit image was scanned at 600 dpi, therefore each pixel represents 1/600 of an inch. The numbers returned by the following methods can be divided by the square of the scanning resolution to obtain the ring's area in square inches.

The simplest way to obtain an area is to count the number of pixels that lie within the region bounded by the particular ring. This can be achieved with a simple flood-fill algorithm, since the spline corresponding to the ring will be continuous and define an enclosed area. The flood fill is a recursive algorithm normally used for filling in closed regions with a specific colour. If the border has a colour C and the region is to be filled with this colour then, given a starting point (x,y), the algorithm would be as follows:

```
procedure FloodFill(x, y, C)
{
    if (colour at pixel (x,y) is equal to C)
        return
    set pixel (x,y) to colour C
    FloodFill(x-1, y, C)
    FloodFill(x+1, y, C)
    FloodFill(x, y-1, C)
    FloodFill(x, y+1, C)
}
```

For each pixel that is coloured, a counter can be incremented. This counter can be used as a measure of area and compared with the pixel counts of other rings.

Unfortunately, the flood fill algorithm can overflow the stack quickly and is also very slow. Other area fill methods can be used instead. As a spline is drawn, the points on the curve are connected with straight lines to eliminate the possibility of a discontinuity. The curve is essentially a polygon with a large number of edges. Thus, any existing polygon filling routine can be used to count the number of pixels within the spline boundary.

A more mathematical approach for finding the area is to use Green's theorem. Given a closed differentiable curve C, Green's theorem states that

$$\text{Area} = \frac{1}{2}\int_C (x\,dy - y\,dx)$$

with the integral taken around the contour C. If the curve is defined parametrically, then the above equation becomes

$$\text{Area} = \frac{1}{2}\int_{C}\left(x(t)\frac{dy(t)}{dt} - y(t)\frac{dx(t)}{dt}\right)dt$$

The cubic splines used to approximate the rings are defined parametrically and have C^2 continuity. Given n spline segments, the area is defined as

$$\text{Area} = \frac{1}{2}\sum_{0}^{n-1}\int_{t_j}^{t_{j+1}}\left(x_j(t)\frac{dy_j(t)}{dt} - y_j(t)\frac{dx_j(t)}{dt}\right)dt$$

with spline segment j and its derivatives defined as

$$x_j(t) = A_{jx}(t-t_j)^3 + B_{jx}(t-t_j)^2 + C_{jx}(t-t_j) + D_{jx}$$
$$y_j(t) = A_{jy}(t-t_j)^3 + B_{jy}(t-t_j)^2 + C_{jy}(t-t_j) + D_{jy}$$
$$\frac{dx_j(t)}{dt} = A_{jx}(t-t_j)^2 + B_{jx}(t-t_j) + C_{jx}$$
$$\frac{dy_j(t)}{dt} = A_{jy}(t-t_j)^2 + B_{jy}(t-t_j) + C_{jy}$$

To compute the integrand for spline segment j, Simpson's rule for numerical quadrature is used. This rule is as follows:

$$\int_{a}^{b} f(t)dt = \sum_{j=0}^{n} w_j f(t_j)$$

with w_j defined as

$$w_0 = w_n = \frac{h}{3}$$
$$w_{2k} = \frac{2h}{3} \qquad 2k \neq 0 \text{ and } 2k \neq m$$
$$w_{2k+1} = \frac{4h}{3}$$
$$\text{with} \quad t_j = a + jh \quad \text{and} \quad h = \frac{b-a}{m}$$

where m is the number of subdivisions used to obtain an approximation to the area under the function (m must be even). Larger m will result in a better approximation to the area.

In parametric form, the area of the parametric cubic spline can be written as

$$\frac{1}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{m} w_j\left(x_i(t_i)\frac{dy_i(t_i)}{dt} - y_i(t_i)\frac{dx_i(t_i)}{dt}\right)$$

where n is the number of knots, and the outer summation is taken from knot 0 to knot n-1. This equation will give a better approximation to the area because it is a subpixelic computation. The flood-fill method will only give the area in whole pixels even though the actual spline crosses pixels at a subpixelic resolution.

Chapter 6
Conclusion

6.1 Summary

In the course of developing a strategy for extracting the rings in a biscuit image, various image processing techniques were investigated. Each of these techniques was applied to the sample biscuit image and evaluated.

An initial problem was the noise occurring in the biscuit image. Thus, edge preserving noise reduction methods were examined with the K-near method producing the most favourable results. The reduction of noise significantly improved the results of those edge detectors examined which did not have noise immunity.

The main focus of the study was on edge detection. Many methods were examined and judged on their ability to detect the ring edges in the sample biscuit image. The LoG edge detector proved to be superior to all, in both noise immunity and edge detection ability. This method was thus used to edge detect the image before edge tracking or ring counting was performed.

In order to obtain spline approximations to the rings, three methods of edge tracking were examined. Problems concerning ring breaks and noise prompted the development of enhancements specifically designed to work with the low curvature that tree rings exhibit. With the enhancements applied, the minimum intensity edge tracking method had the

highest success rate in tracking rings along their entire circumference. Thus, this method was used to obtain spline interpolations to the rings in the sample image.

To demonstrate the usefulness in extracting tree rings, two applications were described. An algorithm was developed that uses the edge detected image to count rings along a radial cross section. The second application was the computation of relative ring area,which is easily achieved once an interpolating spline is obtained.

A case study of the sample biscuit image was presented to demonstrate how to obtain spline approximations to the rings. Although not all rings were processed successfully, due to their poor definition, spline approximations were obtained for the inner 24 rings of the main trunk, and 10 rings of the mistletoe. This showed that the procedure can succeed reasonably well under various conditions.

