THE INVESTIGATION OF THE CHARACTERISATION OF FLOTATION FROTHS AND DESIGN OF A MACHINE VISION SYSTEM FOR MONITORING THE OPERATION OF A FLOTATION CELL ORE CONCENTRATOR

by

Paul James Symonds

A dissertation submitted to the Faculty of Engineering
in fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING

at the

UNIVERSITY OF CAPE TOWN

Cape Town

June, 1992

UCT Academic Supervisor : Prof. G. de Jager

MINTEK Liaison Officer : Dr. T. B. Lange

Pilot plant flotation cells used for experimental flotation runs.

To my Parents, for all the encouragement and support through the years.

If you are planning for a year,
plant rice
If you are planning for 10 years,
plant trees
If you are planning for 100 years,
plant education in the
minds of the young.

An old Chinese proverb.

Declaration

I Paul James Symonds hereby declare that this dissertation is my own unaided work. This dissertation is being submitted for the degree of Master of Science in Engineering at the University of Cape Town and it has not been submitted before for any degree or examination, in any other university.

Signature of candidate : signature removed (P. Symonds) Date : 11th day of June 1992.

Acknowledgements

I would like to thank :

- my supervisor, Professor Gerhard de Jager, for all his time, encouragement and guidance throughout the project. His knowledge on, interest in and enthusiasm for the field of digital image processing provided a stimulating and enjoyable working environment.
- the Council for Mineral Technology (MINTEK), for sponsoring this research work.
- Dr. T.B. Lange of the Measurement and Control Division, MINTEK, for supplying all the data for the project and also for all his advice on morphological image processing.
- Martin C. Harris of the Mineral Processing Division, Department of Chemical Engineering, for all his assistance on froth flotation.
- the image processing group of the Department of Electrical and Electronic Engineering for all their advice and in particular Wayne Borchardt, Greg Cox, Fred Hoare and Gary Kay, for contributing to such an enjoyable and challenging working environment.

Abstract

This dissertation investigates the application of digital image processing techniques in the development of a machine vision system that is capable of characterising the froth structures prevalent on the surface of industrial flotation cells. At present, there is no instrument available that has the ability to measure the size and shape of the bubbles that constitute the surface froth. For this reason, research into a vision based system for surface froth characterisation has been undertaken. Being able to measure bubble size and shape would have far reaching consequences, not only in enhancing the understanding of the flotation process but also in the control and optimization of flotation cells.

The bubbles that constitute the surface froth in flotation cells give rise to complex structures that are difficult to segment using standard segmentation techniques. A technique for segmenting surface froth images has been developed using morphological image processing. Under suitable lighting conditions, the segmentation technique accurately and reliably extracts the bubble boundaries resulting in segmented images that consist of an interconnected network of lines, which give rise to isolated regions or blobs. These blobs are representative of the individual bubbles and are processed to provide a statistical description of the surface froth.

Results show that although the system does not mimic the human visual process exactly, the mean size and shape measurements it produces are capable of differentiating between various surface froth structures. It is therefore possible to use this information to monitor the performance of flotation cells and to assist in the automated control of these cells.

Table of Contents

Acknowledgements		iv
Abstract		v
Table of Contents		vi
List of Figures		xi
List of Tables		xviii
Glossary		xix
CHAPTER 1:	**INTRODUCTION**	1-1
1.1	Format of this Dissertation	1-2
CHAPTER 2:	**MINERAL EXTRACTION AND CONCENTRATION BY FROTH FLOTATION**	2-1
2.1	Mineral Processing	2-2
2.2	Flotation Fundamentals	2-7
2.2.1	The Flotation Cell	2-7
2.2.2	The Physics of Flotation	2-10
(A)	The Hydrodynamics of Bubbles	2-10
(B)	Contact Angles and Three-Phase Interfaces	2-12
2.2.3	The Chemistry of Flotation	2-13
(A)	Collectors	2-14
(B)	Frothers	2-14
(C)	Regulators	2-15
2.3	Froth Structures in Flotation	2-16
2.3.1	Types of Froth Structures	2-16
2.3.2	Factors Affecting Froth Structure	2-16
(A)	Froth Height	2-16
(B)	Froth Drainage	2-17
(C)	Froth Stability	2-18
(D)	Froth Aeration	2-18

2.4 The Analysis of Froth Structures using On-line Machine Vision 2-19
2.4.1 The Motivation behind the use of Machine Vision 2-19
2.4.2 The Control of Flotation Cells 2-20
2.5 Industrial Flotation 2-22
2.5.1 Flotation Machines 2-22
2.5.2 Flotation Circuits 2-24
2.6 The Contribution of this Dissertation 2-26
2.6.1 Previous Work 2-26
2.6.2 Current Work 2-28

CHAPTER 3: MACHINE VISION AND THE RESEARCH EQUIPMENT CONFIGURATION 3-1
3.1 Constituent Components of a Machine Vision System 3-2
3.1.1 Illumination Module 3-2
3.1.2 Image Aquisition and Digitization Module 3-4
3.1.3 Image Processing Module 3-6
3.2 Equipment 3-7
3.2.1 Research Equipment Configuration 3-7
3.2.2 Equipment Limitations 3-9
3.2.3 Requirements for a Dedicated System 3-10

CHAPTER 4: THE SEGMENTATION OF SURFACE FROTH IMAGES 4-1
4.1 Visual Features that Characterise Froth 4-2
4.1.1 Bubble highlights 4-2
4.1.2 Bubble boundaries 4-6
4.2 Segmentation Problems 4-6
4.3 Approach to the Segmentation Problem 4-6

CHAPTER 5: THE CLASSICAL SEGMENTATION OF SURFACE FROTH IMAGES 5-1
5.1 Bubble Highlight Segmentation using Automatic Threshold Selection 5-3
5.1.1 Case Study : Moment-Preserving Bilevel Thresholding 5-4

5.2 Bubble Boundary Segmentation using Edge Detection Techniques 5-10
5.2.1 Case Study 1 : The Sobel Operator 5-12
5.2.2 Case Study 2 : The Laplacian Operator 5-15
5.2.3 Case Study 3 : A Valley Operator 5-16
5.3 Discussion 5-19

CHAPTER 6: THE MORPHOLOGICAL SEGMENTATION OF SURFACE FROTH IMAGES 6-1
6.1 Morphological Image Processing 6-2
6.1.1 Binary Morphology 6-5
6.1.2 Grey Level Morphology 6-6
6.1.3 The Affect of Morphological Operations on Images 6-7
6.2 Morphologic Edge Detection 6-7
6.2.1 Rolling Ball Algorithm 6-9
6.2.2 Rolling Ball Implementation 6-11
6.3 Surface Froth Segmentation using Morphologic Edge Detection 6-15
6.3.1 Bubble Highlight Extraction 6-15
6.3.2 Bubble Boundary Extraction 6-18
6.4 Improving Bubble Boundary Segmentation 6-21
6.4.1 Image Preprocessing 6-21
(A) Low-pass Filtering 6-21
(B) Median Filtering 6-22
(C) Discussion 6-22
6.4.2 Image Postprocessing 6-23
(A) Cleaning 6-23
(B) Thinning 6-26
(C) Discussion 6-27
6.5 The Segmentation of Suboptimal Data 6-29
6.6 Measuring the Effectiveness of the Segmentation Technique 6-30
6.6.1 Working with an Area of Interest 6-31
6.6.2 The Manual Segmentation of Surface froth Images 6-31
6.6.3 The Automatic Segmentation of Surface Froth Images 6-32

6.6.4 The Efficiency of the Segmentation Technique 6-33
6.6.5 Determination of the Optimal Structuring Element Size 6-35

CHAPTER 7: THE ANALYSIS AND CHARACTERISATION OF SEGMENTED SURFACE FROTH IMAGES 7-1

7.1 Size Determination 7-2
7.2 Shape Determination 7-2
7.2.1 Circularity Measure 7-2
7.2.2 Elipticity Measure 7-3
(A) Moments 7-3
(B) Central Moments 7-4
7.3 The Parameterization of Blobs 7-7
7.4 Segmented Surface Froth Analysis 7-8
7.4.1 Boundary Tracing 7-8
7.4.2 Connected Component Analysis 7-9
7.5 The Statistical Characterisation of Surface Froths 7-16
7.5.1 Statistical Analysis 7-16
(A) Size and Shape Distributions 7-19
(B) Chi-square Test as a Measure of "Goodness-of-Fit" 7-23
(C) Modelling the Size Distributions 7-26

CHAPTER 8: THE RESULTS OF CHARACTERISING VARIOUS SURFACE FROTH IMAGES 8-1

8.1 Format of the Results 8-2
8.2 Processed Images 8-3
8.2.1 BUB2.CFI 8-4
8.2.2 BUB3.CFI 8-9
8.2.3 BUB4.CFI 8-14
8.2.4 BUB5.CFI 8-19
8.2.5 BUB6.CFI 8-24
8.2.6 BUB7.CFI 8-29
8.2.7 BUB8.CFI 8-34

8.2.8 BUB9.CFI 8-39
8.2.9 BUB10.CFI 8-44
8.2.10 BUB11.CFI 8-49
8.2.11 BUB12.CFI 8-54
8.3 Discussion of the Results 8-59
8.3.1 The Size Distributions 8-59
8.3.2 The Circularity Distributions 8-63
8.3.3 The Elipticity Distributions 8-64
8.4 A Control Concept Using Mean Bubble Size and Elipticity Measurements 8-64
8.4.1 The 3-Sigma Control Chart 8-64
8.4.2 A 3-Sigma Control Scattergram 8-66

CHAPTER 9: CONCLUSION AND RECOMMENDATIONS 9-1
9.1 Conclusion 9-1
9.2 Recommendations for Future Research 9-3
9.2.1 System Optimization 9-3
9.2.2 System Validation 9-5

References

Appendix A Segmentation Efficiency and Bubble Identification as a Function of SE Radius
Appendix B The Derivation of an Elipticity Measure Using Central Moments
Appendix C Connected Component Analysis - A Detailed Explanation
Appendix D Photographs of the Surface Froth Images Processed
Appendix E Tabulated Results for the Processed Images
Appendix F Software Listings

List of Figures

Figure 2.1 : Line flow sheet of plant operations involving liberation and separation. 2-3
Figure 2.2 : Separator objectives. 2-3
Figure 2.3 : Basic principle of a separator. 2-4
Figure 2.4 : Particle size ranges for concentrating equipment. 2-5
Figure 2.5 : The grade/recovery curve for a separation process. 2-6
Figure 2.6 : A simple dispersed air flotation machine. 2-8
Figure 2.7 : Transport paths of materials in flotation. 2-9
Figure 2.8 : Froth flotation bubbles. 2-9
Figure 2.9 : The terminal velocity of air bubbles in water at 20°C. 2-11
Figure 2.10 : Idealized representation of three-phase equilibrium contact between air, water and mineral surface. 2-12
Figure 2.11 : Collector adsorption on mineral surface. 2-14
Figure 2.12 : Action of the frother. 2-15
Figure 2.13 : The variation of froth structure with froth height. 2-17
Figure 2.14 : Froth drainage. 2-18
Figure 2.15 : A flotation circuit control concept. 2-21
Figure 2.16 : Flow sheet of a simple flotation plant. 2-23
Figure 2.17 : Commonly used flotation machines. (a) Agitair cell, and (b) Wemco Fagergren cell. 2-24
Figure 2.18 : Rougher-scavenger-cleaner system. 2-25

Figure 3.1 : Modular representation of a typical machine vision system. 3-3
Figure 3.2 : Structured lighting techniques. (a) grid coding and (b) light stripping. 3-4
Figure 3.3 : Equipment configuration. 3-8

Figure 4.1 : The definitions of the angles i, e and g. 4-3
Figure 4.2 : Bubble highlight formation for an idealised spherically shaped bubble. 4-5
Figure 4.3 : Bubble highlight formation for an idealised elongated bubble. 4-5

Figure 5.1 : Test image - BUB1.CFI. 5-2
Figure 5.2 : BUB1.CFI intensity profile for row 256. 5-2
Figure 5.3 : Grey level histogram. 5-3
Figure 5.4 : Algorithm for the implementation of moment-preserving bilevel thresholding. 5-8
Figure 5.5 : Grey level probability histogram of BUB1.CFI. 5-9
Figure 5.6 : Thresholded image of BUB1.CFI. 5-9
Figure 5.7 : Elements of edge detection by derivative operators. (a) Light object on a dark background and (b) dark object on a light background. 5-11
Figure 5.8 : Simplified bubble intensity profile indicating possible edges for extraction. 5-12
Figure 5.9 : A 3x3 image region. 5-13
Figure 5.10 : BUB1.CFI edge detected using a 3x3 Sobel operator. 5-14
Figure 5.11 : Thresholded, edge detected image of BUB1.CFI. 5-14
Figure 5.12 : BUB1.CFI edge detected using the Laplacian operator. 5-15
Figure 5.13 : 5x5 Valley operator mask. 5-16
Figure 5.14 : Algorithm for the implementation of the 5x5 valley operator. 5-17
Figure 5.15 : Segmentation of BUB1.CFI using the 5x5 valley operator with $T_1=5$ and $T_2=20$. 5-17
Figure 5.16 : Segmentation of BUB1.CFI using the 5x5 valley operator with $T_1=5$ and $T_2=55$. 5-18
Figure 5.17 : Segmentation of BUB1.CFI using the 5x5 valley operator with $T_1=15$ and $T_2=20$. 5-18

Figure 6.1 : An illustration of the concept of morphological operations, using a 1-D signal. (a) Original signal f(x), (b) dilated signal $f_d(x)$, (c) eroded signal $f_e(x)$, (d) closed signal $f_c(x)$ and (e) opened signal $f_o(x)$. 6-3

Figure 6.2 : Binary dilation or erosion, using a 3x3 square SE. 6-5
Figure 6.3 : The umbra U(f) for a 1-D function f(x). 6-6
Figure 6.4 : Some morphological operations on an image; (a) original image, (b) dilated, (c) eroded, (d) opened and (e) closed images. 6-8
Figure 6.5 : 1-D representation of the Rolling Ball Algorithm. (a) Original surface f(x), (b) spherically closed surface c(x), and (c) residual surface surface $G_c(x)$. 6-10
Figure 6.6 : Pseudo code for generating DSSEs of radius R pixels. 6-11
Figure 6.7 : DSSE of radius 3 pixels. (a) 3-D surface plot. (b) numerical values. 6-12
Figure 6.8 : DSSE of radius 5 pixels. (a) 3-D surface plot. (b) numerical values. 6-13
Figure 6.9 : Pseudo code for the implementation of grey level dilation. 6-14
Figure 6.10 : Pseudo code for the implementation of grey level erosion. 6-14
Figure 6.11 : Run-times of morphological closing using DSSEs, on two computer architectures. 6-15
Figure 6.12 : $G_o(x,y)$ for a DSSE of radius 3 and T = 6. 6-16
Figure 6.13 : $G_o(x,y)$ for a DSSE of radius 5 and T = 10. 6-17
Figure 6.14 : $G_o(x,y)$ for a DSSE of radius 7 and T = 14. 6-17
Figure 6.15 : $G_o(x,y)$ for a DSSE of radius 9 and T = 18. 6-18
Figure 6.16 : $G_c(x,y)$ for a DSSE of radius 3 and T = 1. 6-19
Figure 6.17 : $G_c(x,y)$ for a DSSE of radius 5 and T = 1. 6-19
Figure 6.18 : $G_c(x,y)$ for a DSSE of radius 7 and T = 1. 6-20
Figure 6.19 : $G_c(x,y)$ for a DSSE of radius 9 and T = 1. 6-21
Figure 6.20 : $G_c(x,y)$ for a DSSE of radius 9, after low-pass filtering BUB1.CFI. 6-22
Figure 6.21 : $G_c(x,y)$ for a DSSE of radius 9, after median filtering BUB1.CFI. 6-23
Figure 6.22 : Figure 6.20 opened with a 2x1 SE. 6-24
Figure 6.23 : Figure 6.22 opened with a 1x2 SE. 6-25
Figure 6.24 : Figure 6.23 closed with a 3x3 + SE. 6-25
Figure 6.25 : Neighbourhood arrangement used by the thinning algorithm. 6-26
Figure 6.26 : Figure 6.24 thinned to provide the boundaries. 6-28
Figure 6.27 : BUB1.CFI - the segmented image overlayed on the original image. 6-28
Figure 6.28 : BUB13.CFI - the segmented image overlayed on the original image. 6-29
Figure 6.29 : BUB14.CFI - the segmented image overlayed on the original image. 6-30
Figure 6.30 : BUB1.CFI - manually segmented image showing the identified bubbles within the AOI. 6-32
Figure 6.31 : Efficiency curve for BUB1.CFI. 6-34
Figure 6.32 : Optimal SE curve for BUB1.CFI. 6-35
Figure 6.33 : BUB1.CFI optimal SE curve, after performing three point averaging of the data. 6-36

Figure 7.1 : Definition of the area of a blob b(x,y). 7-2
Figure 7.2 : Object location using Centroids. 7-4
Figure 7.3 : Boundary trace direction codes. 7-9
Figure 7.4 : Located blob, using boundary tracing. 7-10
Figure 7.5 : An image line for illustrating run-length coding. 7-11
Figure 7.6 : Located blob, using connectivity analysis. 7-12
Figure 7.7 : The processing of erroneous boundary connections. 7-13
Figure 7.8 : Run-times of the full processing of surface froth images, for DSSEs of radii 3 to 20 pixels. 7-14
Figure 7.9 : The analysis of the AOI of the optimally segmented BUB1.CFI. 7-15
Figure 7.10 : Figure 7.9 overlayed on the original image. 7-15
Figure 7.11 : The mean, median and mode for (a) Normally distributed data. (b) Positively skewed data. (c) Negatively skewed data. 7-17
Figure 7.12 : Size distributions for the test image BUB1.CFI : (a) Expected distribution and (b) Observed distribution. 7-20
Figure 7.13 : Circularity distributions for the test image BUB1.CFI : (a) Expected distribution and (b) Observed distribution. 7-21
Figure 7.14 : Elipticity distributions for the test image BUB1.CFI : (a) Expected distribution and (b) Observed distribution. 7-22
Figure 7.15 : Log normal distribution, with $X_m = 10$ and $\sigma_g = 2$. 7-27

Figure 7.16 : Log normal fit to the expected size distribution, with best fit parameters μ = 527.67 and σ = 580.38. 7-28
Figure 7.17 : Log normal fit to the observed size distribution, with best fit parameters μ = 659.16 and σ = 759.69. 7-29

Figure 8.1 : BUB2.CFI - original image. 8-4
Figure 8.2 : Manually segmented image. 8-5
Figure 8.3 : Automatically segmented image, using the optimal DSSE of radius 10 pixels. 8-5
Figure 8.4 : Segmentation efficiency curve. 8-6
Figure 8.5 : Three point averaged optimal DSSE curve. 8-6
Figure 8.6 : (a) Expected size distribution with μ = 590.21, σ = 763.48, skewness = 3.99, Q_{50} = 335.00 and mode = 163.00.
(b) Observed size distribution with μ = 632.43, σ = 645.49, skewness = 3.06, Q_{50} = 432.50 and mode = 416.00. 8-7
Figure 8.7 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 552.44 and σ = 574.81.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 645.10 and σ = 724.40. 8-7
Figure 8.8 : (a) Expected circularity distribution with μ = 2.35, σ = 0.46, skewness = 1.35, Q_{50} = 2.25 and mode = 2.08.
(b) Observed circularity distribution with μ = 2.16, σ = 0.53, skewness = 2.13, Q_{50} = 1.98 and mode = 1.99. 8-8
Figure 8.9 : (a) Expected elipticity distribution with μ = 1.63, σ = 0.47, skewness = 1.38, Q_{50} = 1.52 and mode = 1.34.
(b) Observed elipticity distribution with μ = 1.56, σ = 0.38, skewness = 1.40, Q_{50} = 1.47 and mode = 1.29. 8-8
Figure 8.10 : BUB3.CFI - original image. 8-9
Figure 8.11 : Manually segmented image. 8-10
Figure 8.12 : Automatically segmented image, using the optimal DSSE of radius 11 pixels. 8-10
Figure 8.13 : Segmentation efficiency curve. 8-11
Figure 8.14 : Three point averaged optimal DSSE curve. 8-11
Figure 8.15 : (a) Expected size distribution with μ = 464.23, σ = 706.26, skewness = 5.70, Q_{50} = 252.00 and mode = 200.00.
(b) Observed size distribution with μ = 551.93, σ = 657.10, skewness = 5.09, Q_{50} = 357.50 and mode = 86.00. 8-12
Figure 8.16 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 423.04 and σ = 431.73.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 535.77 and σ = 517.09. 8-12
Figure 8.17 : (a) Expected circularity distribution with μ = 2.27, σ = 0.35, skewness = 0.81, Q_{50} = 2.22 and mode = 2.06.
(b) Observed circularity distribution with μ = 2.13, σ = 0.61, skewness = 4.15, Q_{50} = 1.98 and mode = 1.80. 8-13
Figure 8.18 : (a) Expected elipticity distribution with μ = 1.58, σ = 0.40, skewness = 1.37, Q_{50} = 1.47 and mode = 1.35.
(b) Observed elipticity distribution with μ = 1.55, σ = 0.37, skewness = 1.14, Q_{50} = 1.47 and mode = 1.25. 8-13
Figure 8.19 : BUB4.CFI - original image. 8-14
Figure 8.20 : Manually segmented image. 8-15
Figure 8.21 : Automatically segmented image, using the optimal DSSE of radius 10 pixels. 8-15
Figure 8.22 : Segmentation efficiency curve. 8-16
Figure 8.23 : Three point averaged optimal DSSE curve. 8-16

Figure 8.24 : (a) Expected size distribution with μ = 513.51, σ = 582.31, skewness = 2.91, Q_{50} = 295.00 and mode = 219.00.
(b) Observed size distribution with μ = 548.16, σ = 483.36, skewness = 1.99, Q_{50} = 410.50 and mode = 235.00. 8-17
Figure 8.25 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 488.10 and σ = 476.49.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 568.95 and σ = 616.70. 8-17
Figure 8.26 : (a) Expected circularity distribution with μ = 2.34, σ = 0.46, skewness = 1.27, Q_{50} = 2.23 and mode = 1.86.
(b) Observed circularity distribution with μ = 2.07, σ = 0.43, skewness = 1.93, Q_{50} = 1.97 and mode = 1.84. 8-18
Figure 8.27 : (a) Expected elipticity distribution with μ = 1.69, σ = 0.48, skewness = 1.04, Q_{50} = 1.57 and mode = 1.19.
(b) Observed elipticity distribution with μ = 1.58, σ = 0.38, skewness = 1.13, Q_{50} = 1.46 and mode = 1.37. 8-18
Figure 8.28 : BUB5.CFI - original image. 8-19
Figure 8.29 : Manually segmented image. 8-20
Figure 8.30 : Automatically segmented image, using the optimal DSSE of radius 10 pixels. 8-20
Figure 8.31 : Segmentation efficiency curve. 8-21
Figure 8.32 : Three point averaged optimal DSSE curve. 8-21
Figure 8.33 : (a) Expected size distribution with μ = 467.44, σ = 973.27, skewness = 6.25, Q_{50} = 201.00 and mode = 112.00.
(b) Observed size distribution with μ = 469.33, σ = 600.40, skewness = 3.70, Q_{50} = 261.00 and mode = 110.00. 8-22
Figure 8.34 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 383.14 and σ = 469.55.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 459.64 and σ = 584.17. 8-22
Figure 8.35 : (a) Expected circularity distribution with μ = 2.18, σ = 0.39, skewness = 0.88, Q_{50} = 2.11 and mode = 1.93.
(b) Observed circularity distribution with μ = 2.33, σ = 1.02, skewness = 3.50, Q_{50} = 1.99 and mode = 1.78. 8-23
Figure 8.36 : (a) Expected elipticity distribution with μ = 1.77, σ = 0.64, skewness = 2.42, Q_{50} = 1.59 and mode = 1.27.
(b) Observed elipticity distribution with μ = 1.70, σ = 0.51, skewness = 2.90, Q_{50} = 1.62 and mode = 1.32. 8-23
Figure 8.37 : BUB6.CFI - original image. 8-24
Figure 8.38 : Manually segmented image. 8-25
Figure 8.39 : Automatically segmented image, using the optimal DSSE of radius 10 pixels. 8-25
Figure 8.40 : Segmentation efficiency curve. 8-26
Figure 8.41 : Three point averaged optimal DSSE curve. 8-26
Figure 8.42 : (a) Expected size distribution with μ = 372.13, σ = 677.51, skewness = 6.09, Q_{50} = 181.00 and mode = 100.00.
(b) Observed size distribution with μ = 391.61, σ = 441.22, skewness = 5.01, Q_{50} = 275.00 and mode = 68.00. 8-27
Figure 8.43 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 320.43 and σ = 39.60.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 390.03 and σ = 416.34. 8-27
Figure 8.44 : (a) Expected circularity distribution with μ = 2.20, σ = 0.44, skewness = 1.51, Q_{50} = 2.11 and mode = 1.97.
(b) Observed circularity distribution with μ = 2.22, σ = 0.75, skewness = 4.38, Q_{50} = 2.03 and mode = 1.86. 8-28

Figure 8.45 : (a) Expected elipticity distribution with μ = 1.77, σ = 0.61, skewness = 1.99, Q_{50} = 1.63 and mode = 1.19.
(b) Observed elipticity distribution with μ = 1.69, σ = 0.46, skewness = 2.12, Q_{50} = 1.64 and mode = 1.49. 8-28
Figure 8.46 : BUB7.CFI - original image. 8-29
Figure 8.47 : Manually segmented image. 8-30
Figure 8.48 : Automatically segmented image, using the optimal DSSE of radius 12 pixels. 8-30
Figure 8.49 : Segmentation efficiency curve. 8-31
Figure 8.50 : Three point averaged optimal DSSE curve. 8-31
Figure 8.51 : (a) Expected size distribution with μ = 414.73, σ = 695.67, skewness = 4.75, Q_{50} = 195.00 and mode = 112.00.
(b) Observed size distribution with μ = 428.45, σ = 450.09, skewness = 3.17, Q_{50} = 298.00 and mode = 158.00. 8-32
Figure 8.52 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 364.54 and σ = 436.58.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 425.05 and σ = 437.26. 8-32
Figure 8.53 : (a) Expected circularity distribution with μ = 2.24, σ = 0.55, skewness = 2.55, Q_{50} = 2.16 and mode = 2.28.
(b) Observed circularity distribution with μ = 2.15, σ = 0.63, skewness = 2.57, Q_{50} = 1.97 and mode = 1.74. 8-33
Figure 8.54 : (a) Expected elipticity distribution with μ = 1.74, σ = 0.59, skewness = 1.83, Q_{50} = 1.60 and mode = 1.43.
(b) Observed elipticity distribution with μ = 1.60, σ = 0.43, skewness = 1.16, Q_{50} = 1.49 and mode = 1.31. 8-33
Figure 8.55 : BUB8.CFI - original image. 8-34
Figure 8.56 : Manually segmented image. 8-35
Figure 8.57 : Automatically segmented image, using the optimal DSSE of radius 11 pixels. 8-35
Figure 8.58 : Segmentation efficiency curve. 8-36
Figure 8.59 : Three point averaged optimal DSSE curve. 8-36
Figure 8.60 : (a) Expected size distribution with μ = 378.54, σ = 823.55, skewness = 5.81, Q_{50} = 160.00 and mode = 91.00.
(b) Observed size distribution with μ = 399.73, σ = 485.24, skewness = 3.74, Q_{50} = 255.00 and mode = 145.00. 8-37
Figure 8.61 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 300.34 and σ = 348.56.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 389.82 and σ = 445.23. 8-37
Figure 8.62 : (a) Expected circularity distribution with μ = 2.05, σ = 0.39, skewness = 1.98, Q_{50} = 1.98 and mode = 1.98.
(b) Observed circularity distribution with μ = 2.22, σ = 0.81, skewness = 3.78, Q_{50} = 1.99 and mode = 1.87. 8-38
Figure 8.63 : (a) Expected elipticity distribution with μ = 1.67, σ = 0.56, skewness = 2.35, Q_{50} = 1.52 and mode = 1.23.
(b) Observed elipticity distribution with μ = 1.67, σ = 0.42, skewness = 1.07, Q_{50} = 1.59 and mode = 1.92. 8-38
Figure 8.64 : BUB9.CFI - original image. 8-39
Figure 8.65 : Manually segmented image. 8-40
Figure 8.66 : Automatically segmented image, using the optimal DSSE of radius 8 pixels. 8-40
Figure 8.67 : Segmentation efficiency curve. 8-41
Figure 8.68 : Three point averaged optimal DSSE curve. 8-41
Figure 8.69 : (a) Expected size distribution with μ = 217.71, σ = 321.69, skewness = 5.03, Q_{50} = 123.50 and mode = 51.00.
(b) Observed size distribution with μ = 215.22, σ = 214.47, skewness = 3.29, Q_{50} = 150.00 and mode = 54.00. 8-42

Figure 8.70 : (a) Log normal fit to the expected size distribution, with the best fit parameters $\mu = 202.30$ and $\sigma = 233.81$.
(b) Log normal fit to the observed size distribution, with the best fit parameters $\mu = 213.83$ and $\sigma = 209.70$. 8-42
Figure 8.71 : (a) Expected circularity distribution with $\mu = 2.02$, $\sigma = 0.35$, skewness = 1.85, $Q_{50} = 1.94$ and mode = 1.74.
(b) Observed circularity distribution with $\mu = 2.09$, $\sigma = 0.68$, skewness = 4.39, $Q_{50} = 1.89$ and mode = 1.99. 8-43
Figure 8.72 : (a) Expected elipticity distribution with $\mu = 1.52$, $\sigma = 0.41$, skewness = 1.94, $Q_{50} = 1.41$ and mode = 1.33.
(b) Observed elipticity distribution with $\mu = 1.53$, $\sigma = 0.43$, skewness = 4.32, $Q_{50} = 1.44$ and mode = 1.30. 8-43
Figure 8.73 : BUB10.CFI - original image. 8-44
Figure 8.74 : Manually segmented image. 8-45
Figure 8.75 : Automatically segmented image, using the optimal DSSE of radius 11 pixels. 8-45
Figure 8.76 : Segmentation efficiency curve. 8-46
Figure 8.77 : Three point averaged optimal DSSE curve. 8-46
Figure 8.78 : (a) Expected size distribution with $\mu = 408.85$, $\sigma = 1092.94$, skewness = 8.38, $Q_{50} = 197.00$ and mode = 192.00.
(b) Observed size distribution with $\mu = 398.99$, $\sigma = 545.99$, skewness = 4.87, $Q_{50} = 269.00$ and mode = 183.00. 8-47
Figure 8.79 : (a) Log normal fit to the expected size distribution, with the best fit parameters $\mu = 316.42$ and $\sigma = 340.18$.
(b) Log normal fit to the observed size distribution, with the best fit parameters $\mu = 388.83$ and $\sigma = 458.81$. 8-47
Figure 8.80 : (a) Expected circularity distribution with $\mu = 2.10$, $\sigma = 0.34$, skewness = 1.25, $Q_{50} = 2.03$ and mode = 2.10.
(b) Observed circularity distribution with $\mu = 2.31$, $\sigma = 0.99$, skewness = 4.25, $Q_{50} = 2.02$ and mode = 1.96. 8-48
Figure 8.81 : (a) Expected elipticity distribution with $\mu = 1.55$, $\sigma = 0.38$, skewness = 1.30, $Q_{50} = 1.47$ and mode = 1.36.
(b) Observed elipticity distribution with $\mu = 1.64$, $\sigma = 0.47$, skewness = 2.45, $Q_{50} = 1.55$ and mode = 1.31. 8-48
Figure 8.82 : BUB11.CFI - original image. 8-49
Figure 8.83 : Manually segmented image. 8-50
Figure 8.84 : Automatically segmented image, using the optimal DSSE of radius 8 pixels. 8-50
Figure 8.85 : Segmentation efficiency curve. 8-51
Figure 8.86 : Three point averaged optimal DSSE curve. 8-51
Figure 8.87 : (a) Expected size distribution with $\mu = 244.43$, $\sigma = 377.12$, skewness = 6.56, $Q_{50} = 147.00$ and mode = 92.00.
(b) Observed size distribution with $\mu = 241.14$, $\sigma = 283.81$, skewness = 4.91, $Q_{50} = 164.00$ and mode = 69.00. 8-52
Figure 8.88 : (a) Log normal fit to the expected size distribution, with the best fit parameters $\mu = 225.04$ and $\sigma = 242.32$.
(b) Log normal fit to the observed size distribution, with the best fit parameters $\mu = 241.04$ and $\sigma = 275.69$. 8-52
Figure 8.89 : (a) Expected circularity distribution with $\mu = 2.03$, $\sigma = 0.31$, skewness = 1.26, $Q_{50} = 1.97$ and mode = 1.94.
(b) Observed circularity distribution with $\mu = 2.16$, $\sigma = 0.78$, skewness = 4.53, $Q_{50} = 1.97$ and mode = 1.94. 8-53
Figure 8.90 : (a) Expected elipticity distribution with $\mu = 1.54$, $\sigma = 0.40$, skewness = 1.71, $Q_{50} = 1.46$ and mode = 1.30.
(b) Observed elipticity distribution with $\mu = 1.60$, $\sigma = 0.45$, skewness = 2.99, $Q_{50} = 1.51$ and mode = 1.29. 8-53

Figure 8.91 : BUB12.CFI - original image. 8-54
Figure 8.92 : Manually segmented image. 8-55
Figure 8.93 : Automatically segmented image, using the optimal DSSE of radius 10 pixels. 8-55
Figure 8.94 : Segmentation efficiency curve. 8-56
Figure 8.95 : Three point averaged optimal DSSE curve. 8-56
Figure 8.96 : (a) Expected size distribution with μ = 284.44, σ = 399.21, skewness = 5.03, Q_{50} = 177.00 and mode = 144.00.
(b) Observed size distribution with μ = 280.36, σ = 250.20, skewness = 2.54, Q_{50} = 210.00 and mode = 132.00. 8-57
Figure 8.97 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 266.26 and σ = 27.24.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 284.03 and σ = 274.65. 8-57
Figure 8.98 : (a) Expected circularity distribution with μ = 2.06, σ = 0.32, skewness = 1.48, Q_{50} = 1.98 and mode = 1.87.
(b) Observed circularity distribution with μ = 2.19, σ = 0.73, skewness = 3.15, Q_{50} = 1.96 and mode = 1.87. 8-58
Figure 8.99 : (a) Expected elipticity distribution with μ = 1.55, σ = 0.42, skewness = 2.09, Q_{50} = 1.45 and mode = 1.34.
(b) Observed elipticity distribution with μ = 1.61, σ = 0.47, skewness = 3.88, Q_{50} = 1.51 and mode = 1.25. 8-58
Figure 8.100 : Scattergram of the expected number of bubbles (N_E) against the observed number of bubbles (N_O). 8-61
Figure 8.101 : Scattergram of the expected mean bubble size against the observed mean bubble size. 8-61
Figure 8.102 : 3-sigma control chart for the measurement x if μ_x and σ_x are known. 8-65
Figure 8.103 : A 3-sigma control scattergram. 8-67
Figure 8.104 : 3-sigma control scattergram using μ_s and μ_e measurements for the manually segmented images. 8-69
Figure 8.105 : 3-sigma control scattergram using μ_s and μ_e measurements for the automatically segmented images. 8-70

Figure 9.1 : A block diagram illustrating the possible application of the work presented in this dissertation, to a machine vision system that is capable of characterising the froth structures that are prevalent on the surface of industrial flotation cells. 9-3

List of Tables

Table 2.1 : Existing flotation cell instruments and their measurement and control objectives. 2-21
Table 2.2 : Feature measurements made by Woodburn, Stockton and Robbins. 2-27

Table 3.1 : MVP-AT frame buffer memory configurations. 3-9

Table 6.1 : Bubble detection as a function of SE radius, with calculated efficiency and optimal SE values. 6-33

Table 7.1 : Primary object parameters for characterising the blobs of segmented surface froth images. 7-7
Table 7.2 : Processed blob. 7-10
Table 7.3 : Chosen histogram parameters for BUB1.CFI. 7-19
Table 7.4 : χ^2 test for the size distributions. 7-25
Table 7.5 : χ^2 test for the circularity distributions. 7-26
Table 7.6 : χ^2 test for the elipticity distributions. 7-26
Table 7.7 : χ^2 test for the log normal fit to the expected size distribution. 7-28
Table 7.8 : χ^2 test for the log normal fit to the observed size distribution. 7-29

Table 8.1 : Example table for displaying the χ^2 tests. 8-3
Table 8.2 : χ^2 tests for Figures 8.6, 8.7 (a) and (b). 8-7
Table 8.3 : χ^2 tests for Figures 8.8 and 8.9. 8-8
Table 8.4 : χ^2 tests for Figures 8.15, 8.16 (a) and (b). 8-12
Table 8.5 : χ^2 tests for Figures 8.17 and 8.18. 8-13
Table 8.6 : χ^2 tests for Figures 8.24, 8.25 (a) and (b). 8-17
Table 8.7 : χ^2 tests for Figures 8.26 and 8.27. 8-18
Table 8.8 : χ^2 tests for Figures 8.33, 8.34 (a) and (b). 8-22
Table 8.9 : χ^2 tests for Figures 8.35 and 8.36. 8-23
Table 8.10 : χ^2 tests for Figures 8.42, 8.43 (a) and (b). 8-27
Table 8.11 : χ^2 tests for Figures 8.44 and 8.45. 8-28
Table 8.12 : χ^2 tests for Figures 8.51, 8.52 (a) and (b). 8-32
Table 8.13 : χ^2 tests for Figures 8.53 and 8.54. 8-33
Table 8.14 : χ^2 tests for Figures 8.60, 8.61 (a) and (b). 8-37
Table 8.15 : χ^2 tests for Figures 8.62 and 8.63. 8-38
Table 8.16 : χ^2 tests for Figures 8.69, 8.70 (a) and (b). 8-42
Table 8.17 : χ^2 tests for Figures 8.71 and 8.72. 8-43
Table 8.18 : χ^2 tests for Figures 8.78, 8.79 (a) and (b). 8-47
Table 8.19 : χ^2 tests for Figures 8.80 and 8.81. 8-48
Table 8.20 : χ^2 tests for Figures 8.87, 8.88 (a) and (b). 8-52
Table 8.21 : χ^2 tests for Figures 8.89 and 8.90. 8-53
Table 8.22 : χ^2 tests for Figures 8.96, 8.97 (a) and (b). 8-57
Table 8.23 : χ^2 tests for Figures 8.98 and 8.99. 8-58
Table 8.24 : Summarised manually and automatically determined mean bubble sizes, with the corresponding number of detected bubbles. 8-60
Table 8.25 : Manually determined μ_s and μ_e measurements, used to construct Figure 8.104. 8-69
Table 8.26 : Automatically determined μ_s and μ_e measurements, used to construct Figure 8.105. 8-70

Glossary

Area of interest (AOI)	This is a delineated region of an image, which is considered potentially interesting.
Charge-coupled device (CCD)	This is a photosensitive device that consists of metal-oxide semiconductor (MOS) capacitors. The capacitors produce an analogue voltage that is proportional to the amount of light falling on them.
Connected component operator	An operator that analyses binary images and labels the connected members within the image.
Discrete spherical structuring element (DSSE)	This is a quantized (digital) spherical structuring element. See structuring element.
Edge detector	An operator that is used to identify edges in digital images.
Entrainment	A process by which ultrafine particles are transported to the froth surface in flotation.
Flotation	A physical separation technique that is used to separate minerals from their ores. The technique uses the differences in surface characteristics to separate the required mineral.
Grade	Grade is a measure of the quality of the concentrate obtained from a separation process, such as flotation.
Machine vision	The application of digital image processing to the solution of practical vision problems. Machine vision systems perform such tasks as object location, object inspection, object counting and motion estimation.
Mathematical morphology	A field in image processing that uses set theory to extract shape information from digital images.

Mineral processing	A field of chemical engineering that is concerned with the extraction and concentration of minerals, from their ores.
Optimum mesh of grind	This is the particle size at which the most economic recovery of a mineral can be obtained from its processed ore.
Reagent	A general term that refers to the chemical additives used to improve the performance and selectivity of the flotation process.
Recovery	This is a measure of how effectively a separation process, such as flotation, extracts the required mineral from the liberated ore.
Residue edge detection	A method of edge detection using morphological operations, such as dilation, erosion, closing and opening.
Segmentation	The task of subdividing an image into meaningful regions, that can be processed at a later stage.
Shape-from-shading	A technique that uses the variation in surface brightness of an image, to extract shape information.
Structuring element (SE)	This is a function defined on the domain of the spatial pattern of the morphologic operator and whose value at each pixel of the domain, is the weight or coefficient employed by the morphologic operator at that pixel position. The SE plays a role analogous to that of the kernel in a convolution operation.
Structured lighting	A technique of projecting a carefully designed light pattern onto a scene and then viewing the scene from a different direction, in order to extract depth and surface orientation information.
Umbra	This is the set of all points on or below the surface of a grey level image.

$b(x,y)$	Denotes a binary image
$f(x,y)$	Denotes a grey level image
T	Denotes a particular threshold level
∇	Grad operator
∇^2	Laplacian operator
$\in$	Element of
$\cup$	Set union
$\cap$	Set intersection
$\forall$	For all
$\oplus$	Denotes morphological dilation
$\ominus$	Denotes morphological erosion
$\bullet$	Denotes morphological closing
$\circ$	Denotes morphological opening

CHAPTER 1

INTRODUCTION

Of man's five senses (vision, hearing, taste, smell and touch), the sense of vision is the most powerful and the most complicated. It allows us as human beings to interact intelligently with our environment, without direct physical contact. From it we gather a considerable amount of information on our surroundings such as the position, orientation and identity of objects and the relationships between them. It is not surprising therefore that researchers in the field of digital image processing are interested and actively engaged in developing machines that can "see".

Our limited knowledge on biological vision processes and in particular on the human visual process, shows them to be exceptionally complex mechanisms and difficult to model. Thus the efficiency of machine vision systems has been seriously limited by this complexity and the inability to accurately model the systems being mimicked. As a result, many attempts to provide machines with a sense of vision have mostly ended in failure.

However, with the recent technological advances made in computer architecture considerable progress has been made in machine vision systems, particularly for industrial applications. The reasons for this are that industrial machine vision systems work within well controlled visual environments and the machines have clear-cut tasks, such as the recognition and removal of defective parts moving along a conveyor belt. At present though, a "universal"

machine vision system, that is one that is capable of working in any environment, cannot be built using existing technology. Instead, machine vision systems are designed to work in specifically controlled environments, performing well defined tasks.

This dissertation presents work into the investigation and design of a machine vision system that is capable of characterising the froth structures that are prevalent on the surface of industrial flotation cells. The size and shape of the bubbles that constitute the surface froth convey considerable visual information on the state of the flotation process. The ability to extract this information using a machine vision system, is therefore of great interest. Correlating bubble size and shape measurements with other flotation parameters such as pH level, aeration rate, reagent feed and froth height, would greatly enhance the current understanding of the flotation process. The measurements could also be used in the optimization and control of flotation cells; thereby actively reducing operating costs and improving overall plant performance.

1.1 FORMAT OF THIS DISSERTATION

The format of this dissertation is as follows :

Chapter 2 : Fundamental insight and understanding into the application of vision based analysis of the surface froth structures in industrial flotation cells is provided. A general overview of mineral processing is presented to introduce important processing stages and technical terminology. With this background a more detailed study is given on flotation. Here the fundamentals of flotation are presented, which include the physical and chemical factors that influence cell performance. The parameters that actively affect cell performance are discussed and an explanation of how froth structure is indicative of cell performance is given. The motivation behind the use of a vision based system is given together with a discussion on the contribution of this particular work, to previous work.

Chapter 3 : A fundamental look at the components that constitute a machine vision system is given, together with the individual functions they perform in an overall system. The equipment configuration that was available in assisting in this research work is also discussed.

Chapter 4 : The visually identifiable features that can be used to characterise surface froth structures are discussed, along with the approach to the extraction (segmentation) of these features and the segmentation problems encountered.

Chapter 5 : The investigation of the application of standard or classical image processing techniques to the segmentation of surface froth images is presented. The chapter illustrates both qualitatively and quantitatively the complexity of the problem and the inability of classical techniques to adequately segment surface froth images.

Chapter 6 : An overview of morphological image processing is given and the morphological technique that reliably and accurately segments surface froth images is discussed in detail. A thorough description of the full processing technique, that is the preprocessing, segmentation and postprocessing of the images is given, together with images that illustrate the individual processing stages. The measurement of the efficiency of the technique is presented, with a methodology for selecting the best possible filter size for optimally segmenting surface froth images.

Chapter 7 : This chapter discusses the analysis of segmented surface froth images, together with the statistical characterisation of the images.

Chapter 8 : The results of processing different surface froth images, using the techniques developed in Chapters 6 and 7, are presented. The results are interpreted and a control concept, that is capable of recognizing different froth structures using the automatically determined bubble size and shape measurements, is discussed.

Chapter 9 : An overview of the work is presented and relevant conclusions are drawn. Suggestions for system improvement and for further areas of research are given.

CHAPTER 2

MINERAL EXTRACTION AND CONCENTRATION BY FROTH FLOTATION

"A new metallurgical process never springs fully developed from the brain of one person, but is the result of patient investigation, application and improvement by many minds, during many years."

T.J. Hoover, Concentrating Ores by Flotation, Mining Magazine, London, 1912.

As a consequence of the Industrial Revolution, mineral extraction and concentration by froth flotation became a workable process in the late 19th and early 20th centuries. The rapid increase in demand for raw materials put pressure on existing supplies and provided incentive to find new techniques for the processing of low-grade ores.

It is not possible to credit a particular individual with the "invention" of flotation (Ives, 1984) and through the years numerous people have made significant contributions to the understanding and implementation of the process.

Today, froth flotation has grown into the most widely used, most versatile and most important mineral processing technique in operation in industrialised countries; with most of the world's base metals such as lead, copper and zinc, currently being extracted and concentrated this way.

2.1 MINERAL PROCESSING

By definition minerals are natural inorganic substances possessing definite chemical compositions and atomic structures. Most minerals do not exist as free elements in the earth's crust, but occur as compounds in ore deposits. An ore is a mixture of extractable minerals and extraneous rocky (waste) material known as gangue, and can be briefly described as the accumulation of mineral in sufficient quantity, such that the extraction of the mineral is economically viable. Ores are frequently classified according to the nature of their mineral content.

- Native ores : contain the valuable mineral in its elementary form.
- Sulphide ores : contain the valuable mineral as a sulphide compound.
- Oxidised ores : contain the valuable mineral as an oxide, sulphate, silicate, carbonate or some hydrated form of these.
- Complex ores : contain profitable amounts of more than one valuable mineral.

Mineral processing (Wills, 1981), alias ore dressing, mineral dressing or milling, is concerned with the extraction of the valuable mineral content of an ore. It is a physical separation process that consists of two important operations; liberation and separation, as illustrated in Figure 2.1.

The liberation or release of minerals is achieved by comminution; the reduction in size of the mined ore brought about by crushing and if necessary grinding. The objective of liberation is to reduce the ore to a size where the mineral content is just freed from the gangue. Overgrinding of the ore is a wasteful procedure. The extra energy used is a costly loss and efficient separation of the finely ground particles cannot be achieved, which means a substantial amount of the mineral is usually lost at the end of the separation process. Thus the degree of liberation is the key to success in mineral processing operations and ideally, complete liberation is a prerequisite for the optimal separation of a mineral. In practice the complete liberation of a mineral may not be possible or practical and is seldom achieved, with ores usually being ground to an optimum mesh of grind. This is the particle size at which the most economic recovery of the mineral can be obtained, and is determined from laboratory and pilot scale data.

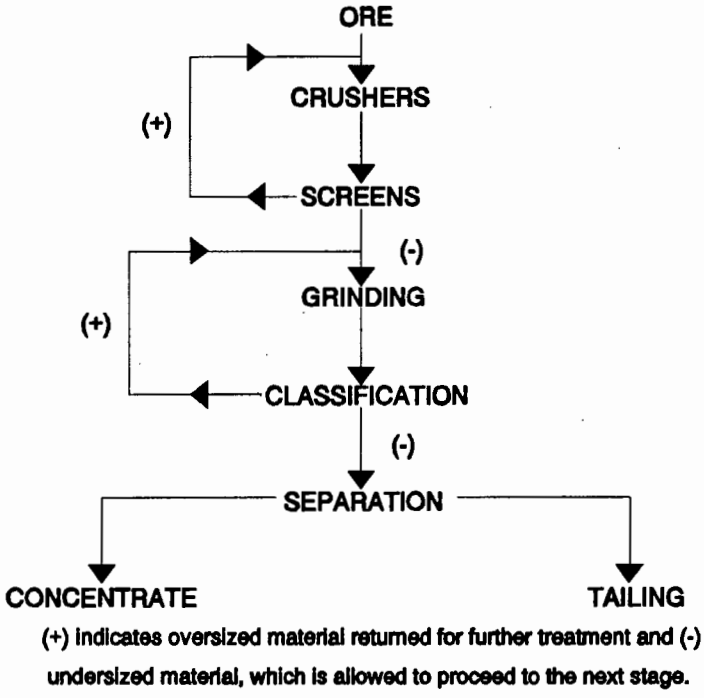


Figure 2.1 : Line flow sheet of plant operations involving liberation and separation.

The resultant liberated ore consists of free mineral particles, middling particles and gangue. Middlings, are particles of "locked" mineral and gangue and release of the mineral content can only be achieved by further comminution.

After the mineral has undergone suitable liberation, it goes through a process of separation or concentration. The objective of separation, as illustrated in Figure 2.2, is to extract the liberated mineral particles from the gangue material, producing a concentrate and tailings (waste).

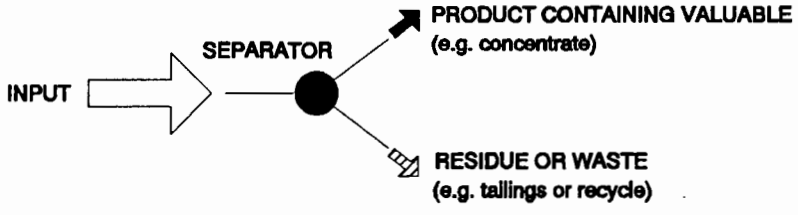


Figure 2.2 : Separator objectives.

Fundamentally, separation is brought about by suspending the liberated mineral in a liquid medium and passing the mixture through an appropriate piece of equipment termed a separator. In the separator, Figure 2.3, a suitable force is applied to the suspended particles.

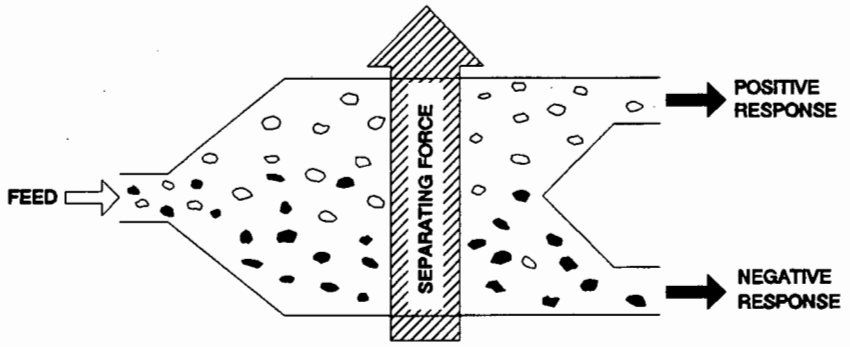


Figure 2.3 : Basic principle of a separator.

As these particles have different properties, depending on their composition or size, the applied force affects them to varying degrees. Those that are strongly affected by the force have a positive response; those that are unaffected have a negative response.

In industrial concentrators, the separating force used utilises some specific difference in physical or chemical properties between the valuable mineral and the gangue. Existing mineral processing techniques are mainly concerned with physical separation methods using differences in :

- optical and radioactive properties
- specific gravities
- the surface characteristics of particles
- magnetic properties
- electrical conductivity

The separation method used, depends on the nature of the ore and the properties (as well as their differences) of the minerals to be separated. Since particle size affects the efficiency of mineral separation and concentration, the size at which liberation occurs may influence the

method of separation used, as illustrated in Figure 2.4. If more than one method exists, the most economical method is chosen.

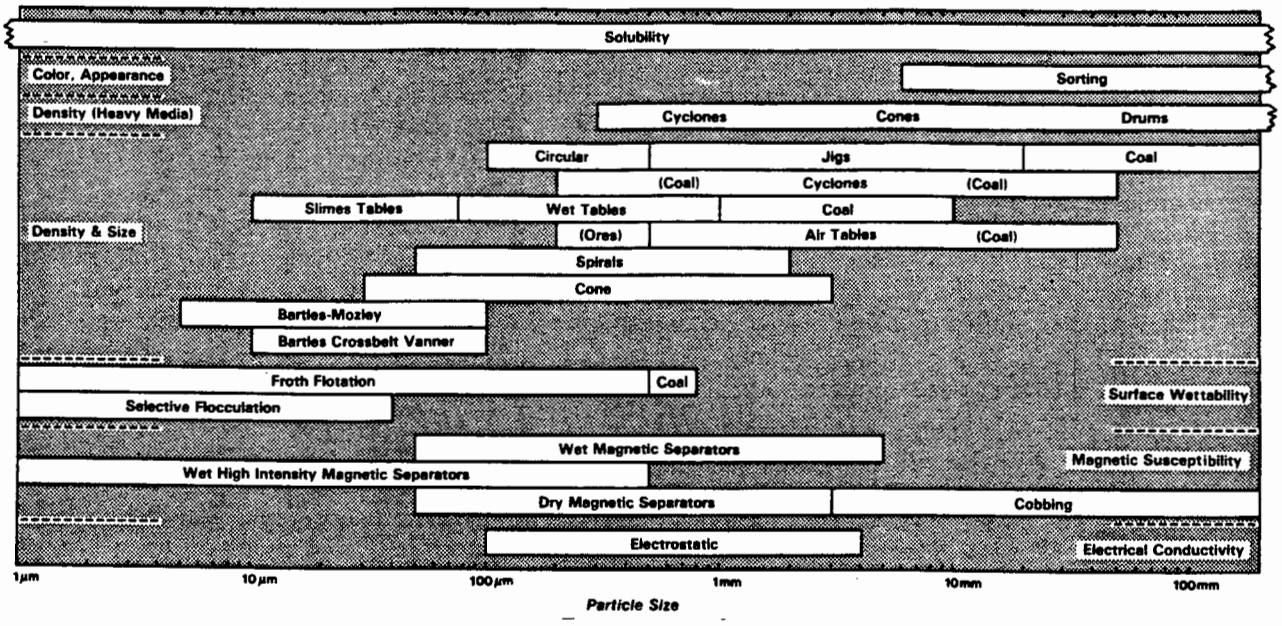


Figure 2.4 : Particle size ranges for concentrating equipment.

Ideal separation where the valuable mineral content is completely recovered in the concentrate and the gangue or waste material goes into the tailings, is never achieved in real operating systems. Thus as a means of gauging the performance of a separation process, two parameters are commonly considered; the grade (assay) and the recovery (Kelly and Spottiswood, 1982).

Grade is a measure of the quality of the concentrate of a separation process. A suitable definition is given by :

$$\text{grade (\%)} = \frac{\text{mass of valuable in stream}}{\text{mass of valuable and waste in stream}} \times 100$$

Recovery is a measure of how effectively the separation process has extracted the valuable mineral content from the liberated ore. A suitable definition is given by :

$$\text{recovery (\%)} = \frac{\text{percentage (or mass) of valuable in product stream}}{\text{percentage (or mass) of valuable in input stream}} \times 100$$

For all separation processes there is an approximately inverse relationship between the grade and the recovery of the concentrate and as such there is always a trade off between the two parameters. If an attempt is made to attain a very high-grade of concentrate, the overall recovery will be low and much of the mineral will be lost in the tailings. Alternatively if a high-recovery of mineral is aimed for, there will be more gangue in the concentrate resulting in a low-grade concentrate. As a result of this inverse relationship, there is always an economic optimum in the combination of grade and recovery, for the concentration of a mineral. The cumulative grade versus cumulative recovery curve, or simply the grade/recovery curve for a separation process, Figure 2.5, is a graphical representation that is frequently used for computing the process operating point.

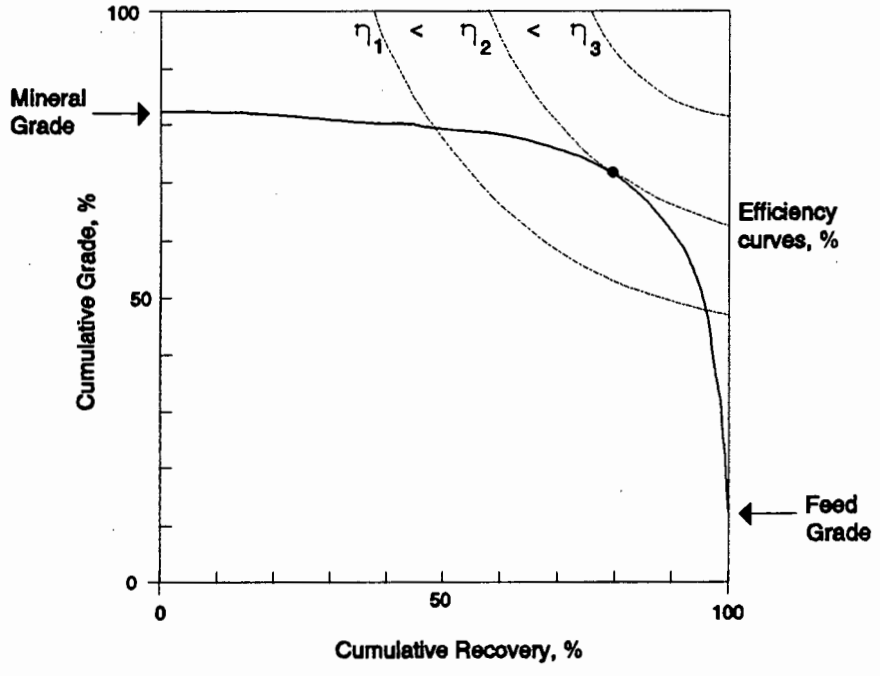


Figure 2.5 : The grade/recovery curve for a separation process.

If the efficiency (η) of the separation process is expressed as a single value and plotted on the graph, it will occur as a series of isobars, as shown (Kelly and Spottiswood, 1982). Intuitively the optimal operating point, marked with a • on the diagram, is that point where the efficiency curve (η_2) just touches the knee of the grade/recovery curve. This however is not necessarily true since the grade/recovery of the process at this point may not be economically

acceptable. For example, it may be more profitable to extract the mineral at a lower grade, but with a higher recovery; this means that the chosen operating point would sit below the point illustrated.

Once the grade/recovery curve for a particular separation method has been established, either from plant data or experimentally obtained data, and the economically optimal operating point has been determined; it is necessary to maintain this operating point in order to achieve the required overall plant performance. In practice, suitably qualified plant workers are charged with the responsibility of monitoring and maintaining the required operating point. However, with the recent technological advances made in cost-effective computer vision systems researchers involved in plant optimization and control have started to investigate the use of on-line vision systems, as a means of fully automating the control of some mineral processing stages. The use of on-line analysis systems for real-time, automated control of certain mineral separation processes, will not only ensure optimal performance in terms of mineral extraction but will also provide further insight and understanding into the separation method. Examples of this current trend towards the use of machine vision systems for automated analysis and control are :

- rock fragmentation analysis (Wigeson, 1987)
- automated control of the separator blade in a spiral ore concentrator (Gold, 1991)
- the recognition of an optimal froth height in the beneficiation of low-rank coals (Woodburn, Stockton and Robbins, 1989).

2.2 FLOTATION FUNDAMENTALS

Froth flotation is a primary separation technique based on the differences between the surface characteristics of mixed, liberated mineral particles. The major difference exploited by flotation is the wettability of a particle; that is whether the particle is hydrophillic and in general will remain in water or whether it is hydrophobic, and therefore shows an affinity for attachment to, and removal by, air bubbles.

2.2.1 THE FLOTATION CELL

The flotation cell, Figure 2.6, is the mechanical machine in which flotation occurs. After

comminution and prior to flotation, the liberated ore is mixed with water and chemical reagents to form a slurry. This pre-processing stage is known as conditioning, and the aim is to render the desirable component in the feed hydrophobic. The feed slurry (pulp) is then pumped into a flotation cell, which can be crudely described as a box with a stirrer. Essentially the cell is a mechanically agitated bath which uses a rotating impeller to thoroughly mix the pulp with the incoming air. The impeller breaks up the air supply resulting in fine air bubbles that disperse throughout the agitated suspension.

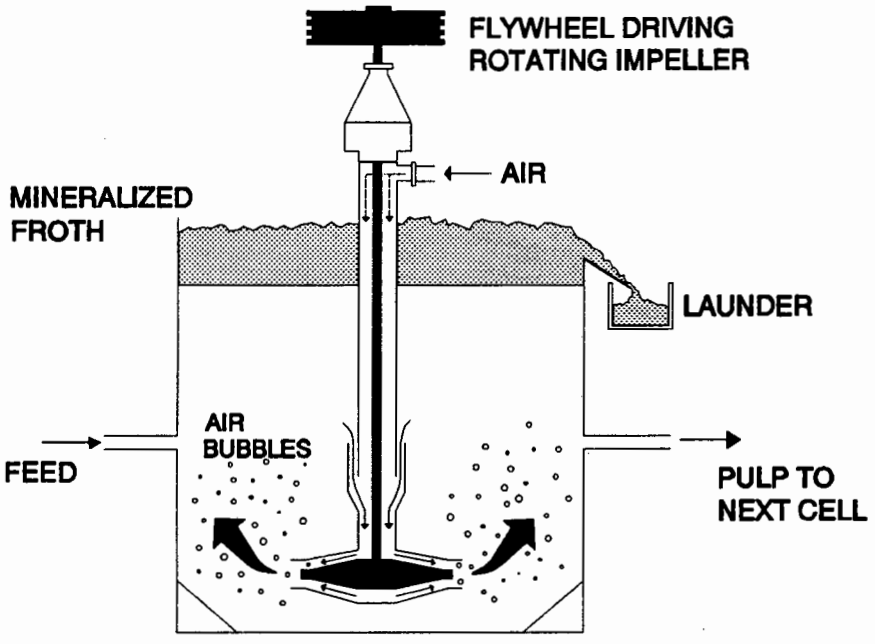


Figure 2.6 : A simple dispersed air flotation machine.