6.2 Further Work

The majority of this work was done on grayscale images. As stated earlier, colour edge detection can provide extra information because differences in intensity and hue can be detected. These merits should be examined upon a properly smoothed colour image. If possible, it should be done on a 24 bit image rather than 8 bit to maintain the maximum amount of information from the original scan.

The advantages of a higher scanning resolution were not investigated due to the large images that can result. The greater detail may reduce error during the edge detection and edge tracking steps. In particular, those rings that are relatively thin or very faint would be better defined if scanned at a higher resolution.

As suggested by Marr and Hildreth [MARR,HILD], the combination of different scales of edge detection would provide more accurate edge localization. In this regard, the simultaneous tracking of the same edges detected at different scales could reduce edge tracking errors. If the edge direction information obtained from the edge detection process

was used to guide the minimum intensity edge tracker, then it might be possible to reduce errors further.

There are probably many more enhancements that can be made to the minimum intensity tracking method. One possibility is to track two adjacent rings at the same time. If the distance between the paths exceeds a certain threshold, or goes below another threshold, then that would indicate one of the paths has deviated from the ring. This, of course, would only work if adjacent rings have a relatively constant separation along their entire circumference. Such a situation may not be very common though.

Although it was not examined, the property of ring density can also be computed algorithmically. This would only require a simple extension to the ring counting algorithm.

References

[AVRA] Gideon Avrahami and Vaughan Pratt, "Sub-Pixel Edge Detection in Character Digitization", Dept. of Computer Science, Stanford University, Stanford CA 94305

[BRAC] R. Bracewell, "The Fourier Transform and its Applications", MacGraw-Hill, New York, 1965

[BRZA] D. Brzakovic, "Rule-Based Multitemplate Edge Detector," Computer Vision, Graphics, and Image Processing, Vol. 53, No. 3, May 1991, 258-268

[CANN] John Canny, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, Nov 1986, 679-698

[CHIN] Roland T. Chin and Chia-Lung Yeh, "Quantitative Evaluation of Some Edge-Preserving Noise-Smoothing Techniques," Computer Vision, Graphics, and Image Processing, Vol 23, 1983, 67-91

[CUMA] Aldo Cumani, "Edge Detection in Multispectral Images," Computer Vision, Graphics, and Image Processing, Vol 53, No. 1, Jan 1991, 40-51

[DAV1] Larry S. Davis, "A Survey of Edge Detection Techniques," Computer Graphics and Image Processing, Vol. 4, 1975, 248-270

[DAV2] L. S. Davis and Azriel Rosenfeld, "Noise Cleaning by Iterated Local Averaging," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-8, No. 9, Sept 1978, 705-710

[DEFE] Ireneusz Defée and Yrjö Neuvo, "Median-Based Zero-Crossing Edge Detectors for Closely Spaced Edges," Computer Vision, Graphics, and Image Processing, Vol. 53, No. 2, March 1991, 196-203

[DER1] Rachid Deriche, "Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector," International Journal of Computer Vision, Vol. 1, No. 2, April 1987, 167-187

[DER2] Rachid Deriche, "Optimal Edge Detection Using Recursive Filtering," 1987 IEEE First International Conference on Computer Vision, 501-505

[EBER] Robert B. Eberlein, "An Iterative Gradient Edge Detection Algorithm," Computer Graphics and Image Processing, Vol. 5, 1976, 245-253

[GASK] Jack. D Gaskill, "Linear Systems, Fourier Transforms, and Optics," John Wiley & Sons, New York, 1978

[GONZ] Rafael C. Gonzalez and Paul Wintz, "Digital Image Processing," Addison-Wesley Publishing Company, 1977

[GRAT] Paolo Grattoni and Fabrizio Pollastri, "A Contour Detection Algorithm Based on the Minimum Radial Inertia (MRI) Criterion," Computer Vision, Graphics, and Image Processing, Vol. 43, 1988, 22-36

[HILD] Ellen C. Hildreth, "The Detection of Intensity Changes by Computer and Biological Vision Systems," Computer Vision, Graphics, and Image Processing, Vol. 22, 1983, 1-27

[HUER] Andres Huertas, "Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Masks," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 5, Sept 1986, 651-664

[KASV] T. Kasvand, "Iterative Edge Detection," Computer Graphics and Image Processing, Vol. 4, 1975, 279-286

[LEIP] R. Leipnik, "The Extended Entropy Uncertainty Principle", Information and Control, 1960, Vol. 3, pp. 18-25

[LIND] Craig A. Lindley, "Practical Image Processing in C," John Wiley & Sons, 1991

[MARR] D. Marr and E. Hildreth, "Theory of Edge Detection," Proceedings of the Royal Society of London, Vol. B 207, 1980, 187-217

[NAGA] Makoto Nagao and Takashi Matsuyama, "Edge Preserving Smoothing," Computer Graphics and Image Processing, Vol. 9, 1979, 394-407

[OPPE] Alan V. Oppenheim and Ronald W. Schafer, "Digital Signal Processing", Prentice-Hall Inc., New Jersey, 1975

[PURC] Edwin J. Purcell, "Calculus with Analytic Geometry", Meredith Publishing Company, 1965

[ROSE1] Azriel Rosenfeld "Digital Picture Processing: 1st Edition," Academic Press, New York, 1976

[ROSE2] Azriel Rosenfeld, "Digital Picture Processing: 2nd Edition," Academic Press, New York, 1982

[SHIO] Akira Shiozaki, "Edge Extraction Using Entropy Operator," Computer Vision, Graphics, and Image Processing, Vol. 36, 1986, 1-9

[ZENZ] Silvano Di Zenzo, "A Note on the Gradient of a Multi-Image," Computer Vision, Graphics, and Image Processing, Vol. 33, 1986, 116-125