Adhesion and entrainment (Subrahmanyam and Forssberg, 1988), are the two important mechanisms by which collection of the liberated mineral particles takes place in flotation. Together with froth drainage and collapse, Figure 2.7, these mechanisms form the basis of the transport of materials within the flotation cell.

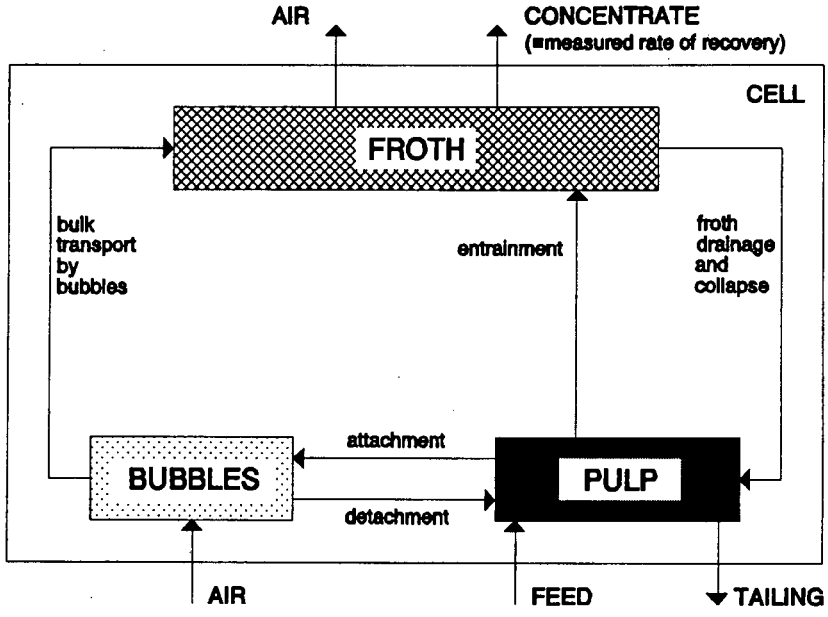


Figure 2.7 : Transport paths of materials in flotation.

Bubble-particle adhesion is the predominant mechanism by which hydrophobic particles are transported to the surface. For adhesion to occur, mineral particles must collide with the upward moving bubbles, Figure 2.8. These collisions result in the rupturing of the bubble

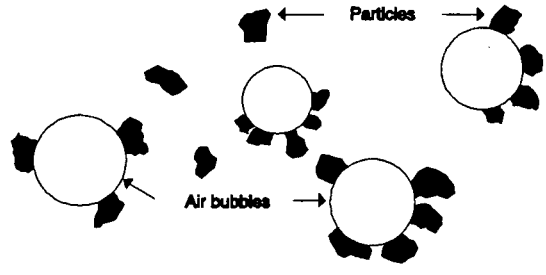


Figure 2.8 : Froth flotation bubbles.

film and the establishment of a three-phase (solid-liquid-gas) interface. As the bubbles rise towards the pulp/froth interface, further collisions result in the bubble surface being covered

with an "armoured" coating of mineral particles.

Entrainment is a characteristic feature of ultrafine particles, that is non-selective resulting in the transportation of both mineral particles and gangue material to the surface. What are ultrafine particles though? For flotation to work, all the particles need to be "fine" and particle sizes are often defined as "coarse", "fine" or "ultrafine". Typically, coarse particles are those of size greater than 150 microns[1], fine particles are those in the size range 25 to 150 microns, and ultrafine or "slime" particles are those of size less than 25 microns. It is these ultrafine particles that are susceptible to entrainment. In entrainment, the ultrafine particles are captured in the liquid lamella of the bubbles and carried to the surface. The thicker the liquid lamella, the higher the water content and hence there is a greater possibility of particles being recovered by entrainment than by flotation. The mechanism is thus closely related to the recovery of water during flotation. If the mineral to be floated is overground, the efficiency of bubble-particle collisions will be very low and the collection of minerals will be predominantly through entrainment. This means the overall selectivity of the flotation process will be poor, resulting in yields of low-grade concentrate.

Once at the surface, the bubbles coalesce to form a mineralized froth which is then bled off. Those particles in the suspension that are hydrophillic, simply remain in suspension where they accumulate to form the tailings of the flotation process and are eventually removed.

2.2.2 THE PHYSICS OF FLOTATION

Bubbles are an essential part of the flotation process. Insight into the underlying physical principles that govern the process of flotation, can be gained from a study of the motion of bubbles in a liquid medium and the interaction of bubbles and suspended mineral particles.

(A) THE HYDRODYNAMICS OF BUBBLES

A bubble is a region in space occupied by a gas and enclosed by a gas-liquid interface (Ives, 1984), with the shape of the bubble being determined by the stress-balances acting on the

[1] A micron (μm) is 10^{-6}m.

interface at equilibrium. When a finite volume of gas is injected into a liquid medium, the gas forms bubbles which rise with a definite shape and velocity; determined by the viscosity and pressure gradient of the liquid, and the surface tension of the gas-liquid interface. As the bubbles rise, the forces acting on them are continually changing which means that their shape, form and velocity are also continually changing. It is this dynamic motion of bubbles through a liquid which is the object of study in bubble hydrodynamics.

Extensive research work has been carried out on the motion of bubbles in a large body of water (Ives, 1984) and the graph of Figure 2.9, representing the cumulative work of many researchers, illustrates the variation of bubble velocity with bubble size, in water.

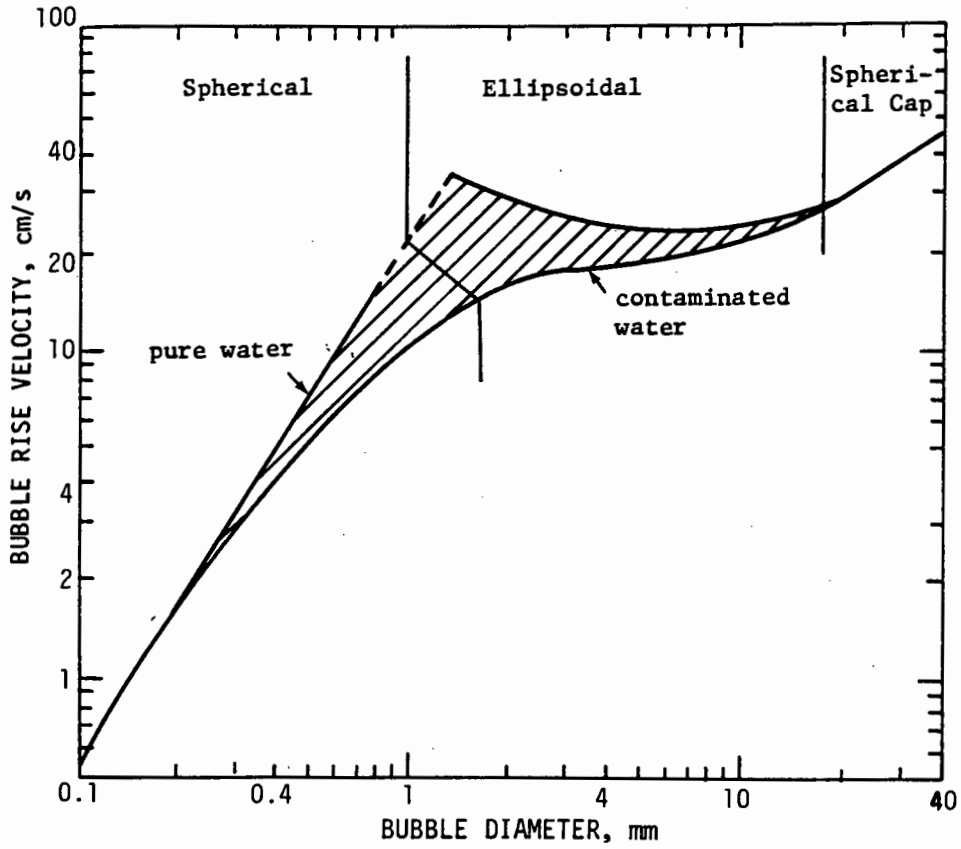


Figure 2.9 : The terminal velocity of air bubbles in water at 20°C.

The graph illustrates three important bubble phases in pure water.

Phase 1 : Small bubbles of diameter < 1mm are approximately spherical in shape and rise in straight lines with stable motion.

Phase 2 : Medium size bubbles of diameter range 1 to 20mm, are ellipsoidal in shape and rise with a rocking, oscillatory or spiral motion indicating instability.

Phase 3 : Large bubbles of diameter > 20mm are greatly deformed, with the head flattened to form a mushroom shape. They rise relatively straight, but are unstable, tending to break into smaller bubbles.

Typically, flotation uses bubbles of size 0.5 to 3mm, and therefore employs phase 1 and the lower end of phase 2. Despite the importance of bubbles in flotation, little attention has been payed to the mechanisms responsible for the generation and dispersion of fine bubbles through the agitated pulp. This is because such evaluation is dependent upon optical measurement techniques, which are of little use in practical cells due to the opaqueness of the agitated pulp.

(B) CONTACT ANGLES AND THREE-PHASE INTERFACES

For flotation to work, bubble-particle adhesion must occur. For bubble-particle adhesion to occur, the bubble must be able to displace water from the surface of the mineral. This will only happen if the mineral is hydrophobic. When bubble-particle adhesion occurs, as discussed in section 2.2.1, an air-water and mineral-water interface is broken down and a three-phase or mineral-water-air interface established. A classical illustration of a three-phase interface involving a smooth ideal mineral surface is given in Figure 2.10.

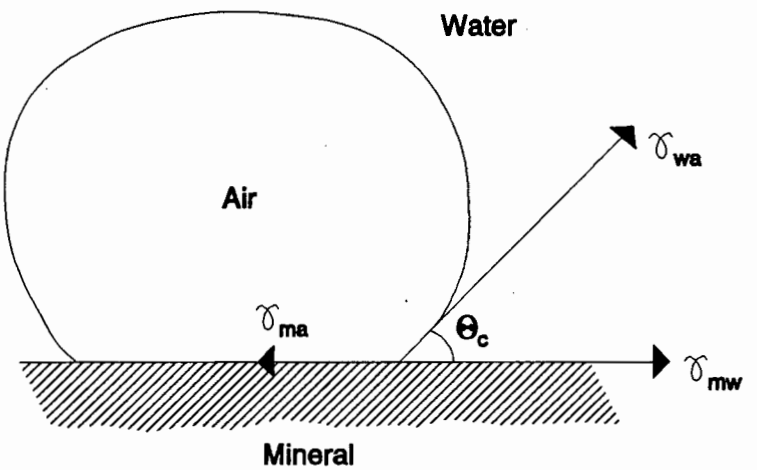


Figure 2.10 : Idealized representation of three-phase equilibrium contact between air, water and mineral surface.

At equilibrium, there exist three interfacial energies γ related by Young's equation :

$$\gamma_{ma} - \gamma_{mw} = \gamma_{wa} \cos \theta_C \quad \ldots(2.1)$$

with θ_C being the contact angle and γ_{wa}, γ_{ma} and γ_{mw} being the interfacial energies at the water-air, mineral-air and mineral-water interfaces, respectively. The contact angle is the angle between the mineral-air interface measured through the liquid phase and indicates the degree of bubble attachment to the mineral surface. Equation (2.1) is an idealized representation of a three-phase interface during bubble-particle adhesion and it is widely accepted that the equation provides an adequate basis for the thermodynamic analysis of three-phase systems (Kelly and Spottiswood, 1982).

When the three-phase interface is established, there is a simultaneous destruction of an air-water and mineral-water interface. Thus for bubble-particle attachment to take place :

$$\gamma_{ma} - \gamma_{mw} < \gamma_{wa} \quad \ldots(2.2)$$

The energy required to break the bubble-particle interface is called the work of adhesion, Γ_{ma}. It is the work required to separate the mineral-air interface and to produce separate air-water and mineral-water interfaces and is given by the expression :

$$\Gamma_{ma} = \gamma_{ma} - (\gamma_{wa} + \gamma_{mw}) \quad \ldots(2.3)$$

For a mineral-air interface to be created and therefore flotation to occur, equation (2.3) shows that Γ_{ma} must be negative. Thus substituting for Young's equation into expression (2.3) gives :

$$\Gamma_{ma} = \gamma_{wa} (\cos \theta_C - 1) \quad \ldots(2.4)$$

From equation (2.4) it can be seen that the more negative Γ_{ma}, the higher the probability of bubble-particle attachment and hence flotation. This equation is often used in the analysis of flotation.

2.2.3 THE CHEMISTRY OF FLOTATION

In section 2.2.2, the underlying physical principles governing the conditions in which flotation

occurs, were introduced. These principles are only applicable to hydrophobic minerals. Most minerals in their natural state are not water-repellant and it is necessary to add chemical reagents to the pulp to assist the flotation process. These chemical reagents, classified as collectors, frothers and regulators, actively enhance the chemical properties of the mineral to be floated.

(A) COLLECTORS

Collectors are the most important of the chemical reagents used. They are organic molecules or ions that selectively absorb on mineral surfaces; rendering the surface hydrophobic so that at equilibrium (as indicated by the contact angle) there is bubble-particle attachment.

Most collectors are weak acids, bases or their salts. They are heteropolar having a non-polar hydrocarbon group that has pronounced water-repellant properties and a polar group, that reacts with water. The polar group is absorbed at the mineral interface either by the chemical reaction of the ions on the mineral surface (chemisorption) or by electrostatic attraction to the mineral surface (physical adsorption). The non-polar organic group, Figure 2.11, is orientated towards the bulk solution, thereby providing a hydrophobic surface to the mineral.

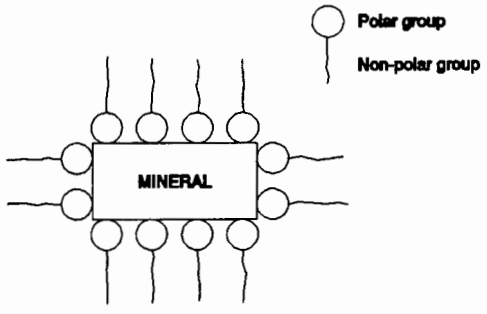


Figure 2.11 : Collector adsorption on mineral surface.

(B) FROTHERS

Frothers are reagents that are used to reduce the size of the bubbles produced in the pulp, and to increase the stability of the bubbles in the froth. After a collector has rendered a mineral surface hydrophobic, the stability of the bubble attachment, especially at the pulp surface, depends to a considerable extent on the efficiency of the frother.

Frothers are water-soluble organic reagents, with heteropolar molecules. The polar group provides water solubility and the non-polar hydrocarbon group, Figure 2.12, is absorbed at the air-water interface.

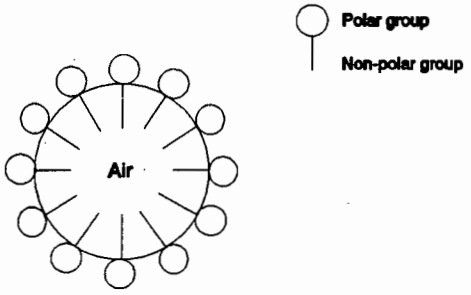


Figure 2.12 : Action of the frother.

In practical flotation plants, as a result of insufficient data on frother performance, the selection of the correct frother for a particular mineral, is mostly based on trial-and-error (Subrahmanyam and Forssberg, 1988).

(C) REGULATORS

Regulators, or modifiers, are used extensively in flotation to modify the action of collectors by actively making the collector more selective to the flotation of a particular mineral and are exceptionally important in the selective separation of a required mineral from a complex ore (section 2.1). This is achieved by enhancing or suppressing the collectors water-repellant effect on a mineral surface. For convenience they are subdivided into activators, depressants, dispersants or alkalinity regulators.

Activators are reagents that assist in the adsorption of the collector to a mineral surface, in the formation of a hydrophobic particle. Consequently they are used to increase the selectivity of flotation.

Depressants are reagents that actively inhibit the absorption of the collector to a mineral surface, thereby rendering the surface hydrophillic and the mineral unfloatable.

Dispersants are often used to prevent flocculation and to ensure that each mineral acts as a discrete entity until separated.

Alkalinity regulators are used to maintain the pH of the process at the desired level, as the pH of the pulp is important, since it can affect the electrical properties of the particles, as

well as the chemical state of the reagents.

2.3 FROTH STRUCTURES IN FLOTATION

The froth structures developed on the surface of flotation cells, provide substantial visual information on cell performance, with respect to the grade and recovery of the mineral being extracted. In industrial flotation plants, experienced operators are able to assess the performance of a flotation cell, by visually inspecting the surface froth structure. This has led to the accumulation of considerable knowledge on the effect of froth structure on grade and recovery at the process operation level, for particular minerals.

2.3.1 TYPES OF FROTH STRUCTURES

Practical flotation cells exhibit one of two predominant froth structures, that are classified as wet or dry.

Wet (shallow) froths have a high water content as a result of low drainage and are characterised by an assembly of small, circularly shaped, fast flowing bubbles. These froths are indicative of cells operating at high recovery.

Dry (deep) froths have a low water content due to the occurrence of drainage within the froth structure. As a result of the reduced water content, the individual bubbles have a degree of stiffness, which causes them to be elongated in the direction of flow. These structures are characterised by an assembly of larger, elliptically shaped, slow moving bubbles, and are indicative of cells operating at high grade.

2.3.2 FACTORS AFFECTING FROTH STRUCTURE

Froth formation is an important phase in the flotation process and is influenced by several physical and chemical factors, that interactively govern the performance of a particular cell.

(A) FROTH HEIGHT

Froth structure varies greatly with froth height. Just above the pulp/froth interface the bubbles

are spherical in shape and closely packed, Figure 2.13. As the height of the froth increases, the bubbles coalesce to form polyhedral shapes and much of the water content drains away, leaving the bubbles less elastic. At a particular height, the liquid lamella reaches a critical value which is related to the particle size, and the bubbles rupture, resulting in a loss of valuable material. This point defines a maximum froth height for a particular set of conditions and is an indication of maximum selectivity at a specified recovery (Engelbrecht and Woodburn, 1975).

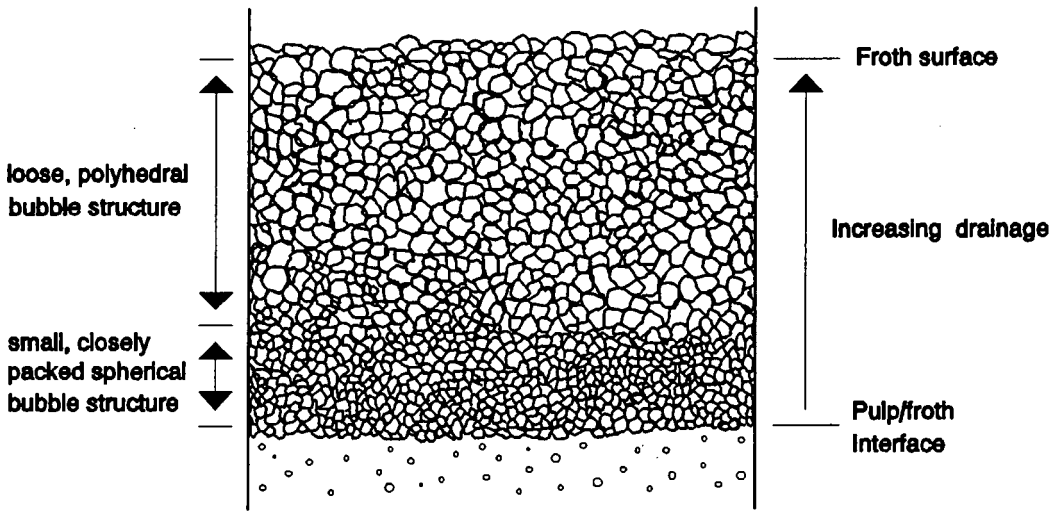


Figure 2.13 : The variation of froth structure with froth height.

(B) FROTH DRAINAGE

Froth drainage is both a function of froth height, as illustrated in Figure 2.13, and the thickness of the liquid lamella of the closely packed bubbles, as illustrated in Figure 2.14.

In shallow froths, the particles entrained in the liquid lamella drain out through the "liquid channels". As the froth deepens though, the excess water is lost through drainage and the liquid lamellas become thinner. This results in the entrained particles being carried to the surface of the froth, thereby influencing the grade of the concentrate.

From the study of froths developed in equilibrium cells (Cutting, Barber and Newton, 1986), two distinct drainage mechanisms have been identified; film drainage and column drainage.

Film drainage is the drainage occurring through the liquid lamella of the bubbles and is prevalent throughout the froth structure.

Column drainage is associated with the regions in the froth where material descends rapidly, at single vertical locations in the froth. This occurs in regions where bubble collapse results in local concentrations that accumulate and penetrate the froth, causing a rapid downward movement of material.

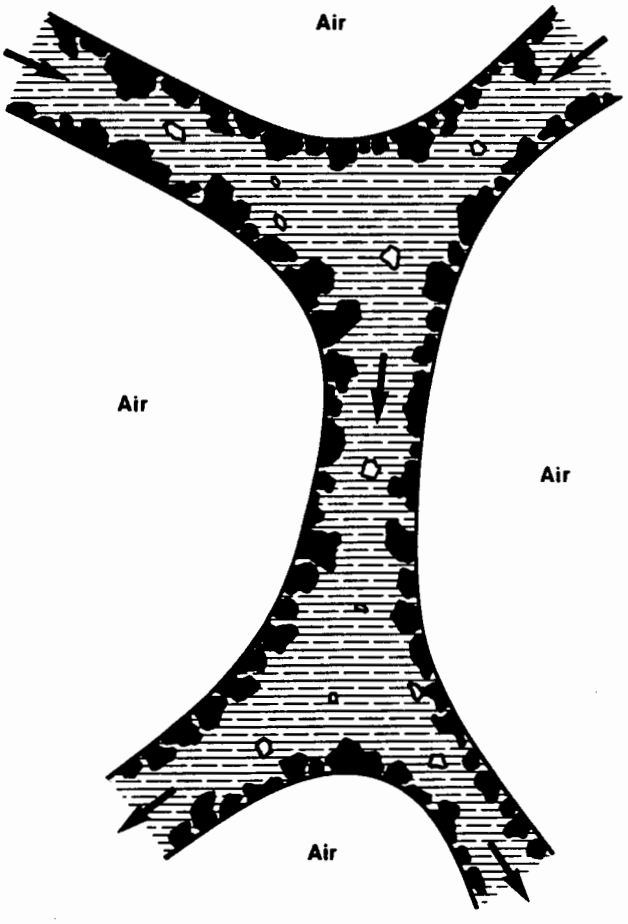


Figure 2.14 : Froth drainage.

(C) FROTH STABILITY

Froth stability is a measure of the persistence of the froth (Subrahmanyam and Forssberg, 1988), and as discussed previously, depends to a considerable extent on the frother used. Unstable froths that collapse rapidly are unsuitable for flotation, so to are overstable froths that have a high degree of persistence, because they are difficult to transport once they have flowed over the cell lip.

(D) FROTH AERATION

Froth aeration refers to the rate at which air is fed into the flotation cell and dispersed by the rotating impeller. The aeration rate affects the quantity of bubbles produced and therefore together with frother concentration indirectly influences the froth height.

2.4 THE ANALYSIS OF FROTH STRUCTURES USING ON-LINE MACHINE VISION

Froth flotation is at present not completely understood and the process is frequently inefficient. This state of affairs can be attributed to a lack of adequate instrumentation on flotation plants, for monitoring those measurable variables that influence cell performance. Greater insight and understanding can therefore be obtained by firstly increasing the instrumentation on existing plants and secondly investigating new measurement techniques.

The application of on-line machine vision analysis to the characterisation of flotation froths is a new measurement technique, and it is necessary to ask two fundamentally important questions.

1) What motivation is there for the use of a machine vision system?
2) How can the results from surface froth characterisation be jointly used with existing measurements to optimize control and thereby improve flotation cell performance?

2.4.1 THE MOTIVATION BEHIND THE USE OF MACHINE VISION

As discussed in section 2.3 it is common operating experience that surface froth structures convey considerable visual information on flotation cell performance. As human beings are capable of assessing cell performance by visually inspecting the surface froth, it follows that it should be possible to automate the inspection using a machine vision system.

Furthermore, the surface froth structures in flotation cells are easily accessible. By viewing the froths from directly above, the complex 3-dimensional surface is reduced to a simpler 2-dimensional surface. Also through the use of correct lighting, individual bubble features that can be used to characterise the froth, can be enhanced. Together these two factors simplify the surface to the extent that the implementation of existing 2-dimensional image processing techniques can be investigated.

Finally, as the surfaces consist of an assembly of variously sized and shaped bubbles, they can be parameterized using statistical descriptions of the resultant size and shape distributions. Thus a machine vision system can produce as output say, a mean size value and shape value

that is indicative of the overall current froth structure of the flotation cell.

2.4.2 THE CONTROL OF FLOTATION CELLS

A number of variables are encountered in flotation, due to variations in the raw material, methods of their preparation, reagentizing and the actual flotation process itself. An understanding of the affect of the variables can help toward proper control of flotation cells, for optimum performance. This insight, as mentioned previously can be obtained by increasing plant instrumentation and investigating new measurement techniques.

The affects of the major physical and chemical variables to flotation have been reviewed in section 2.2. In order to understand how surface froth measurements of bubble size and shape can contribute to flotation cell control, it is necessary to review existing flotation cell measurement techniques.

Currently, the measurement and control of flotation cell parameters has three principle components (Twidle, Engelbrecht and Koel, 1986); field instrumentation, on-stream analysis and a process control (PROSCON) system.

Field instrumentation and on-stream analysis : Table 2.1 lists existing flotation cell instrumentation together with the measurement and control objectives of the instruments and Figure 2.15, illustrates the use of some of the instruments in a flotation circuit control concept.

Proscon System : The proscon system is the heart of the control process. It is usually a mini-computer, that monitors the instruments and implements various control functions to maintain optimum cell performance.

Instrument	Measurement and Control Objectives
mass flowmeters	Used to measure and control the flow of pulp into flotation cells.
pulp-density meters	Used to measure and control the constituency of the in flowing pulp.
pH meters	Used to measure the alkalinity and acidity of the pulp and to control reagent feeders that adjust pH levels.
float sensors	Used to measure the pulp level within the flotation cell. Control is achieved by adjusting a dart valve. Through practical experience, a bubble tube and differential pressure transducer have been found to be the best combination for measuring pulp level. Alternate instrumentation such as submersible floats, capacitance probes and sonic level detectors have been tried but have failed for reasons described by Horst et. al. (Horst and Enochs, 1980).
on-stream X-Ray analyzers	Used to measure the composition of the feed, concentrate and tailing of the cell and to control the addition of chemical reagents to the pulp, to produce the required metallurgical performance.

Table 2.1 : Existing flotation cell instruments and their measurement and control objectives.

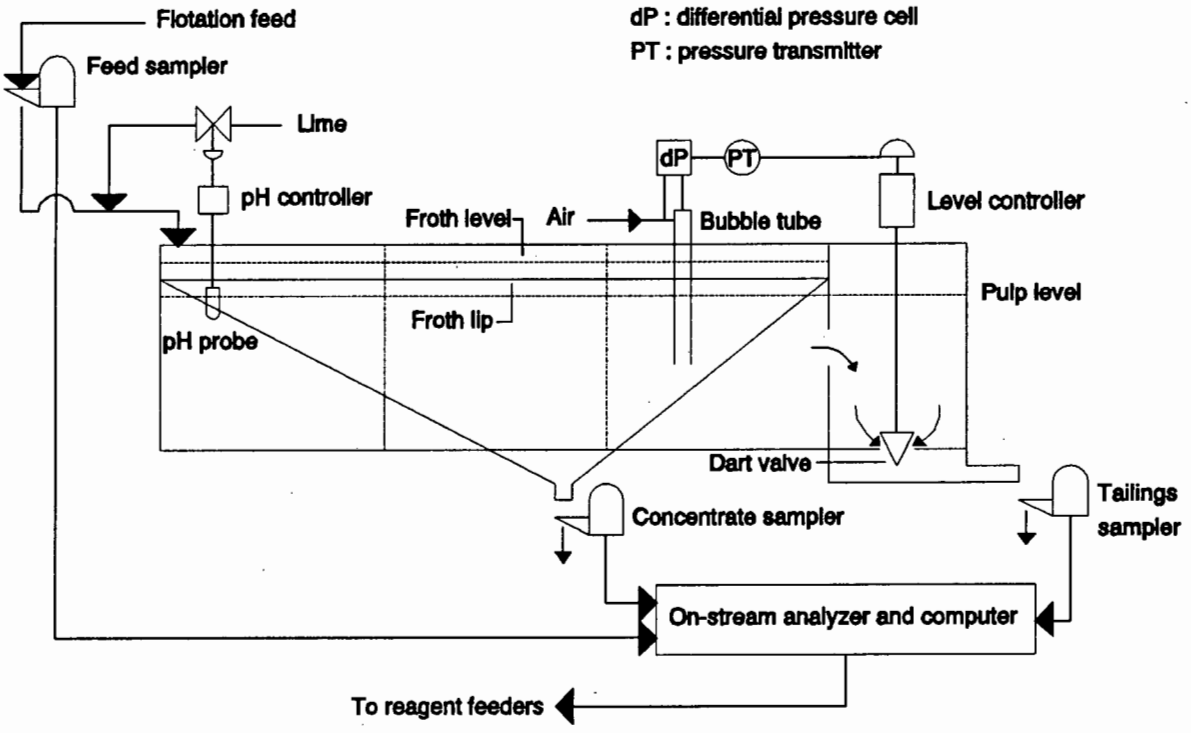


Figure 2.15 : A flotation circuit control concept.

The measurement of bubble size and shape, provides insight into the drainage occurring within the froth structure, as previously discussed in section 2.3 and thus indirectly a measure of the grade/recovery of the cell. Existing instrumentation does not provide such a measure and therefore the ability to measure surface froth structures using an on-line machine vision system, would make a far reaching contribution in the further understanding of flotation processes and the optimization of the control of these processes.

2.5 INDUSTRIAL FLOTATION

An industrial flotation plant, Figure 2.16 (Perry and Green, 1984), is a complex system that processes the mined ore all the way up to the concentrated output. Typically the minerals are concentrated using sequential flotation, whereby the flotation cells are connected in series to form banks and are operated in continuous mode. The cells are housed in a large warehouse structure, where long fluorescent lights often supply the necessary inside illumination and the environment is typically noisy, dusty and wet. The design of a machine vision system is influenced by the environment in which it is intended to operate. It is therefore necessary to look at flotation machines and flotation circuits, to highlight several design considerations for a machine vision system for characterising surface froth structures.

2.5.1 FLOTATION MACHINES

Mineral extraction and concentration by flotation is dependent upon two variables; chemical conditions, as discussed in section 2.2.3 and physical-mechanical conditions, which are determined by the flotation machine characteristics (Kelly and Spottiswood, 1982).

The flotation cell is a well established piece of equipment in the flotation industry. While there exist several variations of the cell, Figure 2.17 (a) and (b), they all perform the same function. The cells act as an agitated bath, using a rotating impeller to disperse fine air bubbles through the mixed pulp. In designing a machine vision system that will sit vertically above one of these cells, there are several practical points to consider :

1) The ambient lighting conditions may be a problem. The severity of the problem though, will depend upon the lighting configuration used to illuminate the surface froth.

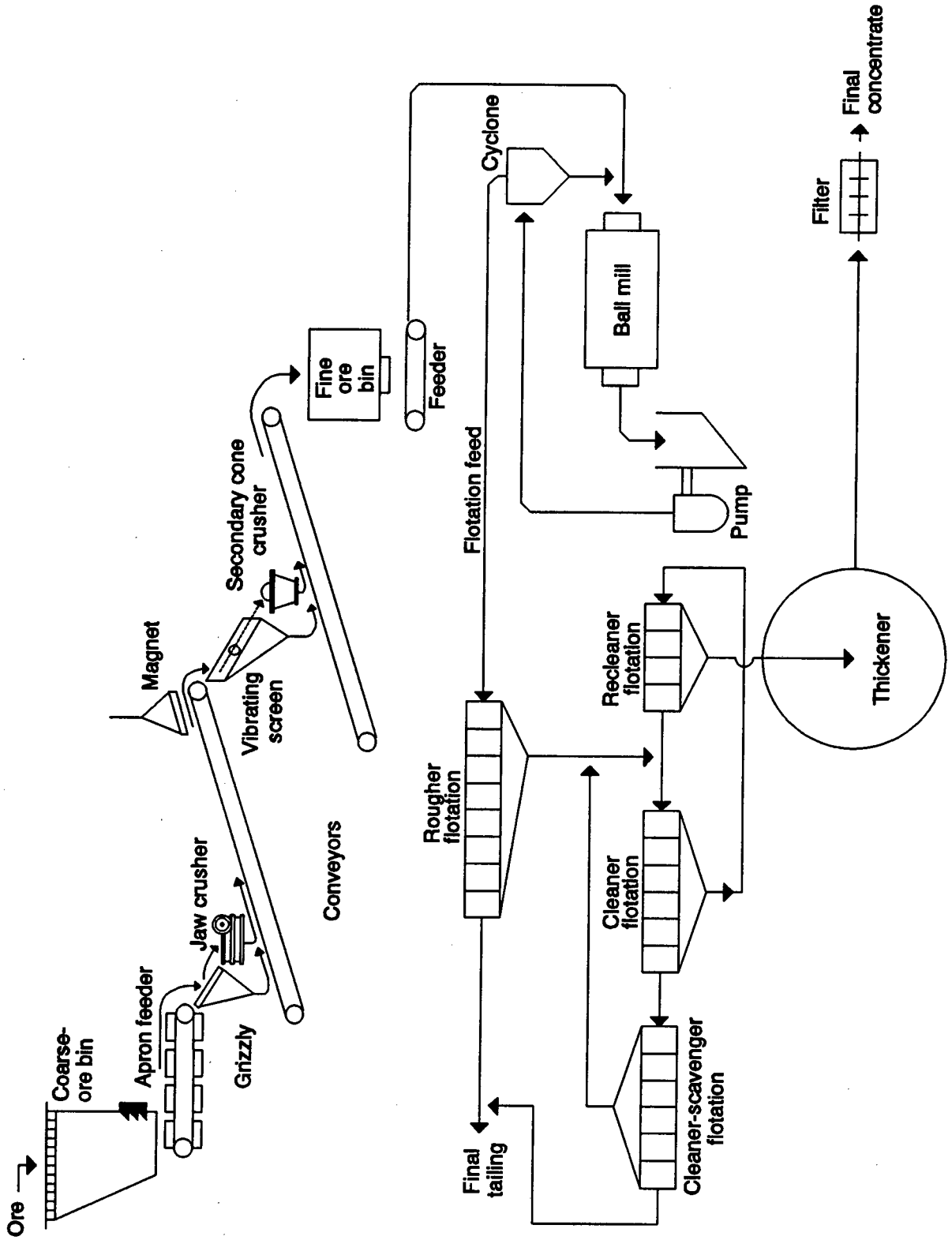


Figure 2.16 : Flow sheet of a simple flotation plant.

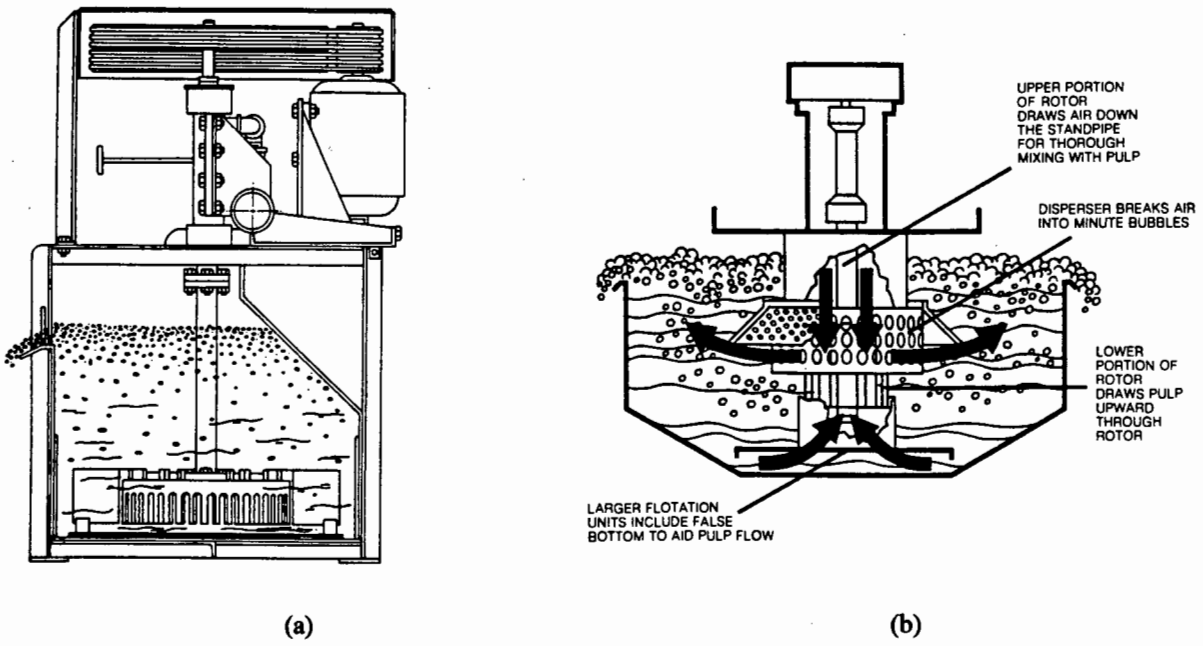


Figure 2.17 : Commonly used flotation machines.
(a) Agitair cell, and (b) Wemco Fagergren cell.

2) The camera and lighting assembly must be placed vertically above the froth structure at a point the maximises the view of the surface froth.
3) Because of the nature of the environment, the camera is liable to get wet and the lens covered in dust. It will therefore be necessary to house the camera in a water and dust tight unit, that has some means of keeping the lens clean. This could be achieved by using a perspex shield in front of the lens, with a wiper system to keep the shield clean. However during operation, the motion of the sweeping wipers is liable to corrupt the images obtained, which is undesirable. Alternatively, as there is an unlimited supply of compressed air, a constant air jet across the camera lens or perspex shield, might suffice in keeping the lens or shield clean. The use of compressed air also has the advantage that it will not inadvertently corrupt the image.

2.5.2 FLOTATION CIRCUITS

As discussed in section 2.1, grade and recovery represent a trade off when extracting and concentrating a mineral. For this reason, the concentration of minerals by froth flotation is performed in specific stages, in order to achieve the required economic output. The specific stages consist of using banks of sequential flotation cells known as roughers, scavengers and

cleaners, and Figure 2.18, illustrates a typical configuration.

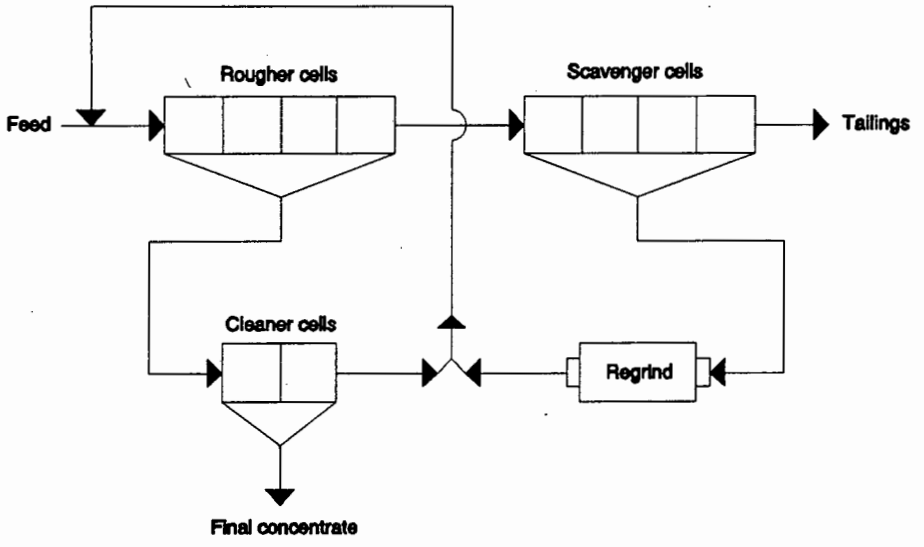


Figure 2.18 : Rougher-scavenger-cleaner system.

Roughing is the first stage, that aims at removing all the easily recoverable valuable mineral content from the pulp in the form of rougher concentrate.

The tailings of the rougher are passed onto the scavenger cells. Here the aim is to recover all of the remaining valuable mineral content that is economically justifiable. The cells are thus operated at high-recovery, low-grade. The scavenger tailings are discarded as they are considered worthless with respect to the mineral being concentrated. The scavenger concentrate on the other hand consists of middlings and is typically reground to liberate the remaining mineral and subsequently returned to the rougher cells.

The rougher concentrate is fed into cleaner cells, which operate at high-grade, low-recovery. Since the increased grade is at the expense of the recovery of the mineral, the cleaner tailings are returned to the rougher cells.

For each bank of cells, there exists a further practical design consideration. Should each cell within a particular bank of cells, be monitored by an individual machine vision system or will it suffice to monitor a bank of cells using a single moveable system located on tracks above the surface froth? The answer to this question, ultimately depends on the performance of the overall machine vision system for an individual cell, in terms of cost and processing time.

2.6 THE CONTRIBUTION OF THIS DISSERTATION

The work presented in this dissertation documents the use of digital image processing in the analysis and characterisation of the surface froth structures in flotation cells; leading to the development of a practical on-line machine vision system capable of monitoring the performance of flotation cells and assisting in the automated control of these cells.

2.6.1 PREVIOUS WORK

The application of computer vision analysis to the characterisation of three-phase froth structures prevalent in flotation cell ore concentrators, was originally investigated by Woodburn *et. al.* (Woodburn, Stockton and Robbins, 1989) in work pertaining to the demineralization of low-rank coals.

As discussed in section 2.3.2 (A), the froth structure in flotation cells varies with froth height and according to Robbins, beneficiation of coal effectively stops at the froth depth where the bubble structure starts to change from close-packed spheres, to the polyhedral form. An optimum froth height therefore exists that maximises separation and Woodburn *et. al.* assert that this froth height, which is visually identifiable as the onset of the polyhedral bubble structure, can be recognised on-line using vision-based image analysis.

The experimental flotation tests were performed using a modified Denver cell. Surface froth images of bubbles at the point of discharge over the cell weir were photographed from above and processed off-line in the following stages :

Stage 1 : The 8 bit resolution (0 to 255 grey scale) images were segmented. This involved the grouping together of contiguous pixels of brightness greater or equal to

a threshold level, to form features that could be further analyzed[2].

Stage 2 : The identified features were then characterized in terms of their area (A), perimeter (P), second moments of area about the x and y axes (I_x, I_y and I_{xy}), and the major and minor axes (I_a and I_b)[3]. The measurements made on each individual feature are quoted in Table 2.2.

Parameter	Parameter Calculation
Area (A)	The total number of pixels in the feature
Perimeter (P)	The total border pixels of the feature
Centroid $(\bar{x},\bar{y})$	$\bar{x}$ = sum of the x positions of each pixel in the feature/n $\bar{y}$ = sum of the y positions of each pixel in the feature/n Where n is the number of pixels in the feature
Moments	$I_x = \text{sum } x^2 - \bar{x}\cdot\text{sum } x$ $I_y = \text{sum } y^2 - \bar{y}\cdot\text{sum } y$ $I_{xy} = \text{sum } xy - \bar{y}\cdot\text{sum } x$ $I_a = \frac{1}{2}(I_x + I_y) + [\frac{1}{2}(I_x - I_y) + I_{xy}^2]^{\frac{1}{2}}$ $I_b = \frac{1}{2}(I_x + I_y) - [\frac{1}{2}(I_x - I_y) + I_{xy}^2]^{\frac{1}{2}}$

Table 2.2 : Feature measurements made by Woodburn, Stockton and Robbins.

Stage 3 : Joint frequency distributions were plotted for the number of features within a size range d_{circ} and elipticity range ellsf, where :

$$d_{circ} = 2.0\cdot(4.0\cdot\pi\cdot A)^{1/2}$$

$$\text{ellsf} = (I_a/I_b)^{1/2}$$

Woodburn *et. al.* concluded that the 3-dimensional surfaces obtained from the surface froth images that were processed, gave a sensitive measure of the appearance of the overflowing froth. Furthermore, they maintain that the optimal froth height can be identified using a machine vision system, and the information obtained from the characterisation of the images can be used to measure the deviation of a particular

---

2 A quantitative description on the methodology used to select the correct threshold value is not given in the paper. Furthermore, from the method of segmentation used it can be concluded that the authors segmented the images using the bubble highlights that are characteristic of individual bubbles, see Chapter 4, section 4.1.1.

3 The expressions for the major and minor axes, I_a and I_b respectively, given in Table 2.2 are incorrect, see Chapter 7, section 7.2.2 and Appendix B.

froth structure from the optimal. Thus forming the basis of a feedback control mechanism for the operation of coal flotation cells.

2.6.2 CURRENT WORK

The work presented in this dissertation makes a significant contribution to that initiated by Woodburn, Stockton and Robbins, in the form of a reliable methodology for the segmentation of surface froth images. This means that for the first time, it is possible to automatically characterise surface froth structures using a machine vision system.

CHAPTER 3

MACHINE VISION AND THE RESEARCH EQUIPMENT CONFIGURATION

"A machine vision system is a system capable of acquiring one or more images of an object, capable of processing, analyzing and measuring various characteristics of the acquired images, and interpreting the results of the measurements in such a way that some useful decision can be made about the object."

R.M. Haralick, Glossary of Computer Vision, Machine Vision International, Ann Arbor, Michigan 48104, USA, 1986.

Machine vision (Horn, 1986) and computational vision (Wechsler, 1990) are branches of the field of computer vision, which, broadly speaking is the science and technology of obtaining models, meanings and control information from visual data.

Computational vision however, seeks a coherent theory of visual perception and understanding, often with the development of computational models of biological vision processes; whereas machine vision is concerned with research and development towards useful practical applications, with the emphasis on working, economical solutions to industrial, medical and military problems, rather than the discovery of new theories on knowledge about human perception.

In practice both the field of machine and computational vision overlap, with the theory of computational vision extending over into machine vision and with the computer architecture developed to solve machine vision problems, influencing the studies in computational vision.

3.1 CONSTITUENT COMPONENTS OF A MACHINE VISION SYSTEM

The objective of a machine vision system is to analyze images of a scene by performing such functions as motion estimation, object location, counting, inspection, recognition or identification and to produce a description of what is imaged. The resultant description must satisfy two criteria (Horn, 1986, p. 4) :

- It must bear some relationship to what is being imaged. This ensures that the description depends in some way on the visual input.
- It must contain all the information needed for performing some given task; that is, the information must be useful. This information can then either be used in a feedback control loop to assist in the automatic control of a desired process, or it can be output for human analysis after which appropriate action can be decided upon.

A machine vision system can be thought of as consisting of three distinct modules, an illumination module, an image acquisition (and digitization) module and an image processing module, as illustrated in Figure 3.1; each of which perform a specific function in the overall operation of the system.

3.1.1 ILLUMINATION MODULE

Illumination is a critical factor in machine vision systems, as the appearance of the image of a scene depends on the lighting conditions, both ambient and controlled. The object of the illumination module is to provide if possible, a controlled lighting environment that visually simplifies the overall scene to be monitored. Take for example, the design of a machine vision system for a particular industrial application. An industrial environment is typically dirty and the ambient lighting is not usually conducive towards any sort of machine vision implementation. It will be easier in the long run to install a high quality lighting configuration

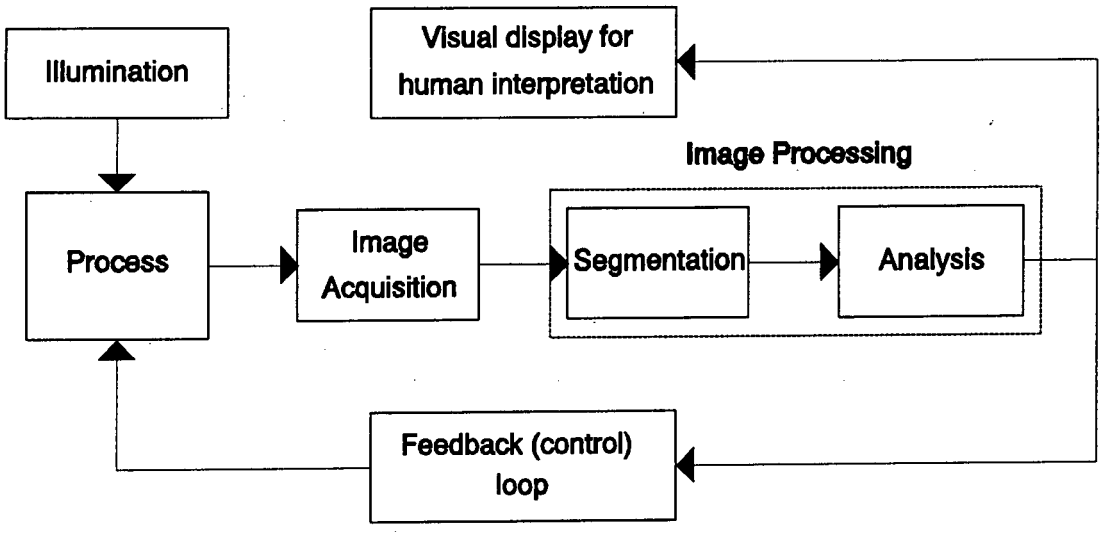


Figure 3.1 : Modular representation of a typical machine vision system.

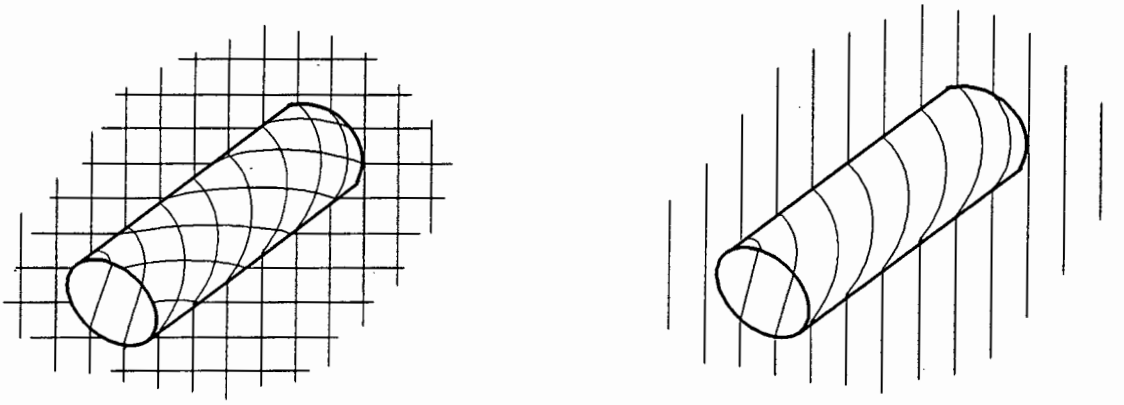
around the scene being viewed, than to spend time and money on the development of algorithms and the purchase of expensive cameras, with the objective of compensating for inadequate or improper lighting conditions.

What then can be defined as good lighting?
In practice this will depend upon the scene being imaged.

If the camera and lighting configuration sits vertically above the scene being viewed, as would be the case in monitoring the surface froth structures in industrial flotation cells, the image is two-dimensional or "flat". A single diffuse lighting source might be used to provide uniform illumination or alternatively, a multiple lighting source might be used to enhance edges and effectively eliminate undesirable shadow regions.

In the case of static three-dimensional scenes, where depth and surface orientation information are of importance, structured lighting techniques are used to spatially encode a scene for analysis (Wang, Mitiche and Aggarwal, 1987), (Dunn, Keizer and Jongdaw, 1989). The technique involves the projection of a known pattern of light, either a grid for grid encoding, Figure 3.2 (a) or single parallel strips for light striping, Figure 3.2 (b), onto the scene and

measuring the distorted pattern to analyze the image.



(a) (b)

Figure 3.2 : Structured lighting techniques.
(a) grid coding and (b) light stripping.

From a knowledge of the geometry of the camera, the light source and the position of the pattern in the measured picture, it is usually a trivial process to calculate the exact coordinate of each recognized point in the pattern. By processing all sides of the object, which is usually achieved by rotating the object in a known way with reference to a fixed light source and camera; it is possible to construct a three-dimensional representation of the image, from which the required information can be extracted.

3.1.2 IMAGE ACQUISITION AND DIGITIZATION MODULE

The objective of this module is to acquire an image of the scene being viewed and then to digitize the image for processing on a computer.

In practice, area-scan cameras are often used for the acquisition of images. These cameras are however inadequate, if a machine vision system is to perform say the on-line analysis of tools on a fast moving conveyor belt, as the images will be blurred due to the conveyor belt motion. If blurring is a problem due to motion of this sort, then it is essential to use a high speed line-scan camera, instead of the conventional area-scan camera. A line-scan camera scans a scene line by line, at regular time intervals and creates an unblurred image of the scene by concatenating the scanned lines. Such a camera may have to be used in the analysis

of fast flowing wet froths in flotation cells, whereas the area-scan camera should suffice for the slower moving dry froths.

Charge-coupled device (CCD) or solid-state cameras are preferred to thermionic cameras for image acquisition because they have a lower power consumption, are more sensitive, more flexible and are reasonably priced (Boyle and Thomas, 1988). The CCD is a photosensitive charge-transfer device[1] that consists of an mxn array of MOS[2] capacitors suitably designed so they are coupled and therefore charges can be moved through the semiconductor substrate in a controlled manner, with each capacitor producing an analogue voltage that is proportional to the amount of light incident on it. The device is essentially an analogue shift register and the charge packets are transferred by using two-, three- or four-phase clocking (Young, 1979, p. 60). Solid-state cameras typically consist of CCD arrays of 256x256 or 512x512 elements and are classified as frame/field transfer or interline transfer devices, which refers to the method used to obtain the video signal.

With a frame/field transfer device, the charge pattern that has accumulated on the photosensitive CCD array is clocked at high speed into a non-photosensitive CCD storage area, during the cameras vertical retrace interval. During the normal horizontal (line) blanking periods the pattern of charges in the storage area are moved downwards by one line into a bottom horizontal register and clocked horizontally to the output to form the video signal.

With an interline transfer device, the storage arrays are arranged alternately with the photosensitive arrays and are connected to them by a transfer gate. During the vertical retrace period, the information is transferred horizontally from the photosensitive electrodes to the storage electrodes by means of the transfer gates and then read out line by line in a similar manner to the frame/field device.

---

1 A semiconductor device in which discrete packets of charge are transferred from one location to the next.

2 Metal-oxide semiconductor.

Monochrome CCD cameras which give rise to black and white images, are often used in preference to colour cameras, for the following reasons :

- Colour cameras are expensive.
- Colour images consist of Red, Green and Blue (RGB) components and therefore need three times the amount of storage required by a single monochrome image.
- It is computationally more expensive to process colour images, because the separate RGB images must all be processed individually.

However in practice, the situation may arise where a particular colour component of a colour image contains substantially more information than that of the monochrome image. It is then preferable to extract and process the identified colour component (Gold, 1991).

The analogue video signal from the camera used to view the scene, is fed into a frame grabber where the signal is digitized at real time video rates into discrete picture elements (pixels), where the resolution of the digitized image is dependant upon the analogue to digital converter that is used. Typically, monochrome images are digitized using 8 data bits, giving a total of 2^8 (256) grey levels, with a range of 0 (black) to 255 (white).

3.1.3 IMAGE PROCESSING MODULE

The objective of the image processing module is to process the image such that suitable features within the image can be identified and parameterized to provide a description of the scene.

This involves two distinct stages :

Stage 1 : Which is a low level processing stage aimed at cleaning and segmenting the image.

Stage 2 : Which is a high level processing stage that uses the segmented image from stage 1, to produce a description of the scene.

Stage 1 is the most important and difficult stage in a machine vision system, as here the machine tries to mimic human perception, by sub-dividing or segmenting the image into

meaningful regions that can then be used to describe the scene. At present there is no general method for reliably segmenting any image and extensive literature exists on various techniques, with the required technique being chosen on the basis of the feature(s) to be extracted (see Chapters 5 and 6).

3.2 EQUIPMENT

In a laboratory setup, the equipment configuration used plays a significant role in the research and development of a machine vision system. The equipment should ideally satisfy two criteria :

- Firstly, it is desirable that there be some sort of control over the data being processed, with regards to the illumination used. It will then be possible to test the reliability of the developed system under various lighting conditions.
- Secondly, a high level computer language is a necessity for the ease of implementation of algorithms. If processing speed is of importance in the final system, the working algorithm can always be optimized either by writing inefficient sections of the code in assembly language or if feasible, by implementing the algorithm in hardware.

3.2.1 RESEARCH EQUIPMENT CONFIGURATION

The equipment configuration used for researching the characterisation of surface froth structures using machine vision consisted of :

- a sophisticated VHS video cassette recorder, with digital memory allowing frame-by-frame analysis of data.
- a Matrox MVP-AT colour frame grabber and RGB monitor, for the acquisition and display of images.
- a 16 MHz 80386/SX IBM pc clone, with math coprocessor and 4 Mbytes of memory, running under MS-DOS.
- a SUN Sparc II workstation running under UNIX, with a network link to the microcomputer.

This configuration is illustrated in Figure 3.3 and shows the interconnection between the

various units.

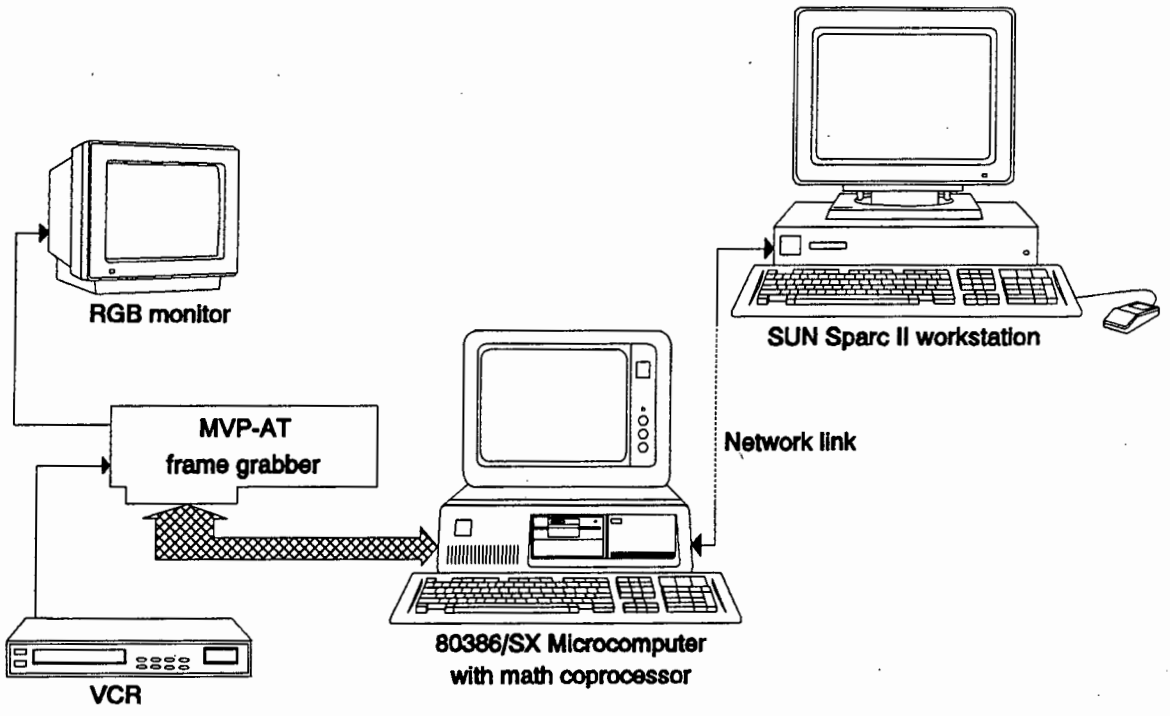


Figure 3.3 : Equipment configuration.

The video recorder proved to be an effective and efficient method of viewing the data to be processed, as it eliminated the need for a costly experimental flotation cell with the necessary camera and lighting assembly. This however, did have a major drawback, as all the data was pre-supplied on video tapes and the author effectively had no control over the camera and lighting configurations used to take the data.

Images for off-line processing were digitized and captured using the MVP-AT frame grabber (MATROX Electronic Systems Limited, 1989); a colour frame grabber, capable of capturing 512x512 black and white images or the R, G and B components of a colour image, and performing a variety of image processing functions. The frame grabber has 1 Mbyte of memory that is mapped into the pc memory space and which can be accessed in 64 Kbyte blocks. This memory can in practice be configured in software, to provide a number of different frame buffers, as given in Table 3.1.

FRAME BUFFERS		
BUFFER #	MEMORY	COMMENT
0, 1, 2, 3	4 x 512 x 512 x 8	The basic buffers.
4	1 x 512 x 512 x 16	Combines FB0 and FB1. FB1 = MSB.
5	1 x 512 x 512 x 16	Combines FB2 and FB3. FB3 = MSB.
6	3 x 512 x 512 x 8	R, G, B = FB0, FB1, FB3. FB2 is used as an overlay buffer.
7	1 x 1024 x 1024 x 8	One big space.
8	2 x 512 x 512 x 8	*Compressed colour. BR = FB0, OG = FB1.
9	2 x 512 x 512 x 8	*Compressed colour. BR = FB2, OG = FB3.
*4 bits each for Red, Green, Blue, Overlay		

Table 3.1 : MVP-AT frame buffer memory configurations.

The initial investigative work into the segmentation of surface froth images was implemented on the pc using the MVP-AT and Microsoft C version 6.0. Microsoft C was specifically chosen because the MVP-AT software libraries were all compiled with Microsoft C and thus could not be used with any other C compiler. However, with the acquisition of two SUN workstations, the work, especially that documented in Chapter 6, was transferred to the workstations for reasons that are discussed in Chapter 6. Thereafter, the MVP-AT was only used for capturing images.

3.2.2 EQUIPMENT LIMITATIONS

As mentioned previously, the experimental equipment configuration used for the research and development of a machine vision system for the characterisation of surface froth structures, had a single drawback; because the data for processing was supplied on video tapes, the author had no direct control over the lighting configurations used and as a result the data was in most cases suboptimal.

This did, however, have one good point, as the eventual segmentation technique was developed using suboptimal data, which means that the system should work well in a suitably controlled operating environment.

Two video tapes of material were supplied for the duration of the project :

- The first video tape, contained data shot at the Black Mountain Mineral Development (Chadwick, 1981), situated in the North Western Cape Province, near Aggeneys. Here lead, copper, zinc and silver are extracted from complex ores by sequential flotation. Arbitrary lighting and camera positioning were used and two images from this tape, BUB13.CFI and BUB14.CFI, are given in Appendix D. The data on this tape was generally ignored because of the improper camera and lighting configuration that was used. The eventual segmentation technique was however applied to the two images (see Chapter 6, section 6.5) and adequately segmented them, although the segmented images were not analyzed any further.
- The second video tape was shot at MINTEK, Randburg, during an experimental graphite flotation run using an existing pilot plant. Here, as a result of suggestions, an improved lighting configuration was used, although it was still not what can be considered as optimal. The surface froth was illuminated with a dual lighting source, and the camera was held vertically above the froth surface inbetween the two lights and at a slight tilt to the surface. The material was considerably better than that of the first tape, and as a result images captured from this tape were used in the characterisation of surface froth structures, see Appendix D, BUB1.CFI to BUB12.CFI.

Because there was inadequate control of the lighting, no detailed research was carried out on particular configurations, because it was not possible to investigate various techniques on a practical basis. As a result there are numerous unanswered questions on the optimal lighting configuration that should be used; such as the number of lights, their intensity and the orientation of the lights with respect to the camera and surface froth.

3.2.3 REQUIREMENTS FOR A DEDICATED SYSTEM

The development of a machine vision system is bounded by certain performance specifications and operational requirements. For the characterisation of surface froth structures, these requirements are :

- The overall system should be a low cost system, capable of being implemented on

a personal computer using a frame grabber.

- The algorithms should be software implementable and flexible. The use of dedicated hardware being undesirable because of the expense involved in manufacturing it.
- The dynamics of flotation cells is such that real-time operation, although preferable, is not an absolute necessity and pseudo real-time operation of within 5 minutes would be acceptable.
- The machine vision system must be capable of providing a reliable statistical description of bubble size and bubble shape, for surface froth structures, so that a mean value of both measurements can be produced as output and used as control information.

CHAPTER 4

THE SEGMENTATION OF SURFACE FROTH IMAGES

"Image segmentation is a process which typically partitions the spatial domain of an image into mutually exclusive subsets, called regions, each one of which is uniform and homogeneous with respect to some property such as tone or texture and whose property value differs in some significant way from the property value of each neighboring region."

R.M. Haralick, Glossary of Computer Vision, Machine Vision International, Ann Arbor, Michigan 48104, USA, 1986.

In machine vision systems, image segmentation is a fundamental processing step that has two objectives :

- Firstly, the grouping of pixels with identical or similar characteristics to form specific regions, giving rise to "objects of interest", which can then be further analyzed to provide a description of the image.
- Secondly, data compaction. Most segmentation techniques produce a binary image from a grey level image, thereby reducing the amount of data required to fully represent the image.

Numerous segmentation techniques exist (Haralick and Shapiro, 1985), each of which differ in the way they emphasize one or more of the desired properties being used to segment the image. For a particular problem the technique used is chosen on the basis of the feature(s) to be extracted.

4.1 VISUAL FEATURES THAT CHARACTERISE FROTH

Froth structures exhibit two inherent visual features that provide a measure of bubble size and shape, and thus can be utilised in the segmentation of surface froth images. These features are bubble highlights and bubble boundaries.

4.1.1 BUBBLE HIGHLIGHTS

Bubble highlights are the bright spots occurring on the surfaces of individual bubbles. Intuitively, these highlights are dependant upon the size and surface curvature of the individual bubbles and consequently provide an indirect and proportional measure of bubble size and shape.

Due to the variation in the assembly of bubbles that constitute the surface froth structures in flotation cells and numerous complex illumination effects, a complete analytical treatment of bubble highlights is exceptionally difficult to derive. It will suffice here, using fundamental shape-from-shading concepts, to provide a qualitative description of bubble highlights.

As previously stated, bubble highlights are identified as the characteristically bright surface spots on individual bubbles and are dependant upon :

- The position of the bubbles relative to the camera.
- The shape, size and surface curvature of the bubbles.
- The lighting distribution.
- The surface reflectance properties of the bubbles.

For any shaped three-dimensional object, the more a surface slopes away from the light falling on it, the less light reaches the camera, thus giving rise to a variation in surface brightness known as *shading*. It is this spatial variation in brightness, that is used in shape-

from-shading techniques, to deduce surface orientation from image intensity. In practice, the analysis of shape-from-shading is complex due to :

- Primary illumination effects, such as reflection, absorption and refraction.
- Secondary illumination effects, where light bounces off one surface and onto another, and thence towards the camera or viewer.
- Mutual illumination effects, where the light reflected from one object, say object A, illuminates a second object say object B, and the light reflected by object B in turn illuminates object A.
- Surface reflectance.

The surface reflectance function profoundly influences the shape-from-shading problem. The fraction of light reflected towards the viewer from a source depends on the microstructure of the reflecting surface, and is usually described as a function of three angles :

- angle of incidence i, between the source and the surface normal.
- angle of emittance e, between the line of sight to the camera (or viewer) and the surface normal.
- phase angle g, between the incident and emitted rays.

These angles are illustrated in Figure 4.1.

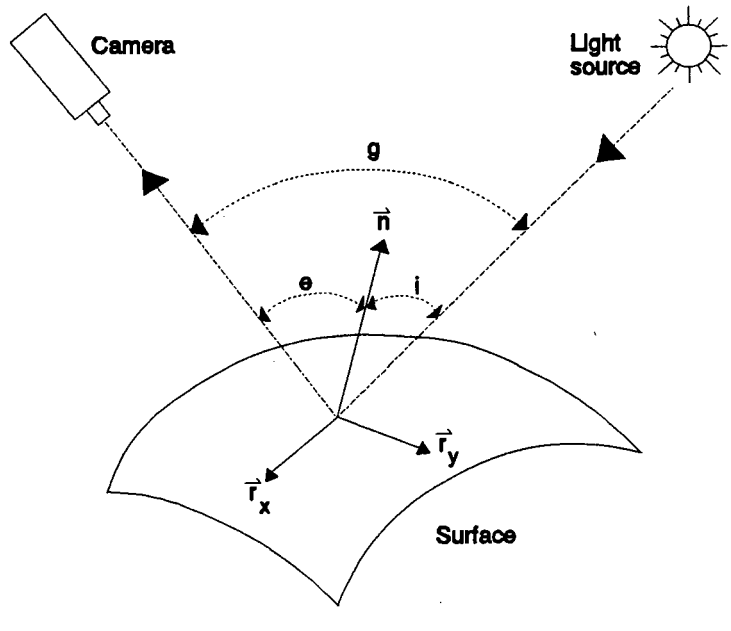


Figure 4.1 : The definitions of the angles i, e and g.

The reflectance function $\Phi(i,e,g)$ (Horn, 1986, p. 210), is defined as the fraction of incident light reflected per unit surface area per unit solid angle in the direction of the viewer, and tells how bright a surface appears from one direction, while light falls on it from another. Typically for matte or diffuse surfaces, the function varies with the cosine of the incident angle (known as the "cosine" or Lambert's law of reflection for matte surfaces, see Horn, 1986, p. 214), indicating that the most light is reflected when the source is normal to the surface. For specular or mirror-like surfaces, maximum reflection is obtained when the surface normal points about half way between the light source and the viewer, that is when $i = e$.

In practice, most everyday surfaces have a reflectance function that combines both a matte and specular component.

The reflectance function is used extensively in the shape-from-shading problem, to generate reflectance maps, that explicitly show the relationship between surface orientation and brightness. Reflectance maps encode information about surface reflectance properties and light source distributions and are a representational tool used in developing methods for recovering surface shape from shading.

How then does all this fit in with bubble highlights?

For simplicity, assuming that surface froth structures are illuminated with a point source, sitting vertically above the surface, at essentially the same place as the camera, the highlights will be visible as bright spots on the bubble surfaces, which will correspond to a maximum point on the reflectance map, for the given lighting configuration. The brightness will fall off from its maximum in the centre, to a minimum at the edges, according to the curvature of the individual bubbles.

This can be explained with the aid of Figures 4.2 and 4.3. Figure 4.2, is an idealised, cross-sectional view of a small, spherically shaped bubble, that can be assumed to be typical of the bubbles in a wet froth. The diagram shows the incident (i) and reflected (e) light rays, with respect to the surface normals, assuming that bubble surfaces exhibit both matte and specular

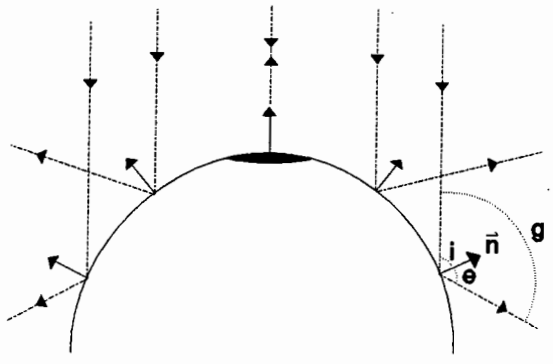


Figure 4.2 : Bubble highlight formation for an idealised spherically shaped bubble.

components. Because of the high surface curvature, there will be a small region on the top of the bubble, where the incident light is near normal to the surface and there will thus be maximum reflection of the light into the camera. This small patch, shaded black, corresponds to the bubble highlight. Due to the surface curvature, light rays falling outside this region strike the surface at an angle of incidence greater than a certain limit and are not reflected directly back into the camera; thus the measured intensity of the bubbles falls off towards the bubble boundaries.

Like Figure 4.2, Figure 4.3 is an idealised cross-sectional view of a bubble, but this time it is of the larger elongated type, associated with dry froths.

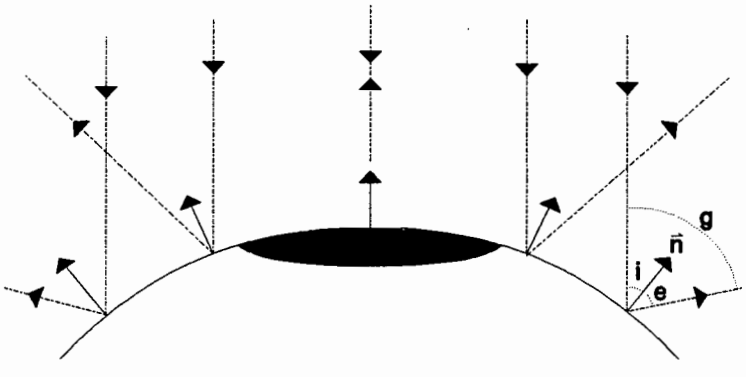


Figure 4.3 : Bubble highlight formation for an idealised elongated bubble.

Because of the low surface curvature, there is a larger area on the top of the bubble for which the incident light is near normal to the surface and is therefore reflected directly back into the camera. This region, again shaded black, corresponds to the bubble highlight. As previously discussed, outside of this region, the incident light is no longer near normal to the bubble surface and the measured intensity falls off towards the bubble boundary.

4.1.2 BUBBLE BOUNDARIES

Bubble boundaries are the valleys separating individual bubbles and an intensity profile (see Figure 5.2, Chapter 5) shows them to be luminance minima (Pearson and Robinson, 1985). The boundary of a bubble completely specifies the surface area occupied by the region of the bubble exposed at the froth surface and therefore provides a direct measure of the size and shape of the surface bubbles.

4.2 SEGMENTATION PROBLEMS

Bubble highlights and boundaries are both lighting dependant features and as such their segmentation is complicated or simplified according to the lighting configuration used. From the video material analyzed, bubble boundaries tend to be the more stable and reliable feature under varying lighting conditions; additionally, several prominent problems arising from incorrect lighting were identified. These being :

- bubble highlight saturation if the lighting source is not diffuse.
- multiple highlights per bubble if more than one lighting source is used.
- shadow regions that mask the sharpness of bubble boundaries, if the angle of the lighting is too obtuse.

These problems are detailed in Chapters 5 and 6.

4.3 APPROACH TO THE SEGMENTATION PROBLEM

Knowing what features to look for in surface froth images, the segmentation problem reduces to understanding what can be defined as correct segmentation and how to go about segmenting the images using these features.

What is correct segmentation?

The answer to this question depends on what is to be done with the results. In the case of surface froth structures, the objective is to (statistically) characterise these structures in terms of bubble size and shape. Thus the correct segmentation of surface froth images can be defined as the ability to "accurately" and reliably segment the images using either the individual bubble highlights or preferably the bubble boundaries. As no segmentation technique can claim to be error-free and thus 100% accurate, it is therefore imperative that the performance of the eventual segmentation technique be gauged (see Chapter 6, section 6.6.4), to provide a measure of the efficiency of the process and thus its accuracy.

What segmentation technique should be used?

Froth structures occurring in industrial flotation cells are complex 3-dimensional surfaces that consist of considerable natural shape information, in the form of the curvature of the individual bubble surfaces. This information is conveyed by the grey level intensities of the 2-dimensional images. Intensity profiles of surface froth images reveal well defined peaks and deep crevices. The former are indicative of bubble highlights and the latter of bubble boundaries.

Initially, conventional segmentation techniques were investigated under the assumption that they may be suitable for extracting the peak intensity regions, as well as the deep crevices. The work, documented in Chapter 5, includes the use of automatic threshold selection for bubble highlight extraction, and case studies of various edge detection techniques for bubble boundary extraction. As will be shown, the techniques investigated failed to give correct segmentation.

Shape-from-shading techniques were ruled out for two reasons. Firstly these techniques are computationally intensive and are therefore time consuming and secondly, the objective of shape-from-shading techniques is to recover the shape of the surface, using the spatial variation of brightness to estimate surface orientation. This would not justifiably simplify the segmentation problem.

From further analysis of intensity profiles, it was realized that a suitably shaped probe might be used to effectively extract the peaks or crevices. This realization motivated an investigation into morphological image processing techniques. The work, documented in Chapter 6, resulted in the development of a reliable method for the automatic segmentation of surface froth images.

CHAPTER 5

THE CLASSICAL SEGMENTATION OF SURFACE FROTH IMAGES

The surface froth image illustrated in Figure 5.1, is typical of the type of images processed (see Appendix D) and will be used as a test image in both this and the next chapter, to illustrate the performance of the segmentation techniques that were investigated.

The inverted line through row 256 of the image, represents the location through which a horizontal intensity profile was taken. From the profile, Figure 5.2, the bubble highlights and bubble boundaries are clearly discernable features which motivated an initial investigation into the application of standard (linear) or *classical* segmentation techniques. Furthermore, excessive shadow regions are identifiable, which are unusable due to a substantial loss of unrecoverable information.

On viewing the image, the human eye uses both the bubble highlights and bubble boundaries to quickly and efficiently identify the individual bubbles. For a machine vision system though, this process of identification is considerably more difficult and it must be stressed that the success of automated characterisation hinges on the ability, as mentioned previously, to accurately and reliably segment surface froth images.

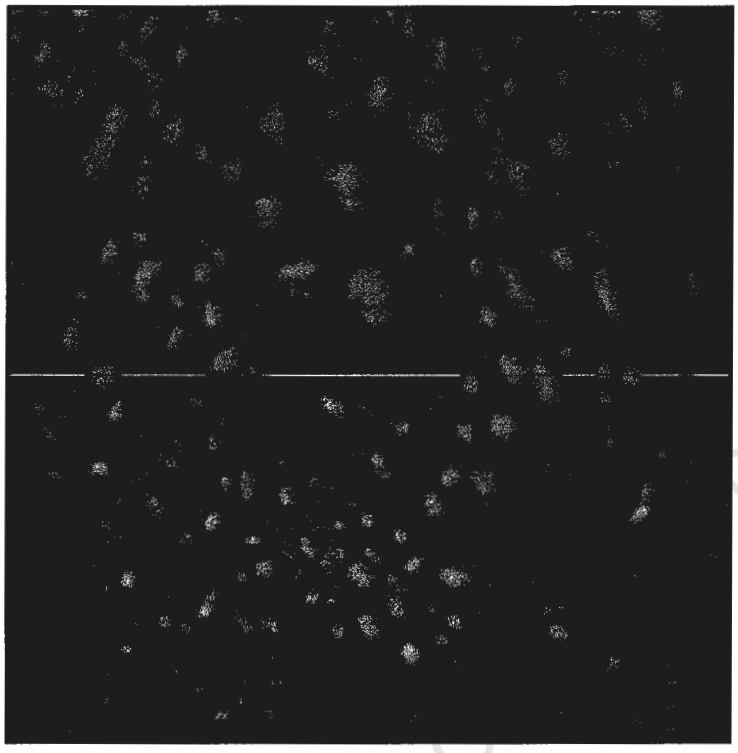


Figure 5.1 : Test image - BUB1.CFI.

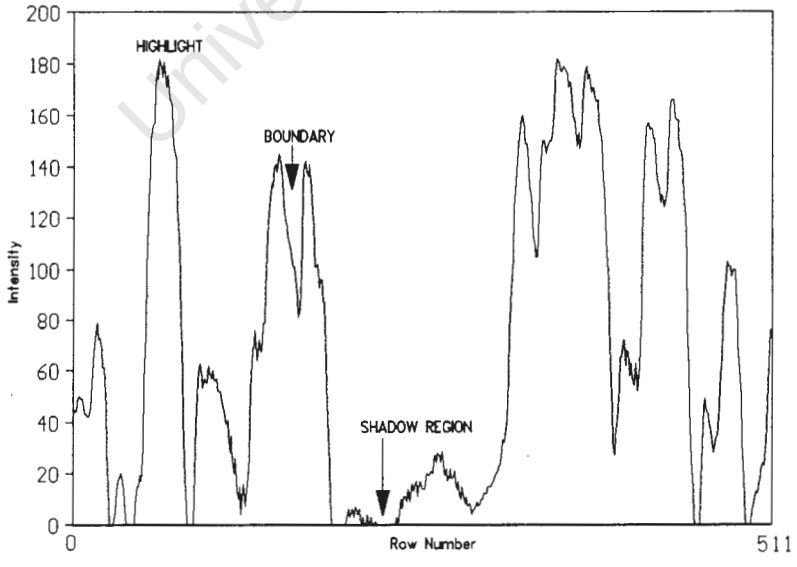


Figure 5.2 : BUB1.CFI intensity profile for row 256.

5.1 BUBBLE HIGHLIGHT SEGMENTATION USING AUTOMATIC THRESHOLD SELECTION

Image thresholding is a popular and well documented segmentation technique. If f(x,y) represents the input image, b(x,y) the thresholded output image and T a threshold level, the relationship between the input and output image is given by :

$$b(x,y) = \begin{cases} 0 & , \ f(x,y) < T \\ 1 & , \ f(x,y) \geq T \end{cases} \qquad \ldots(5.1)$$

The grey level histogram, Figure 5.3, is used in the selection of a threshold level T. It is a function h, that shows the number of pixels h(j) in an image with intensity j. The function provides a useful description of the distribution of the grey levels within an image; but all spatial information about the individual pixels is lost.

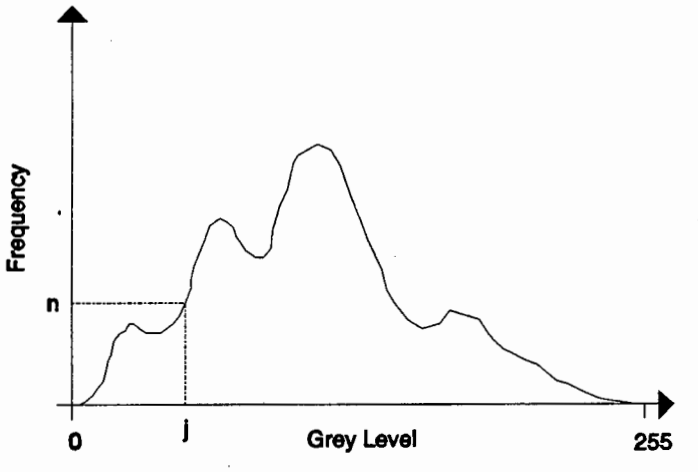


Figure 5.3 : Grey level histogram.

For simple images consisting of a light object on a dark background, the grey level histogram is bimodal and the optimum threshold level is the minimum level in the valley between the two peaks.

Real images though, give rise to unimodal or polymodal histograms that makes threshold selection difficult and it is necessary to use automatic threshold selection techniques to compute the optimum threshold level. A variety of different techniques exist which have been

comprehensively covered by Sahoo *et. al.* (Sahoo, Soltani and Wong, 1988) in a paper updating the work by Weszka (1978).

5.1.1 CASE STUDY : MOMENT-PRESERVING BILEVEL THRESHOLDING

The use of moment-preserving techniques in image processing are not new and have been successfully investigated and implemented by several researchers. Tabatabai and Mitchell (1984) developed a new edge operator using moment theory. They found that an ideal edge could be fitted to a sequence of empirical edge data, by preserving the first three moments of the input data. Delp and Mitchell (1979) have developed an image compression technique known as Block Truncation Coding (BTC), using moment-preserving theory. They found that suitable image compression, that preserves the local statistics of an image, could be obtained by splitting the image into nxn blocks and then coding the individual blocks into a two level signal; with the levels chosen such that the first two moments of the blocks were preserved. Otsu (1979) has used the zeroth- and first-order moments of the grey level histogram for optimal threshold selection.

Moment-preserving thresholding was presented by Tsai (1985) as a new approach for automatic threshold selection, that computes the optimum threshold level using the first three moments (mean, variance and skewness) of the image. The threshold level is selected in such a way as to preserve the moments of the thresholded image. The technique produces consistently good results, a fact that has been substantiated by Brink (1987).

The formulation of the moment-preserving equations

The formulation of the moment-preserving equations for bilevel thresholding have only been considered because the segmentation of surface froth images to extract bubble highlights, is a bilevel problem. However, the techniques presented are easily extended to cater for multilevel thresholding.

The grey level histogram of an image with 8 bit quantisation, is given by $[n_0,n_1,\ldots,n_j,\ldots, n_{254}, n_{255}]$, where the n_j entry, Figure 5.3, denotes the number of pixels n, with grey level j.

If the total number of pixels in the image is N, the grey level probability histogram is calculated using the expression :

$$P_j = \sum_{j=0}^{255} \frac{n_j}{N} \qquad \ldots(5.2)$$

Where the ith moment of the grey level probability histogram is given by :

$$m_i = \sum_{j=0}^{255} P_j \cdot (j)^i \qquad \ldots(5.3)$$

Using equation (5.3) the zeroth, first (mean), second (variance) and third (skewness) moments are :

$$\begin{aligned} m_0 &= \sum_{j=0}^{255} P_j = 1 \\ m_1 &= \sum_{j=0}^{255} P_j \cdot (j) \\ m_2 &= \sum_{j=0}^{255} P_j \cdot (j)^2 \\ m_3 &= \sum_{j=0}^{255} P_j \cdot (j)^3 \end{aligned} \qquad \ldots(5.4)$$

For bilevel thresholding there is a single threshold level T, that will be used to dichotomize the image pixels into two classes z_0 and z_1. If p_0 is denoted as those fraction of pixels below T and p_1 as those fraction above T, then the new moments are given by the expression :

$$m'_i = \sum_{k=0}^{1} p_k (z_k)^i \qquad \text{For } i = 0,1,2,3 \qquad \ldots(5.5)$$

Which on expanding yields :

$$\begin{aligned} m'_0 &= p_0(z_0)^0 + p_1(z_1)^0 \\ m'_1 &= p_0(z_0)^1 + p_1(z_1)^1 \\ m'_2 &= p_0(z_0)^2 + p_1(z_1)^2 \\ m'_3 &= p_0(z_0)^3 + p_1(z_1)^3 \end{aligned} \qquad \ldots(5.6)$$

The moments of the original and thresholded image are preserved by equating m_i and m'_i (for i=0,1,2,3), thereby yielding four equalities which are known as the moment-preserving equations for bilevel thresholding :

$$\begin{aligned} p_0 + p_1 &= 1 && (i) \\ p_0 z_0 + p_1 z_1 &= m_1 && (ii) \\ p_0 (z_0)^2 + p_1 (z_1)^2 &= m_2 && (iii) \\ p_0 (z_0)^3 + p_1 (z_1)^3 &= m_3 && (iv) \end{aligned} \qquad \ldots(5.7)$$

To find the required threshold level T, equations (5.7) (i), (ii), (iii) and (iv) have to be solved to obtain p_0 and p_1. The threshold T, is then chosen by determining the p_0-tile from :

$$p_0 = \sum_{j=0}^{T} P_j \qquad \ldots(5.8)$$

In general the p_0-tile is not solved for exactly and the required threshold value is taken as the nearest grey level value.

Solving the moment-preserving equations

Equations (5.7) involve four unknowns, p_0, p_1, z_0 and z_1, that need to be solved. Rewriting equations (5.7) (i) and (ii) in matrix format, expressions for solving p_0 and p_1 are easily derived.

$$\begin{bmatrix} 1 & 1 \\ z_0 & z_1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \end{bmatrix} = \begin{bmatrix} 1 \\ m_1 \end{bmatrix} \qquad \ldots(5.9)$$

The unknowns p_0 and p_1 can be solved using Cramer's rule, to give :

$$\begin{aligned} p_0 &= \frac{1}{p_{det}} \begin{vmatrix} 1 & 1 \\ m_1 & z_1 \end{vmatrix} \\ p_1 &= 1 - p_0 \end{aligned} \qquad \ldots(5.10)$$

Where :

$$p_{det} = \begin{vmatrix} 1 & 1 \\ z_0 & z_1 \end{vmatrix} = z_1 - z_0$$

Equation (5.10) shows that the evaluation of p_0 and p_1 is dependent upon z_0 and z_1. It has been shown, according to Tsai, that in general the moment-preserving equations can be solved indirectly in the following two steps :

Step 1: Using the moment values m_0, m_1, ..., m_{TL+1} and auxiliary values c_0, c_1, ..., c_{TL}, where the subscript TL is the number of classes the image is to be segmented into, a set of linear equations can be established. For bilevel thresholding (TL=2) the set of linear equations are given by :

$$\begin{aligned} c_0 m_0 + c_1 m_1 + c_2 m_2 &= 0 \\ c_0 m_1 + c_1 m_2 + c_2 m_3 &= 0 \end{aligned} \qquad \text{...(5.11)}$$

Setting c_2 to 1, the system can be rewritten in matrix format to give :

$$\begin{bmatrix} m_0 & m_1 \\ m_1 & m_2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} -m_2 \\ -m_3 \end{bmatrix} \qquad \text{...(5.12)}$$

Again using Cramer's rule, the unknowns c_0 and c_1 can be solved to give :

$$\begin{aligned} c_0 &= \frac{1}{c_{det}} \begin{vmatrix} -m_2 & m_1 \\ -m_3 & m_2 \end{vmatrix} \\ c_1 &= \frac{1}{c_{det}} \begin{vmatrix} m_0 & -m_2 \\ m_1 & -m_3 \end{vmatrix} \end{aligned} \qquad \text{...(5.13)}$$

Where :

$$c_{det} = \begin{vmatrix} m_0 & m_1 \\ m_1 & m_2 \end{vmatrix} = m_0 m_2 - m_1^2$$

Step 2: The representative grey levels z_0, z_1, ..., z_{TL-1} are obtained by solving a polynomial in z of degree TL with coefficients c_0, c_1, ..., c_{TL}. For bilevel thresholding, the required second order polynomial is given by :

$$z^2 + c_1 z + c_0 = 0 \qquad \text{...(5.14)}$$

The solutions for z_0 and z_1 are obtained using the standard solution for a second order polynomial :

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2} \qquad \ldots(5.15)$$

giving :

$$z = \frac{-c_1 \pm \sqrt{c_1^2 - 4c_0}}{2} \qquad \ldots(5.16)$$

or :

$$z_0 = \frac{-c_1 - \sqrt{c_1^2 - 4c_0}}{2}$$
$$z_1 = \frac{-c_1 + \sqrt{c_1^2 - 4c_0}}{2} \qquad \ldots(5.17)$$

Implementation

The technique was implemented according to the algorithm listed in Figure 5.4.

```
for (row=0;row<512;row++)
        for (col=0;col<512;col++)
                compute the grey level probability histogram      /* Using equation (5.2)   */

for (i=0;i<255;i++)
        compute the zeroth, first, second and third moments
        of the grey level probability histogram                   /* Using equations (5.4)  */

compute c_det, c_0 and c_1                                        /* Using equations (5.13) */
compute z_0, z_1                                                  /* Using equations (5.17) */
compute p_det, p_0 and p_1                                        /* Using equations (5.10) */

t = 0
do {
        t = sum grey level probability histogram values           /* Using equation (5.8)   */
} while (t <= p_0)

threshold T = nearest grey level value to t*255
```

Figure 5.4 : Algorithm for the implementation of moment-preserving bilevel thresholding.

The histogram of the test image, illustrating the selected threshold level, is given in Figure 5.5 and the resultant thresholded image in Figure 5.6.

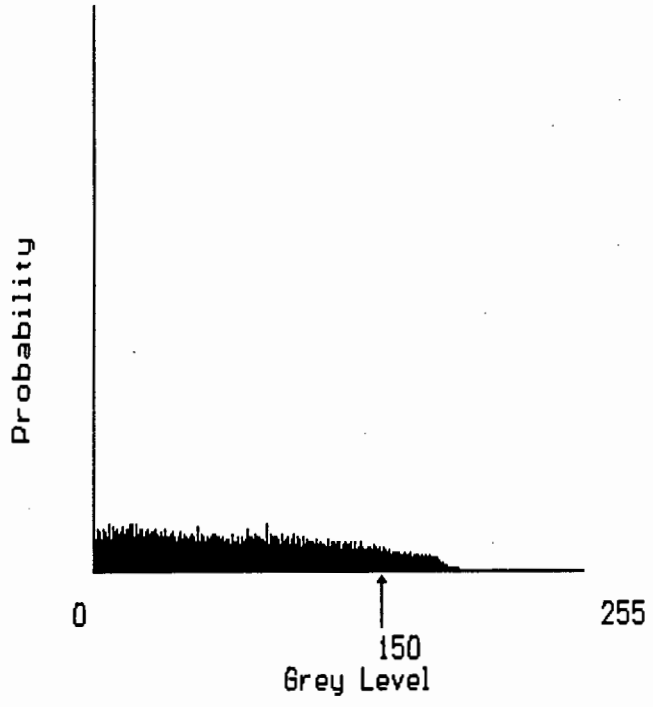


Figure 5.5 : Grey level probability histogram of BUB1.CFI.

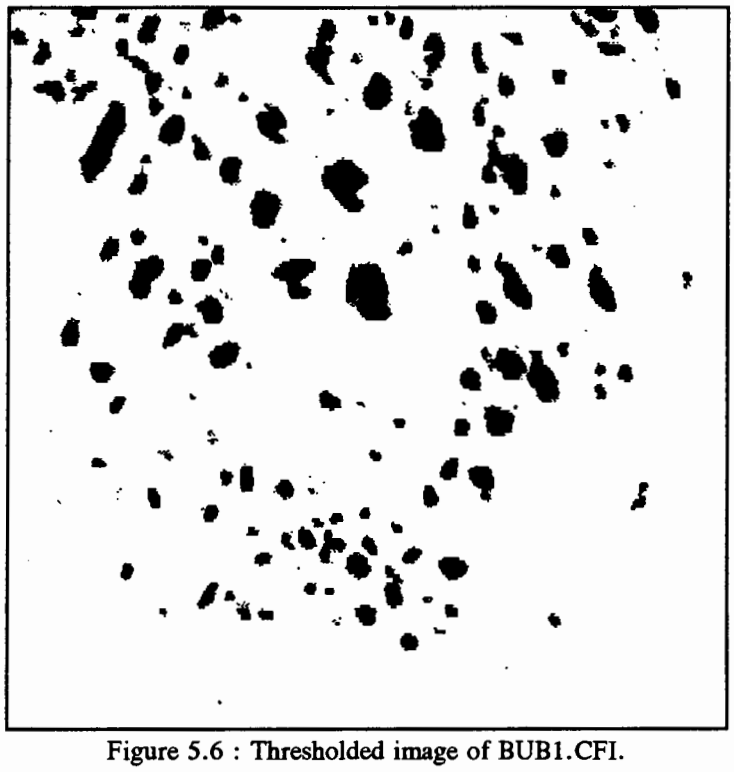


Figure 5.6 : Thresholded image of BUB1.CFI.

A comparison of the thresholded image with the original image (Figure 5.1), shows that the computed threshold level effectively extracts the highlights of the larger bubbles, but misses the highlights of the smaller bubbles, resulting in inadequate segmentation. By subdividing the image into nxn blocks of size 32x32, 64x64 or 128x128 and applying moment-preserving thresholding to the individual blocks, it was found that merging of the bubble highlights occurred and it was thus concluded that the use of threshold selection techniques for the extraction of bubble highlights, was not a suitable approach to the segmentation of surface froth images.

5.2 BUBBLE BOUNDARY SEGMENTATION USING EDGE DETECTION TECHNIQUES

Edges are curves or lines in an image where rapid changes occur in brightness or in the spatial derivatives of brightness and arise from :

- occluding contours of objects
- discontinuities in surface orientation
- discontinuities in surface reflection properties

The fundamental underlying principle of most edge-detection techniques is the computation of a local derivative operator, Figure 5.7, that provides a scalar quantity that is indicative of edge strength and such edge-oriented methods give rise to two classes of classical edge operators :

- Gradient (∇) Operators : that compute edge strength and direction using the first derivative.
- Laplacian (∇^2) or Zero-crossing Operators : that compute edge strength using the second derivative. There is no edge direction information.

In practice, local 3x3 edge operators are used and segmentation of an image is performed in two stages (Lacroix, 1988) :

Stage 1 : Involves the formation of an edge image, where all pixels within the image are either classified as edge points or non-edge points. This is achieved by designing a suitable edge operator that computes edge strength and direction for each pixel. The

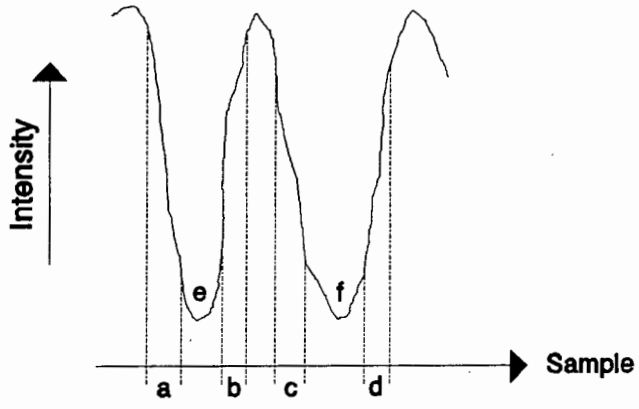


Figure 5.8 : Simplified bubble intensity profile indicating possible edges for extraction.

5.2.1 CASE STUDY 1 : THE SOBEL OPERATOR

Mathematically, the vector differential or grad operator (∇) for a two dimensional system is defined as :

$$\nabla = \vec{i}\,\frac{\partial}{\partial x} + \vec{j}\,\frac{\partial}{\partial y} \qquad \ldots(5.18)$$

When operating on a function f(x,y), the resultant magnitude is a scalar quantity which represents the maximum rate of change of f at the point x,y.

$$\textit{Gradient} = |\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad \ldots(5.19)$$

Square roots are computationally expensive and it is common practice to approximate the gradient by absolute values :

$$|\nabla f| \approx \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right| \qquad \ldots(5.20)$$

The direction $\alpha(x,y)$ of the gradient, measured with respect to the x-axis, is given by :

$$\alpha(x,y) = \tan^{-1}\left[\frac{\left(\frac{\partial f}{\partial y}\right)}{\left(\frac{\partial f}{\partial x}\right)}\right] \qquad \ldots(5.21)$$

y \ x			
	x-1,y-1	x,y-1	x+1,y-1
y	x-1,y	x,y	x+1,y
	x-1,y+1	x,y+1	x+1,y+1

Figure 5.9 : 3x3 image region.

Thus for a 3x3 mask, Figure 5.9, the Sobel Operator S(x,y) is defined as :

$$S(x,y) = |s_x| + |s_y| \qquad \ldots(5.22)$$

Where s_x is the x component of the gradient vector :

$$s_x = (f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)) - (f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)) \qquad \ldots(5.23)$$

and similarly s_y is the y component :

$$s_y = (f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)) - (f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)) \qquad \ldots(5.24)$$

Figure 5.10 shows the result of edge detecting the test image using a 3x3 Sobel operator. For aesthetic reasons only, the image has been inverted so that the strongly detected edges are represented by black and regions with no edges are represented by white. It can be seen from the edge detected image that there is a considerable amount of noise, which was to be expected since the Sobel operator is a first derivative operator and in general is noise sensitive. To complete the segmentation of the image, the edge detected image was thresholded using an arbitrarily selected threshold level of 128, to extract the bubble boundaries. The thresholded image is given in Figure 5.11 with the extracted bubble boundaries displayed in black. It can be seen that although the chosen threshold level has eliminated most of the noise, the Sobel operator has given rise to incomplete edge detection. It can be concluded therefore that this form of edge detection is not suitable for the extraction

of bubble boundaries.

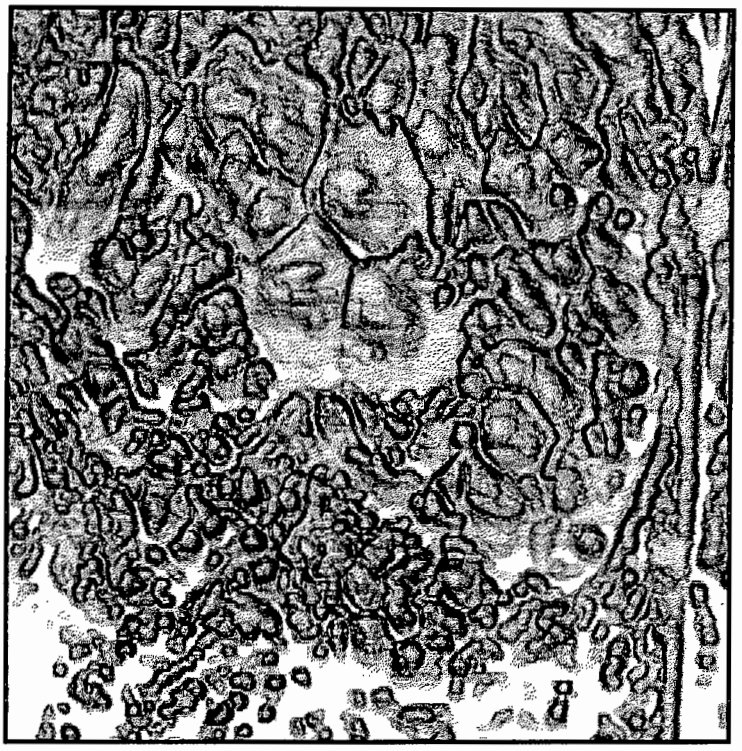


Figure 5.10 : BUB1.CFI edge detected using a 3x3 Sobel operator.

Figure 5.11 : Thresholded, edge detected image of BUB1.CFI.

5.2.2 CASE STUDY 2 : THE LAPLACIAN OPERATOR

Mathematically, the Laplacian (∇^2) operator is defined as :

$$\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \qquad \ldots(5.25)$$

For a function f(x,y), the Laplacian of the function is a scalar quantity given by :

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \ldots(5.26)$$

Horn (1986, pp. 164-168), provides a comprehensive treatment of the discrete Laplacian operator and it will suffice here to say that the discrete implementation of the Laplacian operator, with reference to Figure 5.9, is given by :

$$L(x,y) \approx f(x-1,y) + f(x,y+1) + f(x+1,y) + f(x,y-1) - 4f(x,y) \qquad \ldots(5.27)$$

Application of the operator to the test image produced the result illustrated in Figure 5.12. Again the image has been inverted and the detected edges are visible as the black lines in the image. Because the operator is a second derivative operator, there is considerable high

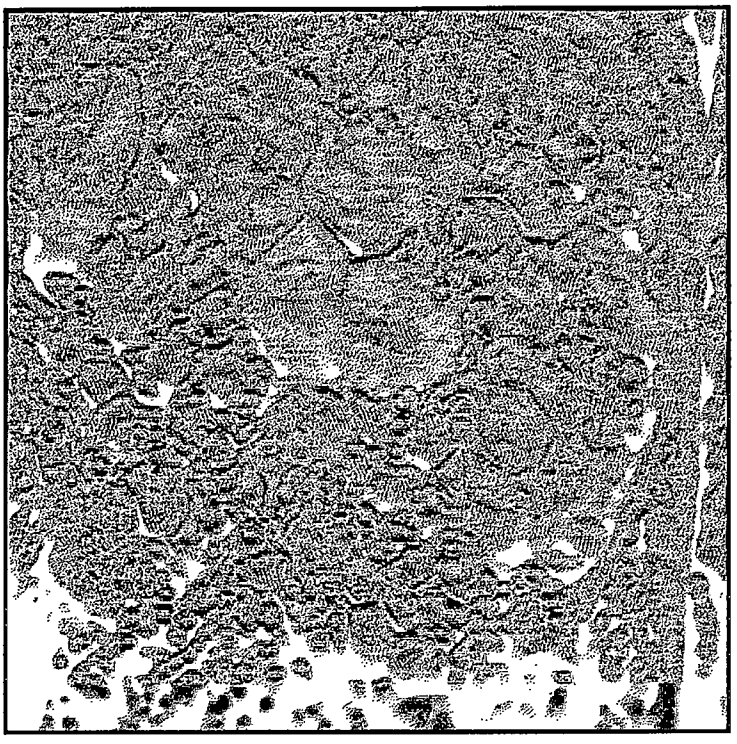


Figure 5.12 : BUB1.CFI edge detected using the Laplacian operator.

frequency noise and thresholding the image to extract the detected edges is not feasible.

5.2.3 CASE STUDY 3 : A VALLEY OPERATOR

A valley operator differs from conventional edge detection techniques in the sense that it does not detect changes in intensity but looks for a minimum intensity or valley point within a region of an image. The valley operator presented here was developed and used by Pearson and Robinson (1985) and is an oriented valley detector, implemented using the 5x5 mask illustrated in Figure 5.13.

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

Figure 5.13 : 5x5 Valley operator mask.

The operator sequentially checks for valleys in the four orientations, N-S, NE-SW, E-W and SE-NW, using three threshold levels to determine the presence of a valley. The first level T_1, which is applied to a 3x3 image region, is used as an initial test for all orientations. If it is passed and the orientation is either horizontal or vertical, then the threshold level T_2, which is applied to a larger 5x5 image region, is used as a decisive check. If the orientation is diagonal, then T_3, which is also applied to a 5x5 image region, is used as a decisive check. All detected valleys are recorded. The algorithm for the implementation of the operator is listed in Figure 5.14.

The implemented operator was used to segment the test image with various threshold levels T_1 and T_2. Three such results are illustrated in the Figures 5.15, 5.16 and 5.17, which show that the selection of suitable threshold levels T_1 and T_2 is achieved by trial-and-error. In Figure 5.15, 29045 valley points were detected, in Figure 5.16, 11119 and in Figure 5.17, 19822. From the results, it can be seen that the best performance is achieved by setting T_1 to a small value so that all possible valley points are detected, and then to eliminate non-valley points, by setting T_2 to an arbitrarily larger value.

```
read T1, T2, and calculate T3 = round(T2·√2)

if ( (l-m)>T1 or (n-m)>T1 ) then              /* Check for a vertical valley (N-S) through m    */
begin
  if (f+k+p+j+o+t - 2(h+m+r)) > T2
     and (g+l+q+i+n+s - 2(h+m+r)) > (h+m+r+f+k+p - 2(g+l+q))
     and (g+l+q+i+n+s - 2(h+m+r)) > (h+m+r+j+o+t - 2(i+n+s))
  then there is a valley through m
end
  else if ( (h-m)>T1 or (r-m)>T1 ) then       /* Check for a horizontal valley (E-W) through m  */
  begin
    if (b+c+d+v+w+x - 2(l+m+n)) > T2
       and (g+h+i+q+r+s - 2(l+m+n)) > (l+m+n+b+c+d - 2(g+h+i))
       and (g+h+i+q+r+s - 2(l+m+n)) > (l+m+n+v+w+x - 2(q+r+s))
    then there is a valley through m
  end
    else if ( (g-m)>T1 or (s-m)>T1 ) then     /* Check for a diagonal valley (NE-SW) through m  */
    begin
      if (k+g+c+w+s+o - 2(q+m+i)) > T3
         and (p+l+h+r+n+j - 2(q+m+i)) > (q+m+i+k+g+c - 2(p+l+h))
         and (p+l+h+r+n+j - 2(q+m+i)) > (q+m+i+w+s+o - 2(r+n+j))
      then there is a valley through m
    end
      else if ( (i-m)>T1 or (q-m)>T1 ) then   /* Check for a diagonal valley (SE-NW) through m  */
      begin
        if (k+q+w+c+i+o - 2(g+m+s)) > T3
           and (l+r+x+b+h+n - 2(g+m+s)) > (g+m+s+k+q+w - 2(l+r+x))
           and (l+r+x+b+h+n - 2(g+m+s)) > (g+m+s+c+i+o - 2(b+h+n))
        then there is a valley through m
      end
```

Figure 5.14 : Algorithm for the implementation of the 5x5 valley operator.

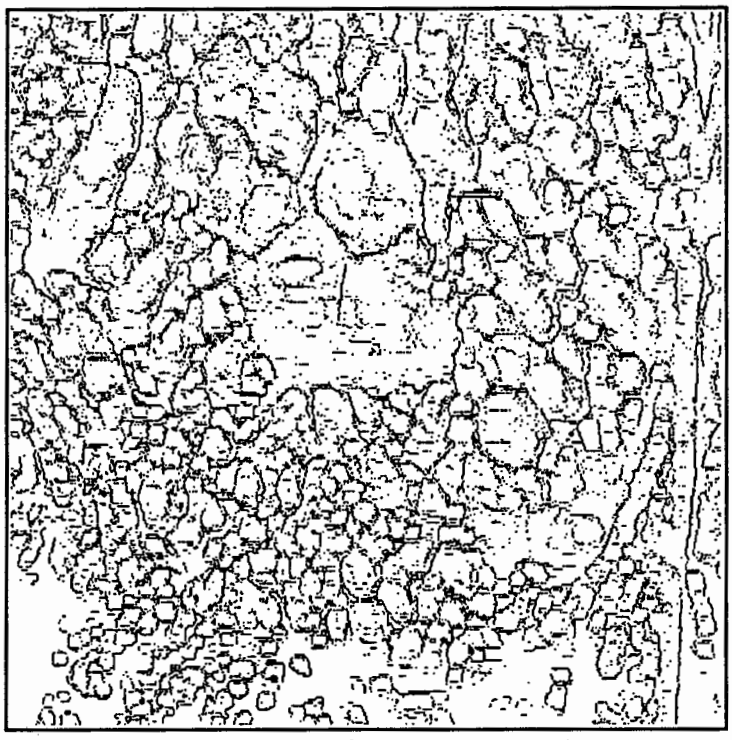


Figure 5.15 : Segmentation of BUB1.CFI using the 5x5 valley operator with T_1=5 and T_2=20.

Figure 5.16 : Segmentation of BUB1.CFI using the 5x5 valley operator with $T_1=5$ and $T_2=55$.

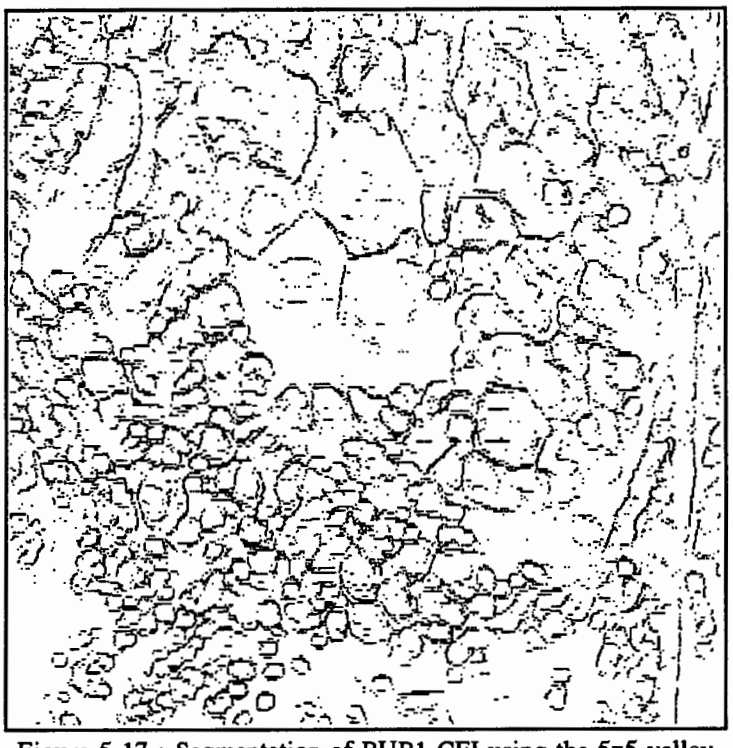


Figure 5.17 : Segmentation of BUB1.CFI using the 5x5 valley operator with $T_1=15$ and $T_2=20$.

5.3 DISCUSSION

The results obtained from the application of classical segmentation techniques, were in the most part discouraging and highlighted the complexity and difficulty of the segmentation problem.

However, the results obtained using the 5x5 valley operator indicated that the best way to go was to segment the images using the valleys (bubble boundaries). A study of intensity profiles from various other images showed that all boundaries were characteristically the same and it was realized that by moving a suitably shaped probe across the top of the image, it may be possible to extract the valley information. This realization motivated an investigation into morphological image processing techniques.

CHAPTER 6

THE MORPHOLOGICAL SEGMENTATION OF SURFACE FROTH IMAGES

"Morphology : study of the forms of things, especially (Biol.) study of the form of animals and plants, (Philol.) study of the form of words."

The Concise Oxford Dictionary, Seventh Edition, 1982.

Mathematical morphology is an approach to image processing that uses shape for the extraction of image features. Unlike classical image processing techniques that have traditionally been based on the concepts and theory of linear systems and which typically focus on the grey level intensities of image pixels, morphological operators are non-linear and use the language of set theory to analyze the geometrical structure within an image (Maragos and Schafer, 1990).

Object identification, extraction and defects correlate directly with shape and morphological operators therefore tend to be the natural processing approach to real images resulting in simplified images where the intrinsic shape characteristics have been preserved and all irrelevant information has been eliminated. It is not surprising therefore, that morphological techniques have become widely used in machine vision and automated industrial inspection

systems.

6.1 MORPHOLOGICAL IMAGE PROCESSING

Morphological image processing can be thought of as the interaction of two different images; image X which is the raw image and image B which is known as the *structuring element* (SE). The SE non-linearly transforms the raw image and can be considered as a probe that is used to extract various object features according to their shape, with the size and shape of the probe depending on the application. In essence, the SE of a morphologic operator has a role analogous to the role of the kernel in a convolution operation.

Two fundamental operations (transforms) exist, that form the building blocks for all morphological processing (Maragos and Schafer, 1990). They are :

- DILATION of X by B.
- EROSION of X by B.

All other morphological operations are defined as combinations of erosions and dilations, such as the following two important operations :

- CLOSING of X by B.
- OPENING of X by B.

The concepts of these morphological operations are illustrated in Figures 6.1 (b), (c), (d) and (e) and their mathematical definitions follow.

Numerous authors have covered the general theory on morphological dilations and erosions (Serra, 1986), (Haralick, Sternberg and Zhuang, 1987) and (Heijmans and Ronse, 1990) and it will suffice here to quote the definitions given by Haralick, Sternberg and Zhuang (1987).

If E^N represents N-dimensional Euclidean space; x, b are elements of X, B where $x = \{x_1, \ldots, x_N\}$ and $b = \{b_1, \ldots, b_N\}$ are N-tuples of element coordinates, then :

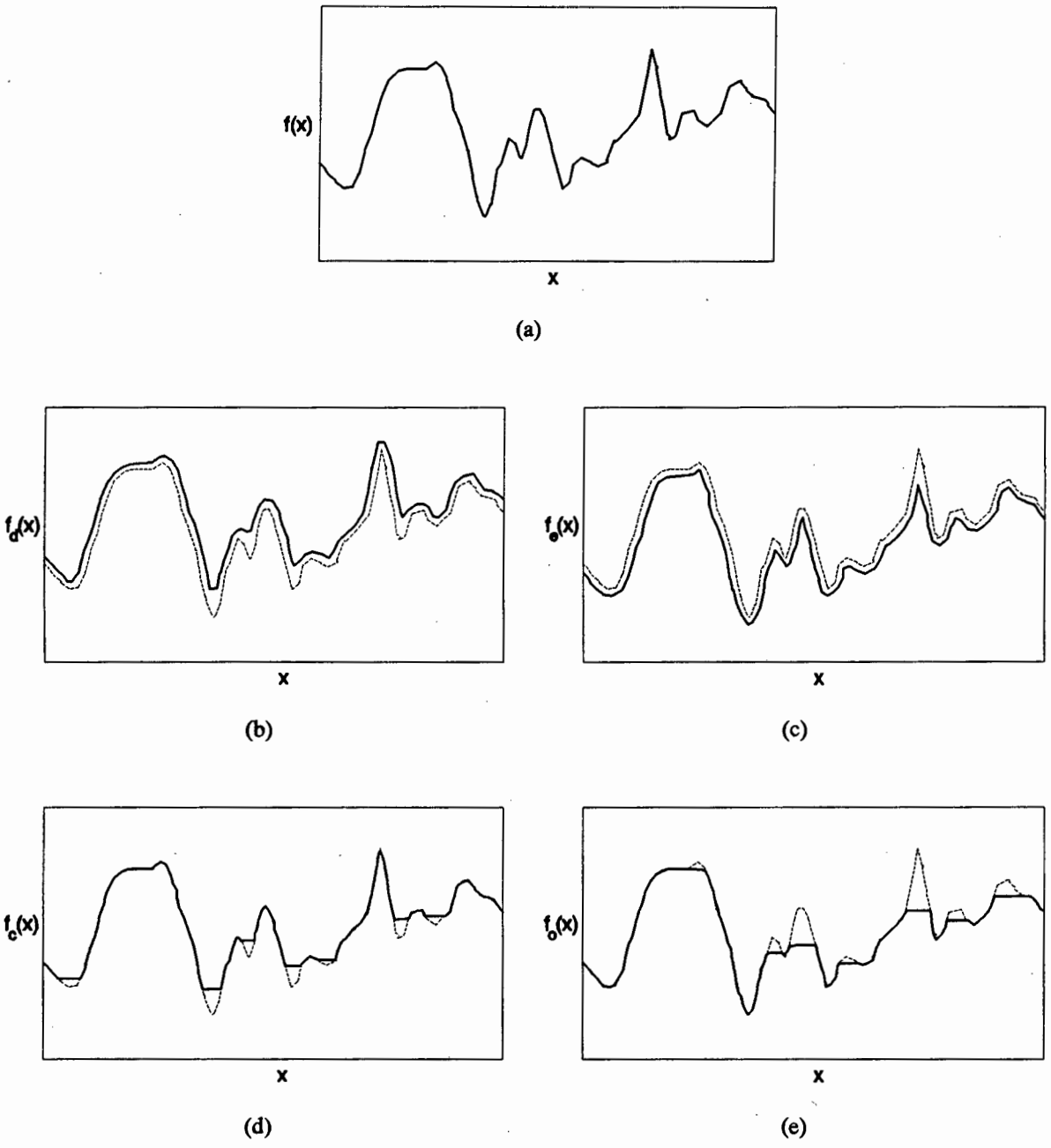


Figure 6.1 : An illustration of the concept of morphological operations, using a 1-D signal. (a) Original signal f(x), (b) dilated signal $f_d(x)$, (c) eroded signal $f_e(x)$, (d) closed signal $f_c(x)$ and (e) opened signal $f_o(x)$.

Definition 6.1 : Let X and B be subsets of E^N. The DILATION of X by B is denoted by $X \oplus B$ and is defined by :

$$X \oplus B = \{c \in E^N \mid c = x + b \text{ for some } x \in X \text{ and } b \in B\}$$

That is, the output of a dilation operator is the set of points c, computed as the union of the translations of X by the elements of B :

$$X \oplus B = \bigcup_{b \in B} (X)_b$$

Where $(X)_b$, denotes the translation of X by b and is defined by :

$$(X)_b = \{c \in E^N \mid c = x + b \text{ for some } x \in X\}$$

Definition 6.2 : Let X and B be subsets of E^N. The EROSION of X by B is denoted by $X \ominus B$ and is defined by :

$$X \ominus B = \{d \in E^N \mid d + b \in X \text{ for every } b \in B\}$$

or expressed as a difference of elements x and b :

$$X \ominus B = \{d \in E^N \mid \text{for every } b \in B, \text{ there exists an } x \in X \text{ such that } d = x - b\}$$

That is, the output of an erosion operator is the set of points d, computed as the intersection of the translations of X by the points -b, where $b \in B$:

$$X \ominus B = \bigcap_{b \in B} (X)_{-b}$$

Definition 6.1 is also known as Minkowski (or vector) addition and its inverse, definition 6.2, as Minkowski (or vector) subtraction (Ghosh, 1988). Using these fundamental operations, morphological closing and morphological opening are defined as :

Definition 6.3 : The morphological CLOSING of an image X by a SE B is denoted by $X \bullet B$ and is defined by :

$$X \bullet B = (X \oplus B) \ominus B$$

Definition 6.4 : The morphological OPENING of an image X by a SE B is denoted by $X \circ B$ and is defined by :

$$X \circ B = (X \ominus B) \oplus B$$

6.1.1 BINARY MORPHOLOGY

Binary images are typified by black (0) and white (1), and the set of all white pixels and the set of all black pixels in a binary image constitute a complete description of the image. In binary space, definitions 6.1 and 6.2 and consequently 6.3 and 6.4 are implemented using the boolean operations of logical AND ($\wedge$) and logical OR ($\vee$), (Nakagawa and Rosenfeld, 1978).

Binary dilation, which effectively amounts to expanding the 1's or alternatively shrinking the 0's, is achieved by logically ORING the SE data with the image data and conversely, binary erosion, which effectively amounts to expanding the 0's or alternatively shrinking the 1's, is achieved by logically ANDING the SE data with the image data. Figure 6.2, illustrates these binary morphological operations using a 3x3 square SE.

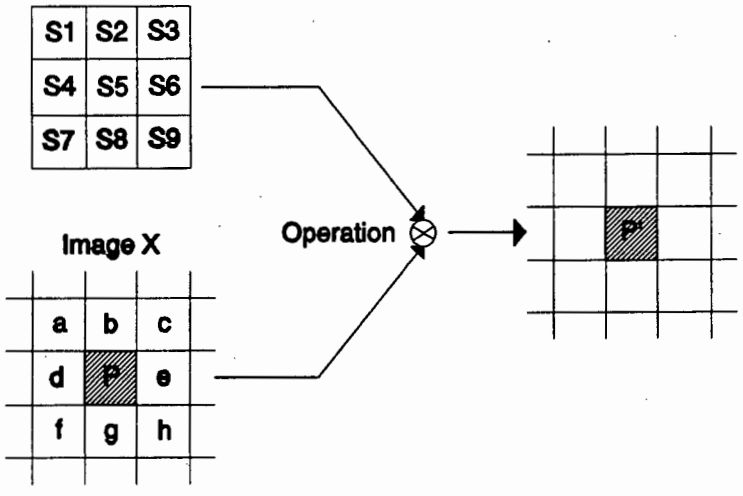


Figure 6.2 : Binary dilation or erosion, using a 3x3 square SE.

If the operation is binary dilation, then P' is given by :

$$P' = (S1 \vee a) \vee (S2 \vee b) \vee (S3 \vee c) \vee (S4 \vee d) \vee (S5 \vee P) \vee (S6 \vee e) \vee (S7 \vee f) \vee (S8 \vee g) \vee (S9 \vee h)$$

If the operation is binary erosion, then P' is given by :

$$P' = (S1 \wedge a) \wedge (S2 \wedge b) \wedge (S3 \wedge c) \wedge (S4 \wedge d) \wedge (S5 \wedge P) \wedge (S6 \wedge e) \wedge (S7 \wedge f) \wedge (S8 \wedge g) \wedge (S9 \wedge h)$$

Huang, Jenkins and Sawchuk (1989), have extended the mathematical concepts of binary morphological operations into an algebra called Binary Image Algebra (BIA) for the development of a formal programming language for a specific parallel architecture known as the Digital Optical Cellular Image Processor (DOCIP).

6.1.2 GREY LEVEL MORPHOLOGY

For grey level images, definitions 6.1 and 6.2 are implemented using maximum and minimum operators respectively, as the local max operator is equivalent to dilation and the local min operator to erosion (Nakagawa and Rosenfeld ,1978). As such, binary dilations, erosions, closings and openings are naturally extended to grey level images, where grey level morphological filters operate on the umbra (U) of an image, in a local region (Sternberg, 1986), (Haralick, Sternberg and Zhuang, 1987) and (Heijmans, 1991). The umbra of an image is the set of points on or below the graph of the function and for a function f(x) the umbra U(f) is given by :

$$U(f) = \{(x,a): a \leq f(x)\}$$

illustrated by the shaded region in Figure 6.3.

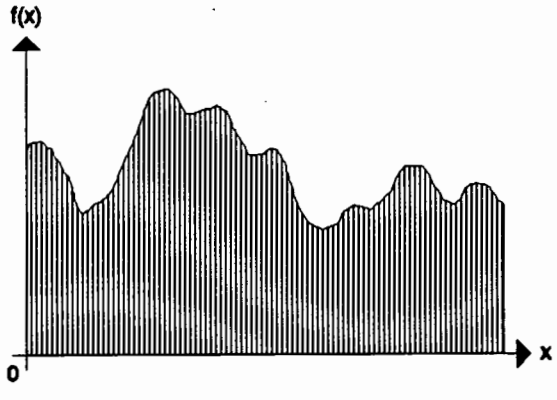


Figure 6.3 : The umbra U(f) for a 1-D function f(x).

The function f(x) can be reconstructed from its umbra since :

$$f(x) = \max \{a:(x,a) \in U(f)\}, \quad \forall x$$

Thus for an image f(x,y) and SE s(i,j), the grey level dilation d(x,y) of f by s is given by (Sternberg, 1986) :

$$d(x,y) = \max_{i,j} [\, f(x-i,y-j) + s(i,j) \,] \qquad \ldots(6.1)$$

Conversely, the grey level erosion e(x,y) of f by s is given by (Sternberg, 1986) :

$$e(x,y) = \min_{i,j} [\, f(x-i,y-j) - s(-i,-j) \,] \qquad \ldots(6.2)$$

These min, max expressions are similar to the convolution expression that occurs frequently in image processing :

$$f * s = \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{n}{2}}^{\frac{n}{2}} f(x-i,y-j) \cdot s(i,j) \qquad \ldots(6.3)$$

It can be seen however, that the sums and differences of equations (6.1) and (6.2), replace the multiplication of equation (6.3) and the max and min operators of (6.1) and (6.2) replace the convolution sum of (6.3).

6.1.3 THE AFFECT OF MORPHOLOGICAL OPERATIONS ON IMAGES

Figure 6.4, illustrates the affect of (b) dilating, (c) eroding, (d) opening and (e) closing an image X with a square structuring element B. Note the regions that have been numerically labelled in (a) and their transformations after opening and closing the image. It can be seen that the opening of an image eliminates small islands (1), removes narrow necks of land (3) and narrow bridges (2). Whereas closing connects closely separated regions (4), blocks long narrow channels (6), and fills in small "lakes" (5,7).

If the SE has a regular shape, both closing and opening of the image can be thought of as non-linear filters which smooth the image contours, with the shape and size of the SE determining the nature and degree of smoothing.

6.2 MORPHOLOGIC EDGE DETECTION

Grey level morphologic operations can be used to perform edge detection, where the simplest method is given by the difference between a raw image and the image after it has been morphologically dilated or eroded with a SE (Lee, Haralick and Shapiro, 1987). This method of edge detection is referred to as residual edge detection.

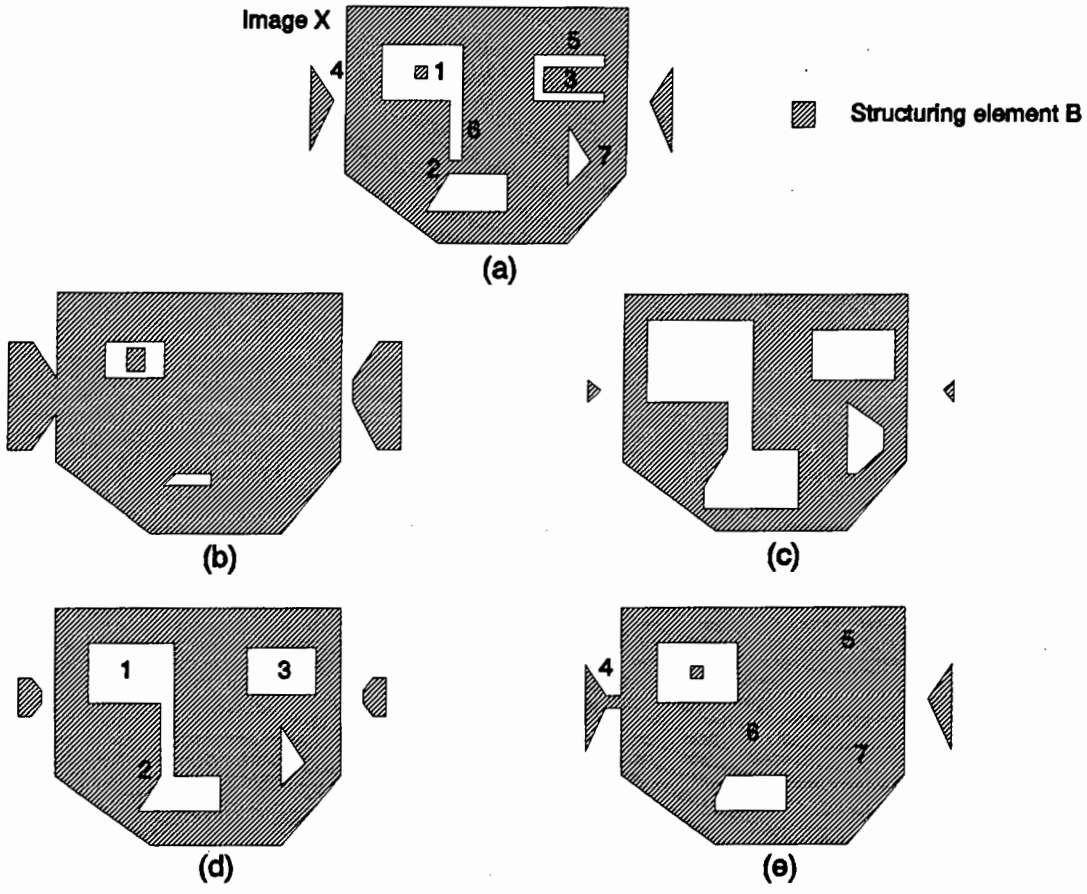


Figure 6.4 : Some morphological operations on an image; (a) original image (b) dilated, (c) eroded, (d) opened and (e) closed images.

If f(x,y) represents the raw image, e(x,y) the eroded image and d(x,y) the dilated image, the erosion residue edge detector denoted by $G_e(x,y)$, is defined as :

$$G_e(x,y) = f(x,y) - e(x,y)$$

and the dilation residue edge detector denoted by $G_d(x,y)$, is defined as :

$$G_d(x,y) = d(x,y) - f(x,y)$$

The dilation and erosion residue edge detectors do not perform any form of averaging of the raw image and they are therefore non-linear Laplacian-like operators that are noise sensitive and are therefore in general not good edge detectors for noisy images.

The concept of residue edge detection is easily extended to cover images that have undergone morphological openings and closings. If f(x,y) is again representative of the raw image, o(x,y) the opened image and c(x,y) the closed image, the opened residue edge detector

denoted by $G_o(x,y)$, is defined as :

$$G_o(x,y) = f(x,y) - o(x,y)$$

and the closed residue edge detector denoted by $G_c(x,y)$, is defined as :

$$G_c(x,y) = c(x,y) - f(x,y)$$

Because of the nature of their operations, openings and closings offer an intuitively simple and mathematically formal way for peak or valley detection (Maragos and Schafer, 1990). The shape and size of the SE used, as previously mentioned, determines the effectiveness with which the opened residue edge detector $G_o(x,y)$ extracts peaks or with which the closed residue edge detector $G_c(x,y)$ extracts valleys. Due to its uniform shape, formed by surfaces of all possible orientations, the sphere is a very popular SE used for residual edge detection and the opening or closing of an image using a spherical SE is commonly known as the "Rolling Ball Algorithm".

6.2.1 ROLLING BALL ALGORITHM

The closing of a grey level image f(x,y) by a spherical SE s(i,j) of radius R, can be visualised in terms of the closing of the umbra U(f) by s(i,j) (Sternberg, 1986). Conceptually this is achieved by placing a ball on the surface of the image, Figure 6.5, and rolling it along the top.

The ball accurately follows smooth surface variations and surface peaks, but not deep crevices and narrow surface pits, that are less than the diameter of the ball. This means that the new surface, which is formed by the locus of all points of contact between the ball and original surface, is free of such crevices. Subtracting the original image from the closed image to obtain the residue $G_c(x,y)$ yields the valleys or low intensity regions in the image.

Conversely, the opening of a grey level image with a spherical SE can be thought of as pushing the ball up against the underside of the surface. As the ball traverses the surface, it follows the smooth surface variations, crevices and pits, but will not be able to go up peaks that are of diameter less than the ball. The residue $G_o(x,y)$, obtained by subtracting the opened image from the original image, yields the peaks or high intensity regions in the image.

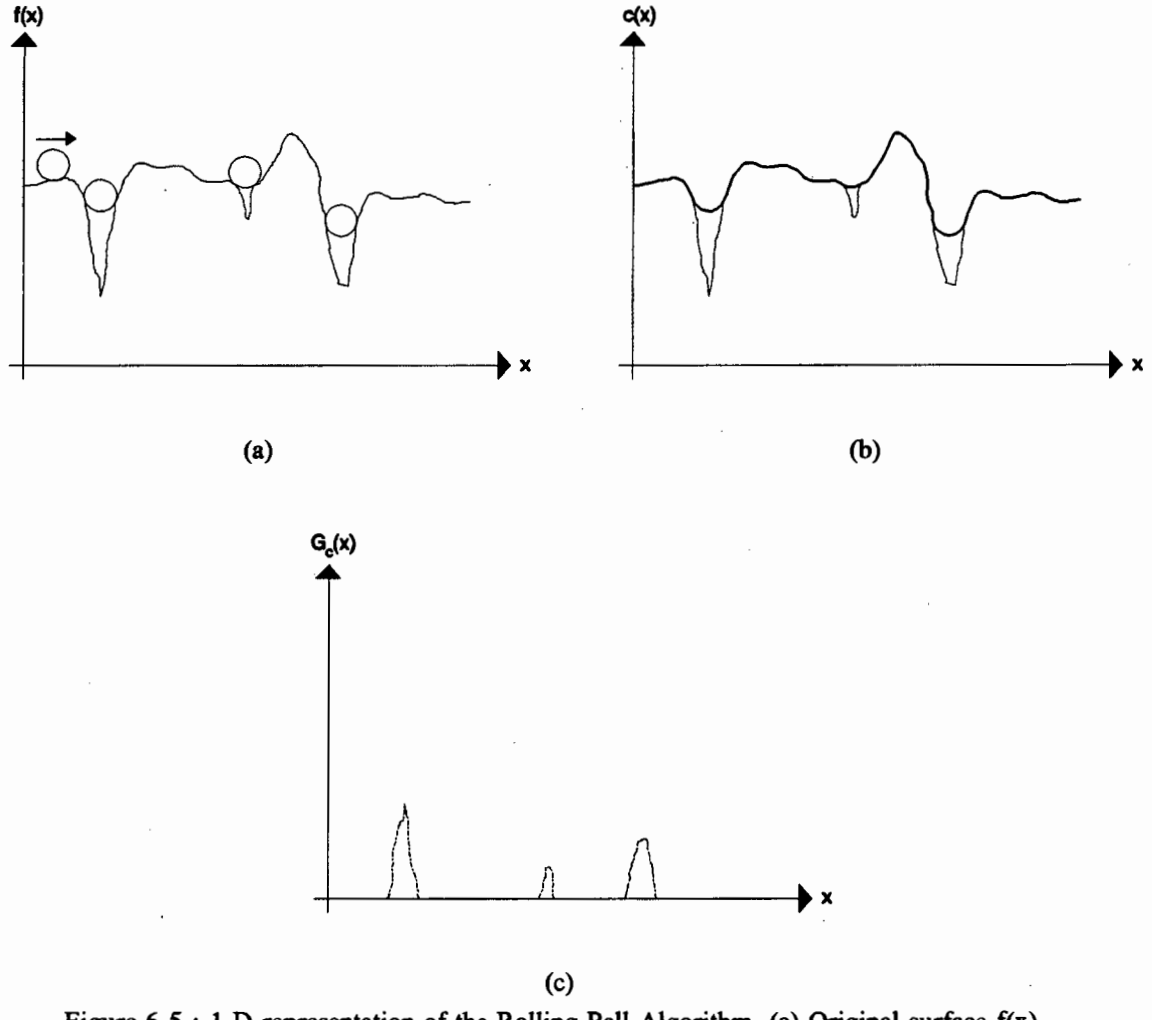


Figure 6.5 : 1-D representation of the Rolling Ball Algorithm. (a) Original surface f(x), (b) spherically closed surface c(x), and (c) residual surface $G_c(x)$.

As can be appreciated, the size of the spherical SE used greatly influences the degree to which the surface crevices or surface peaks are detected. Intuitively, a larger ball will provide a better detection than a smaller ball, with a particular size providing optimal detection.

It has been shown (Chapter 5, Figure 5.2), that bubble highlights are represented by intensity peaks and that, except for regions of excessive shadow, bubble boundaries are represented by deep, narrow crevices and intuitively, a suitably sized spherical SE should be able to extract either one of these features. Consequently, the Rolling Ball Algorithm was implemented in software to investigate its application to the segmentation of surface froth structures.

6.2.2 ROLLING BALL IMPLEMENTATION

In the software implementation of the algorithm, a hemisphere is used for opening an image, and an inverted hemisphere for closing an image. For a hemisphere of radius R, the data points for the discrete spherical SE (DSSE) are calculated using the equation :

$$s(i,j) = \sqrt{R^2 - i^2 - j^2} \qquad \ldots(6.4)$$

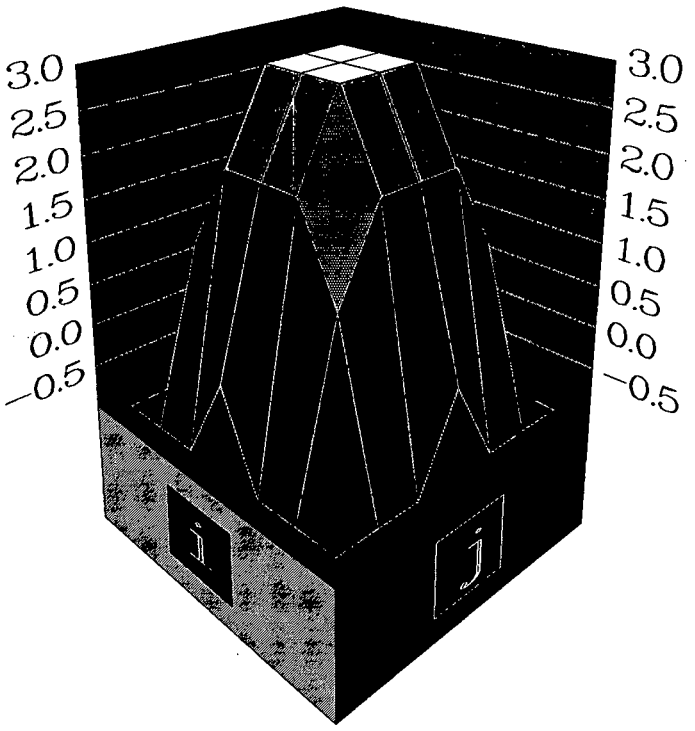
$$\textit{Where}: \quad -R \le i \le R \quad , \quad -R \le j \le R$$

For ease of implementation and computing optimization, signed 8 bit (byte) data types, having a range -128 to 127, were used; with DSSE points less than zero being set to -1, to identify them as points to be ignored during processing. The pseudo code used to generate DSSEs of radius R, is listed in Figure 6.6, and Figures 6.7 and 6.8 are examples of two DSSEs of radii 3 pixels and 5 pixels respectively, that were generated using the code.

```
for (i=-R;i<=R;i++)
   for (j=-R;j<=R;j++)
      if ( (R^2 - i^2 - j^2) >= 0)
         s(i,j) = (char) ROUND( SQRT(R^2 - i^2 - j^2) )
      else
         s(i,j) = -1
```

Figure 6.6 : Pseudo code for generating DSSEs of radius R pixels.

The grey level openings and closings were implemented in software using the expressions given in equations 6.1 and 6.2, for the fundamental operations of grey level dilation and grey level erosion. The pseudo code for the implementation of grey level dilation is listed in Figure 6.9 and that for grey level erosion in Figure 6.10. Because of the size of the images processed (512x512 pixels) and the number of computations that have to be performed for increasing SE size, all images that were processed were done so off-line, because the real-time morphological processing of surface froth images in software was not possible with the available computer architecture.

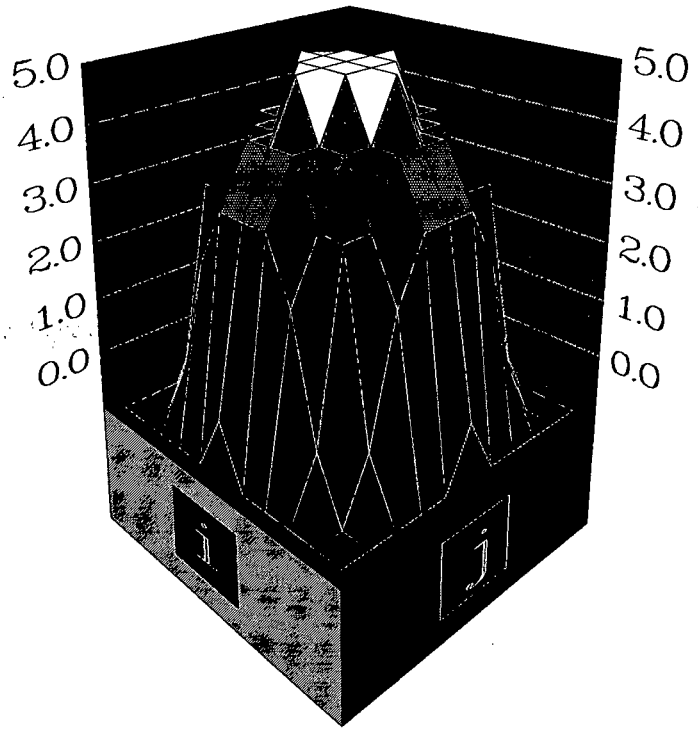


(a)

$$s(i,j) = \begin{bmatrix} -1 & -1 & -1 & 0 & -1 & -1 & -1 \\ -1 & 1 & 2 & 2 & 2 & 1 & -1 \\ -1 & 2 & 3 & 3 & 3 & 2 & -1 \\ 0 & 2 & 3 & 3 & 3 & 2 & 0 \\ -1 & 2 & 3 & 3 & 3 & 2 & -1 \\ -1 & 1 & 2 & 2 & 2 & 1 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 \end{bmatrix}$$

(b)

Figure 6.7 : DSSE of radius 3 pixels.
(a) 3-D surface plot.
(b) numerical values.



(a)

$$s(i,j) = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & 2 & 3 & 3 & 3 & 2 & 0 & -1 & -1 \\ -1 & 0 & 3 & 3 & 4 & 4 & 4 & 3 & 3 & 0 & -1 \\ -1 & 2 & 3 & 4 & 4 & 5 & 4 & 4 & 3 & 2 & -1 \\ -1 & 3 & 4 & 4 & 5 & 5 & 5 & 4 & 4 & 3 & -1 \\ 0 & 3 & 4 & 5 & 5 & 5 & 5 & 5 & 4 & 3 & 0 \\ -1 & 3 & 4 & 4 & 5 & 5 & 5 & 4 & 4 & 3 & -1 \\ -1 & 2 & 3 & 4 & 4 & 5 & 4 & 4 & 3 & 2 & -1 \\ -1 & 0 & 3 & 3 & 4 & 4 & 4 & 3 & 3 & 0 & -1 \\ -1 & -1 & 0 & 2 & 3 & 3 & 3 & 2 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

(b)

Figure 6.8 : DSSE of radius 5 pixels.
(a) 3-D surface plot.
(b) numerical values.

```
for (row=R;row<512-R;row++)
   for (col=R;col<512-R;col++)
   begin
     max = 0;                                  /* unsigned 8 bit data type for holding the maximum value */
     for (i=-R;i<=R;i++)
        for (j=-R;j<=R;j++)
        begin
            if ( s(i,j) >= 0 ) then
            begin
               temp = f(row+i,col+j) + s(i,j);              /* temp is a 16 bit signed data type */
               if (temp > 255) then
                  temp = 255;                                          /* watch for overflow */
               if (temp > (int) max) then
                  max = (unsigned char) temp;
            end
        end
        d(row,col) = max;                                              /* dilated image */
   end
```

Figure 6.9 : Pseudo code for the implementation of grey level dilation.

```
for (row=R;row<512-R;row++)
   for (col=R;col<512-R;col++)
   begin
     min = 0;                                  /* unsigned 8 bit data type for holding the minimum value */
     for (i=-R;i<=R;i++)
        for (j=-R;j<=R;j++)
        begin
            if ( s(i,j) >= 0 ) then
            begin
               temp = f(row+i,col+j) - s(i,j);              /* temp is a signed 16 bit data type */
               if (temp < 0) then
                  temp = 0;                                          /* watch for underflow */
               if (temp < (int) min) then
                  min = (unsigned char) temp;
            end
        end
        e(row,col) = min;                                              /* eroded image */
   end
```

Figure 6.10 : Pseudo code for the implementation of grey level erosion.

Initially surface froth structures were processed on the 80386/SX clone using byte arithmetic and the MVP-AT memory for the storage of intermediate processing stages. For reasons discussed above, the run-time of the segmentation technique on the 16 MHz machine was unacceptable for DSSEs of increasing size (see Figure 6.11) and the work was transferred to the SUN workstation where the run-times obtained were considerably better.

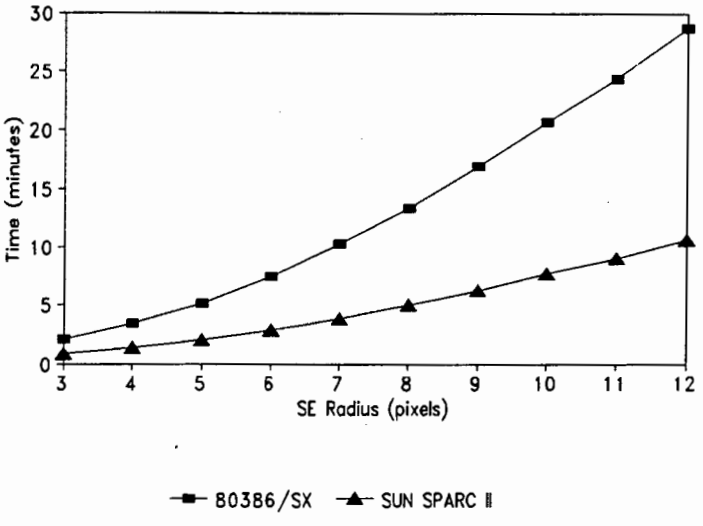


Figure 6.11 : Run-times of morphological closing using DSSEs, on two computer architectures.

6.3 SURFACE FROTH SEGMENTATION USING MORPHOLOGIC EDGE DETECTION

In an attempt to extract bubble highlights and bubble boundaries using residual edge detection, surface froth images were morphologically opened and closed using DSSEs of increasing radius. The edge residues $G_o(x,y)$ and $G_c(x,y)$ for the test image BUB1.CFI, were computed and thresholded, and in order to gauge the effectiveness of the techniques, no preprocessing of the raw image or postprocessing of the segmented image was performed.

6.3.1 BUBBLE HIGHLIGHT EXTRACTION

Prior to morphologically opening the test image, a dc level of 20 was added to every pixel in the image, to eliminate points sitting at 0 (see Figure 5.2) and thus to prevent the unsigned 8 bit image data from wrapping around[1].

---

[1] An illustration of 8 bit unsigned data wrapping around can be presented by subtracting the decimal value 4 from the decimal value 3. In binary this is given by :

```
  00000011
  00000100
  --------
  11111111
  --------
```

This gives the incorrect decimal result of 255.

Figures 6.12, 6.13, 6.14 and 6.15, illustrate the result of residual edge detecting the test image BUB1.CFI, by opening the image with DSSEs of radii 3, 5, 7 and 9 pixels respectively.

Each image has been thresholded with a value twice that of the DSSE radius that was used, to limit noise and the merging of bubble highlights. This threshold level was found by trial-and-error.

For ease of interpretation, the extracted highlights are displayed in black and correspond to all those pixels of intensity greater than or equal to the selected threshold level. As the DSSE size increases, the extracted highlights become more prominent, but start to coalesce. This presents a problem as the highlights lose their individuality and as such, cannot be suitably processed to provide a reliable segmentation of the bubbles.

From the results presented, it is therefore possible to conclude that although the highlights are extractable to an extent, they are not the optimal feature to use for the segmentation of surface froth structures.

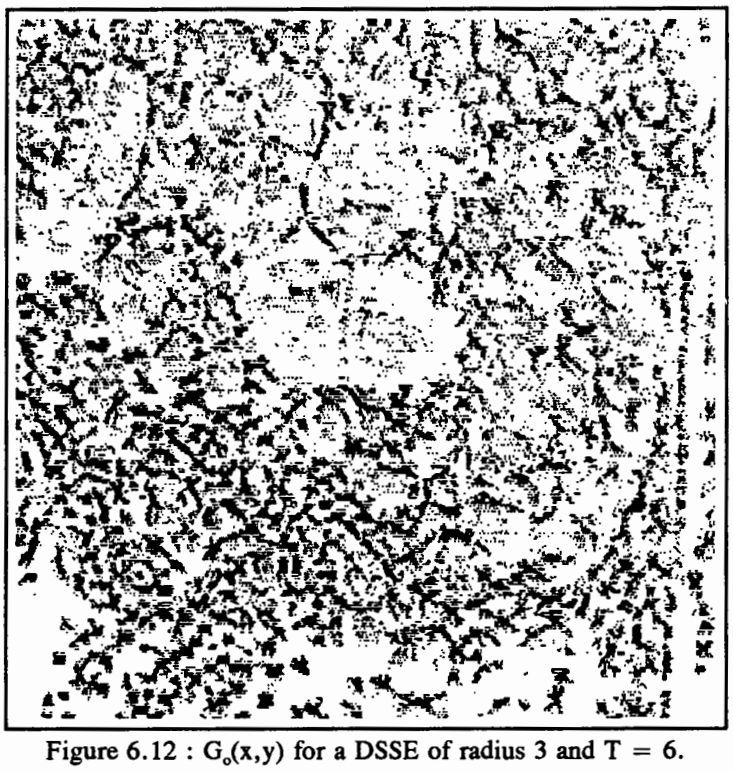


Figure 6.12 : $G_o(x,y)$ for a DSSE of radius 3 and T = 6.

Figure 6.13 : $G_o(x,y)$ for a DSSE of radius 5 and T = 10.

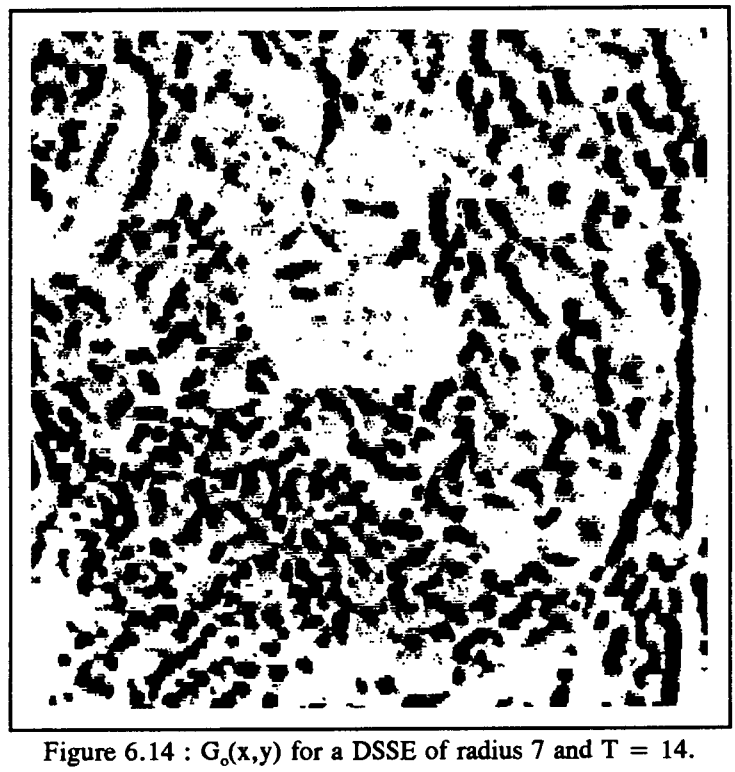


Figure 6.14 : $G_o(x,y)$ for a DSSE of radius 7 and T = 14.

Figure 6.15 : $G_o(x,y)$ for a DSSE of radius 9 and T = 18.

6.3.2 BUBBLE BOUNDARY EXTRACTION

Figures 6.16, 6.17, 6.18 and 6.19, illustrate the result of residual edge detecting the test image BUB1.CFI, by closing the image with DSSEs of radii 3, 5, 7 and 9 pixels respectively.

For ease of interpretation, the extracted boundaries are displayed in black. All the residual images were thresholded using a level of 1. As a rule of thumb though, it was found that a threshold level in the range $0 < T < R$ did not result in any substantial loss of information and thus it was only a matter of preference that a threshold level of 1 was chosen. In each of the images the bubble boundaries are clearly discernable features above the horizontal scan noise, with increasing SE size, improving detection. Note the distinct white patches in the images. These correlate with excessive shadow regions in the image (c.f. Figure 5.1) and due to insufficient boundary information in these regions, the algorithm effectively falls down. However, with increasing ball radius it can be seen that some boundary information is extracted from these regions.

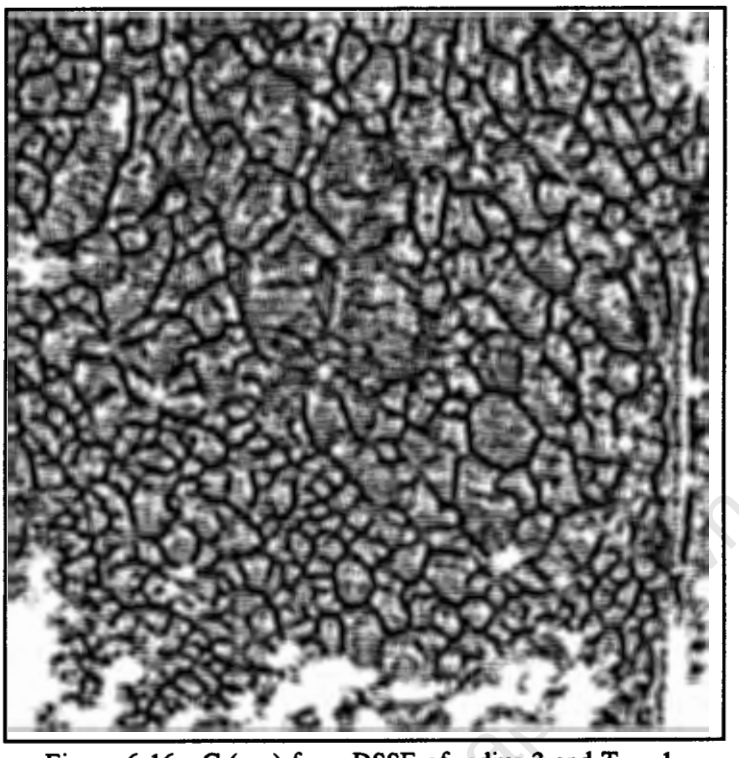


Figure 6.16 : $G_e(x,y)$ for a DSSE of radius 3 and T = 1.

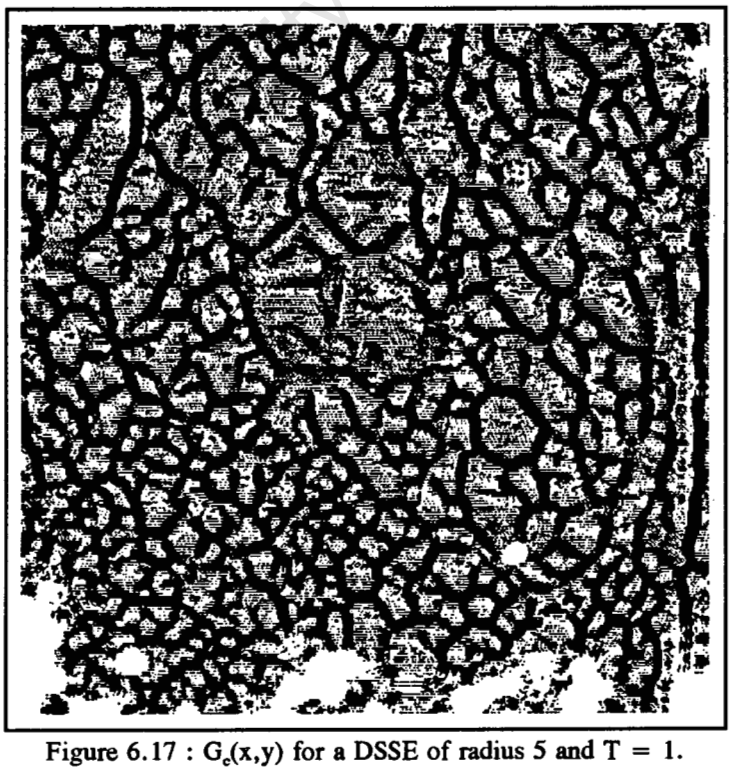


Figure 6.17 : $G_e(x,y)$ for a DSSE of radius 5 and T = 1.

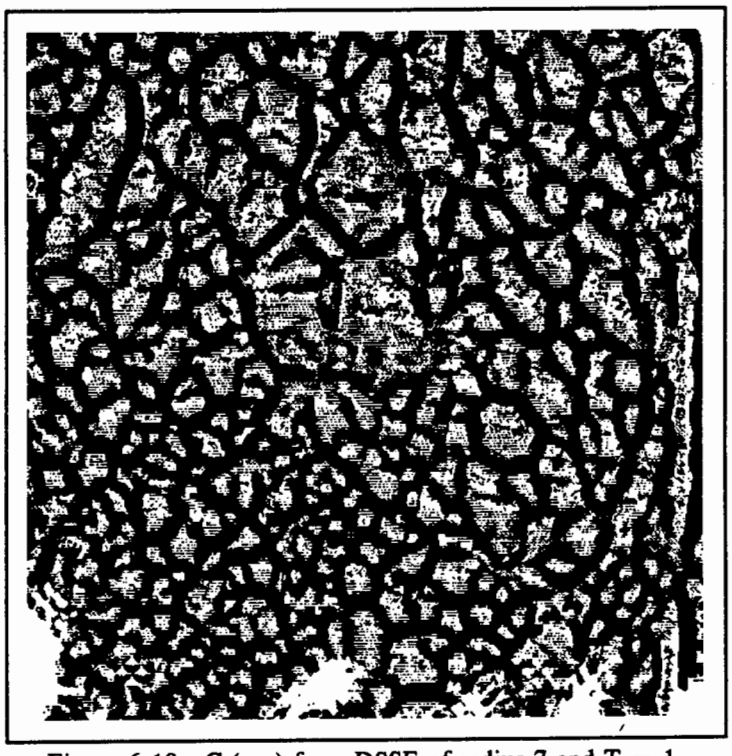


Figure 6.18 : $G_c(x,y)$ for a DSSE of radius 7 and T = 1.

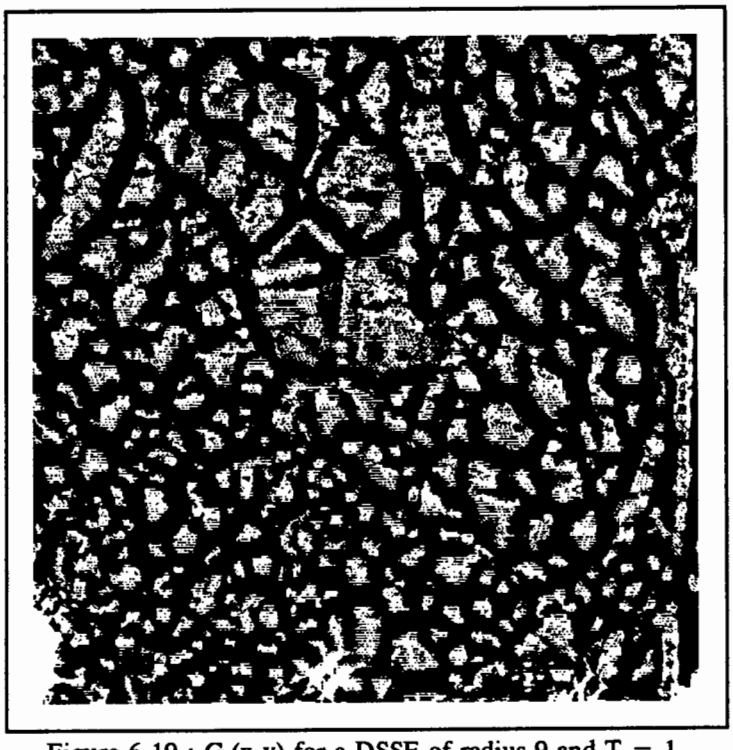


Figure 6.19 : $G_c(x,y)$ for a DSSE of radius 9 and T = 1.

6.4 IMPROVING BUBBLE BOUNDARY SEGMENTATION

The results illustrated in sections 6.3.1 and 6.3.2, substantiate the statement made in Chapter 4, section 4.2, that bubble boundaries are the more stable and reliable feature with which to characterise surface froth structures. The subsequent development of a reliable segmentation technique, using morphological image processing, concentrated on improving the images obtained by extracting the bubble boundaries.

An analysis of the segmented images reveals that the majority of the bubble boundaries, except for those in excessive shadow regions, have been extracted; with the strength of the detection being dependant upon the size of the DSSE used. However, the images show that preprocessing of the raw image and postprocessing of the segmented image is desirable, as :

- There is considerable horizontal scan noise, which should be reduced by suitable preprocessing of the image.
- The segmented images contain an interconnected network of fat lines that are the extracted bubble boundaries. Due to erroneous or incomplete detection, their exist unconnected branches and free islands, which need to be either eliminated or joined up using some form of image postprocessing.

It must be noted though, that complex postprocessing techniques, such as curve joining for unconnected branches, are undesirable due to the added computational complexity. Ideally, the postprocessing should clean up the image quickly and make it "presentable" for an image analysis routine.

6.4.1 IMAGE PREPROCESSING

Two preprocessing methods were investigated; low-pass filtering and median filtering of the image. These methods were applied to Figure 6.19.

(A) LOW-PASS FILTERING

Figure 6.20, illustrates the result of low-pass filtering BUB1.CFI using a 3x3 averaging kernel, prior to morphologically closing the image with a spherical SE of radius 9.

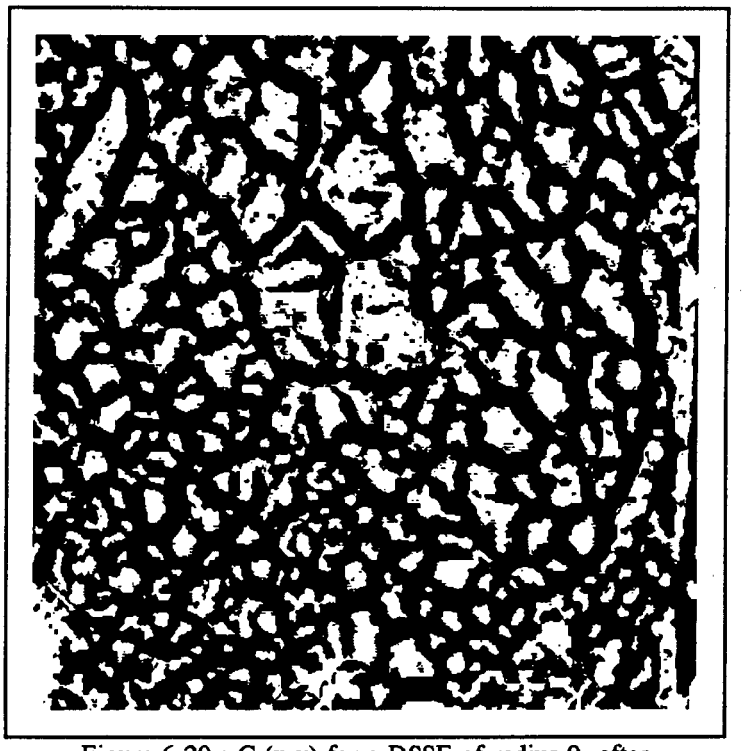


Figure 6.20 : $G_c(x,y)$ for a DSSE of radius 9, after low-pass filtering BUB1.CFI.

(B) MEDIAN FILTERING

Figure 6.21, illustrates the result of median filtering BUB1.CFI using a 3x3 kernel, prior to morphologically closing the image with a spherical SE of radius 9.

(C) DISCUSSION

A comparison of the two images shows that both preprocessing methods investigated essentially produce the same result, with the low-pass filter producing a slightly better end result, that contains less salt and pepper noise. Due to the fact that a 3x3 low-pass filter is easier to implement and is computationally quicker than a median filter, as no sorting of the data is required, it was chosen as the required preprocessor.

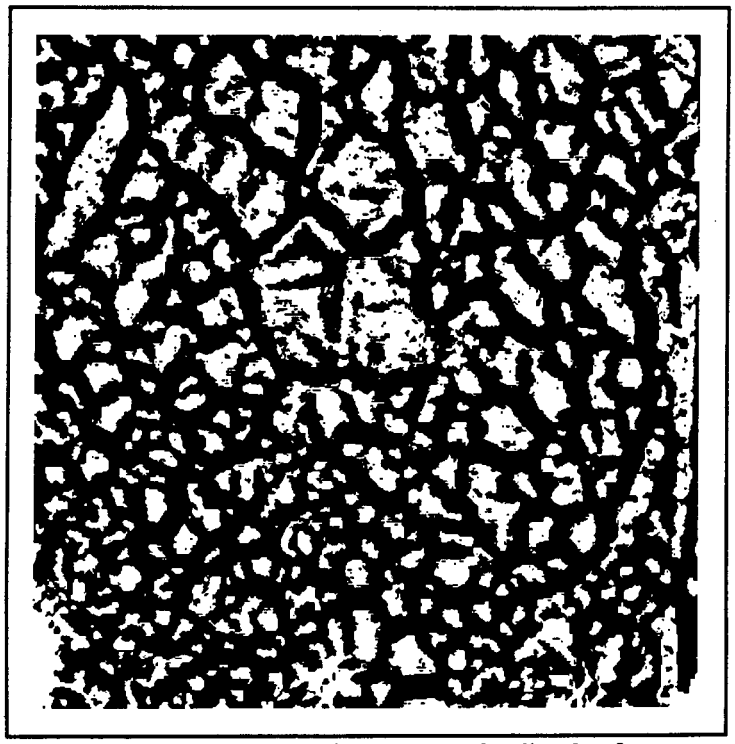


Figure 6.21 : $G_c(x,y)$ for a DSSE of radius 9, after median filtering BUB1.CFI.

6.4.2 IMAGE POSTPROCESSING

An analysis of Figure 6.20, reveals that the postprocessing of segmented images must perform two functions :

- Firstly, it must clean the images by removing the remaining horizontal scan noise, the salt and pepper noise, the small "islands" and if possible join disconnected branches.
- Secondly, in order to achieve a 1:1 correspondence between the segmented and actual bubble boundaries and therefore a direct measure of the bubble size, it is desirable to thin the interconnected network of fat lines, to lines of 1 pixel in width.

(A) CLEANING

Suitable image cleaning was achieved using binary morphological opening and closing, with specifically shaped SEs. However, with binary images, since the size and shape of the SE used greatly influences the "shape" of the processed image, it was decided to use SEs of size no greater than 3x3, to eliminate this potential bias.

To remove the remaining horizontal scan noise, Figure 6.20 was morphologically opened with a 2x1 rectangular SE, the result of which is illustrated in Figure 6.22. Next, to remove the remaining salt and pepper noise and the smaller islands, Figure 6.22 was morphologically opened with a 1x2 rectangular SE, the result of which is illustrated in Figure 6.23.

In order to remove small holes, that would on thinning the thresholded image, be mistakenly identified as bubbles, Figure 6.23 was morphologically closed with a 3x3 plus (+) SE. The result of which is illustrated in Figure 6.24.

These three operations :

- opening the segmented image with a 2x1 SE
- opening the segmented image with a 1x2 SE
- closing the segmented image with a 3x3 cross-hair SE

yielded a suitably clean image, that could then be thinned.

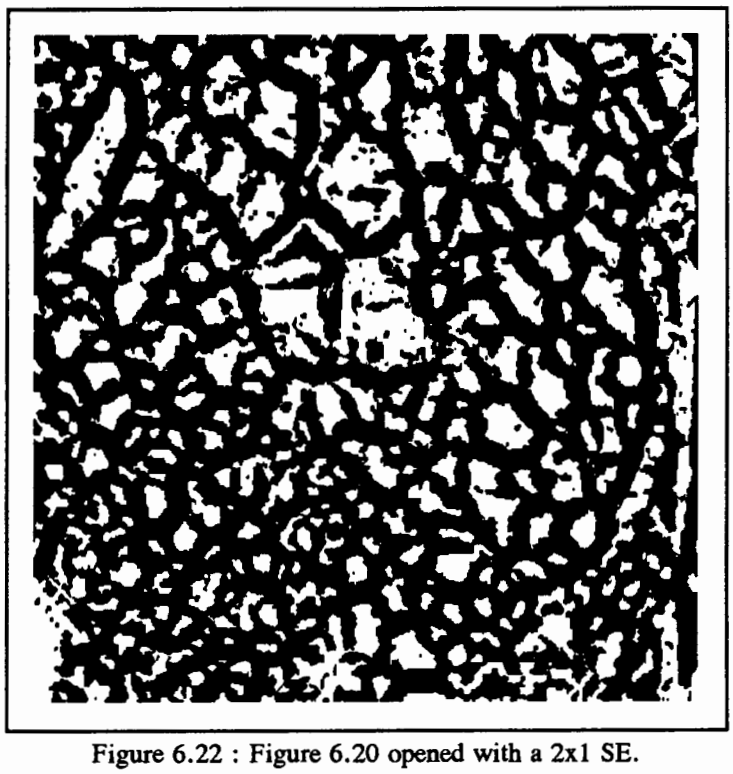


Figure 6.22 : Figure 6.20 opened with a 2x1 SE.

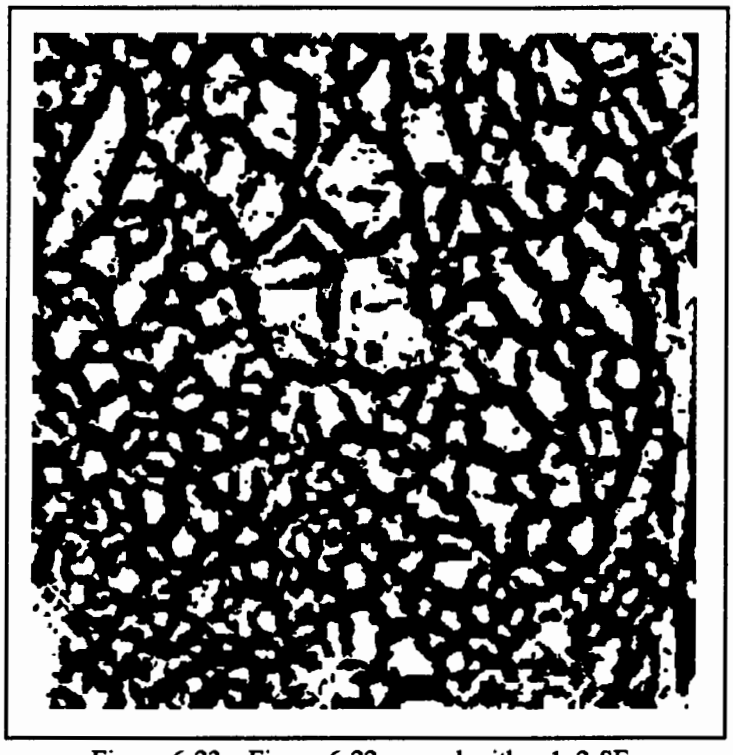


Figure 6.23 : Figure 6.22 opened with a 1x2 SE.

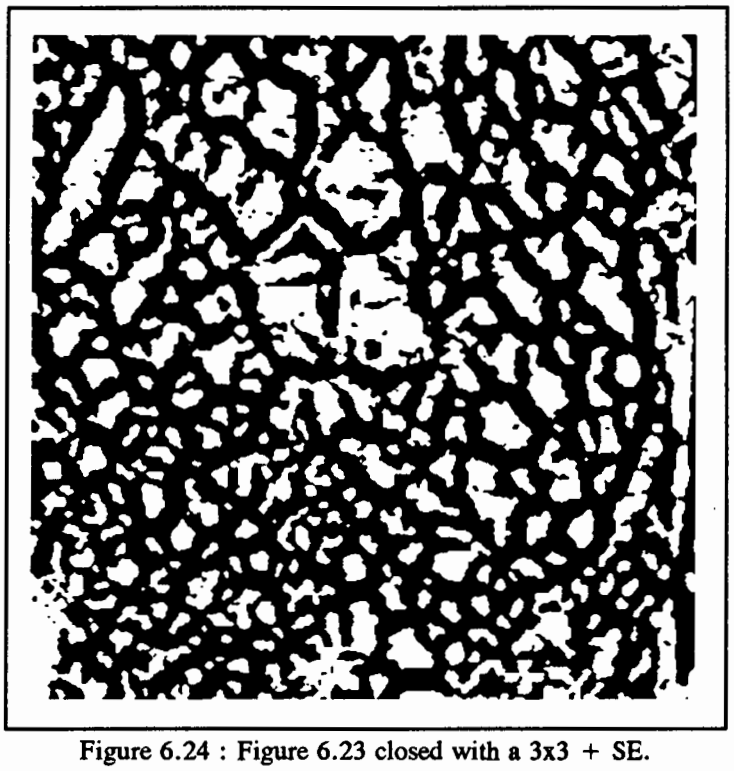


Figure 6.24 : Figure 6.23 closed with a 3x3 + SE.

(B) THINNING

The thinning of segmented surface froth images, not only produces an aesthetically pleasing result that is ideal for an image analysis routine, but it also gives a 1:1 correspondence between actual and segmented bubble boundaries that can be obtained by overlaying the segmented image on the original image, as will be shown later, making it possible to visually ascertain the performance of the segmentation technique.

The algorithm implemented to thin segmented surface froth images, is an iterative method that is discussed by Gonzalez and Wintz (1987, pp. 398-402). Thinning of the image is achieved in two steps, using the 8-connected neighbourhood given in Figure 6.25.

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Figure 6.25 : Neighbourhood arrangement used by the thinning algorithm.

With reference to Figure 6.25, the first step flags a contour point p for deletion, if the following conditions are satisfied :

(a1) $2 \leq N(p_1) \leq 6$

(b1) $S(p_1) = 1$

(c1) p_2 AND p_4 AND $p_6 = 0$

(d1) p_4 AND p_6 AND $p_8 = 0$

where $N(p_1)$ is the number of nonzero neighbours of p_1 :

$$N(p_1) = p_2 + p_3 + ... + p_8 + p_9$$

and $S(p_1)$ is the number of 0 to 1 transitions in the ordered sequence of $p_2, p_3, \ldots, p_8, p_9$.

Condition (a1) is violated when the point p_1 has only one or seven 8-neighbours valued 1. If p_1 has only one neighbour, then it is clearly an end point and should not be deleted and if it has seven neighbours, then deletion of the point would cause undesirable erosion into the region. Condition (b1) is violated when the mask is applied to a stroke one pixel thick and this condition therefore retains connectivity. Conditions (c1) and (d1) are satisfied simultaneously by the following minimum set of values : $p_4 = 0$ or $p_6 = 0$ or ($p_2 = 0$ and

$p_8 = 0$). Thus with reference to Figure 6.25, a point that satisfies conditions (a1), (b1), (c1) and (d1) is an east or south boundary point, or a northwest corner point in the boundary.

The second step flags a contour point p for deletion, if the following conditions are satisfied :

(a2) $2 \leq N(p_1) \leq 6$

(b2) $S(p_1) = 1$

(c2) p_2 AND p_4 AND $p_8 = 0$

(d2) p_2 AND p_6 AND $p_8 = 0$

Conditions (a2) and (b2) are violated as per the discussions for conditions (a1) and (b1). Conditions (c2) and (d2) are satisfied simultaneously by the following minimum set of values : $p_2 = 0$ or $p_8 = 0$ or ($p_4 = 0$ and $p_6 = 0$). A point that satisfies conditions (a2), (b2), (c2) and (d2) is a north or west boundary point, or a southeast corner point.

Using the set of conditions, a single iteration of the thinning algorithm consists of :

- Applying step 1 to flag border points for deletion.
- Deleting the flagged points.
- Applying step 2 to flag the remaining border points for deletion.
- Deleting the flagged points.

This basic procedure is iteratively applied, until no further points are deleted, at which point the algorithm terminates, yielding the thinned borders. Figure 6.26, illustrates the result obtained on applying the algorithm to the segmented image given in Figure 6.24.

(C) DISCUSSION

The morphological closing of surface froth images, provides a reliable but not error-free method for the segmentation of surface froth structures, using bubble boundaries as the extractable feature. The resultant thinned interconnected network of bubble boundaries, Figure 6.26, give rise to distinct regions or "blobs". By overlaying the thinned image on the original image, as illustrated in Figure 6.27, the blobs can be seen to be representative of the individual bubbles and as such the characterisation of the surface can be achieved by counting

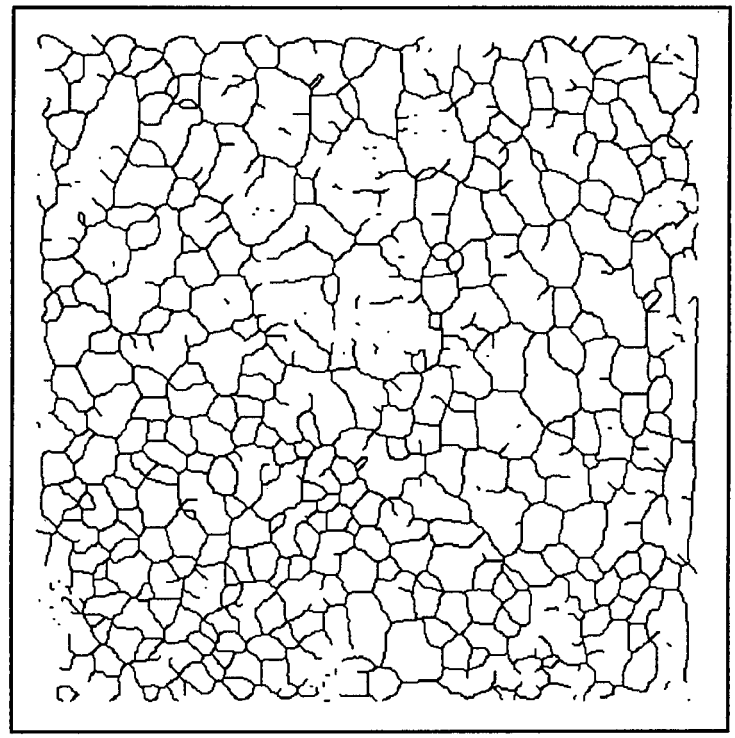


Figure 6.26 : Figure 6.24 thinned to provide the boundaries.

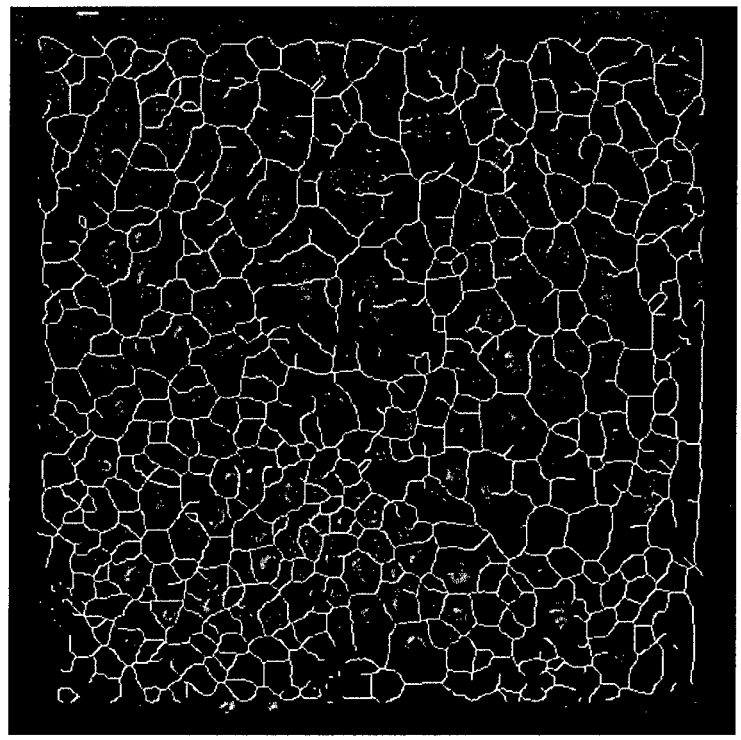


Figure 6.27 : BUB1.CFI - the segmented image overlayed on the original image.

the individual blobs and determining their corresponding size and shape. As will be shown, the best possible segmentation of surface froth images is given by an optimally sized SE, which in practice can be determined automatically.

6.5 THE SEGMENTATION OF SUBOPTIMAL DATA

To test the reliability of the segmentation technique, two worst case images BUB13.CFI and BUB14.CFI were captured from the first video tape that was supplied for the project (Chapter 3, section 3.2.2) and were segmented using DSSEs of radius 9. The resultant segmented images were overlayed on the original images and are given in Figures 6.28 and 6.29.

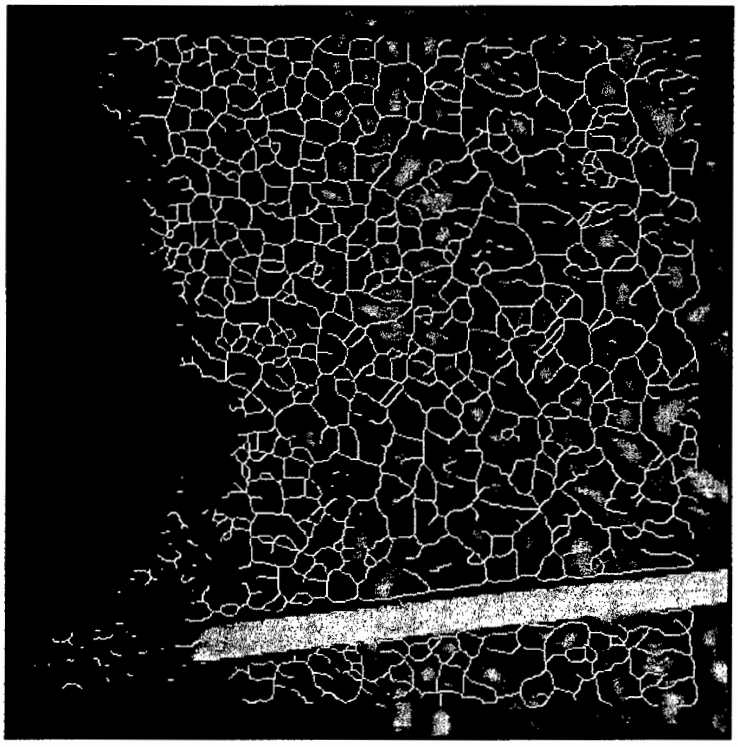


Figure 6.28 : BUB13.CFI - the segmented image overlayed on the original image.

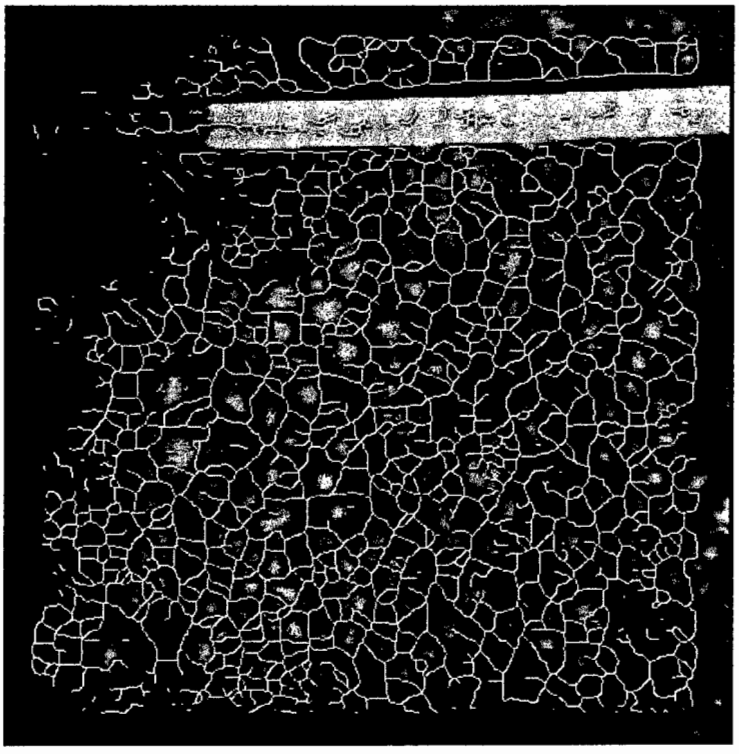


Figure 6.29 : BUB14.CFI - the segmented image overlayed on the original image.

The results show that most of the boundaries have been efficiently extracted. The segmented image is however shifted, this is due to the obtuse angle at which the images were taken and it tends to introduce a bias into the segmentation process. However, a careful analysis of the images shows that the size of the segmented "blobs" are for all intent and purposes equal to the actual bubble sizes.

6.6 MEASURING THE EFFECTIVENESS OF THE SEGMENTATION TECHNIQUE

It has been shown that the morphological processing of surface froth images using DSSEs, provides suitable but not error-free segmentation. As mentioned in Chapter 4, section 4.3, it is desirable to be able to numerically quantify the efficiency and thus accuracy of the segmentation process, to gauge the overall performance of the technique.

This can be achieved by :

- Working with an area of interest (AOI) of an image (c.f. Figure 6.27).
- Manually segmenting the bubbles within the AOI, to yield an expected number of

bubbles.

- Automatically segmenting images using various sized DSSEs, and then analyzing the AOI (see Chapter 7, section 7.4) to obtain an observed number of bubbles.

6.6.1 WORKING WITH AN AREA OF INTEREST

The results presented so far show that for areas with excessive shadow regions, the segmentation technique effectively falls down and typically a larger SE may be used to extract the relevant boundary information, if it is possible to do so. It is therefore preferable when analyzing segmented surface froth images, that these regions are ignored and this is achieved by only analyzing a specific region or AOI of an image.

As will be shown in Chapter 8, sections 8.2.1 to 8.2.11, the AOIs vary from image to image and in each case, the AOI was chosen such that the bubble boundaries were clearly discernable features, thus ensuring that they would be reliably segmented to give the best possible image for analysis. The efficiency of the segmentation technique is determined by identifying and counting the number of bubbles within the AOI that lie within a specified size range. The analysis techniques for doing this are discussed in Chapter 7.

As an example Figure 6.30 shows the processed AOI for the test image BUB1.CFI, which was chosen to be :

$$\{(x_1,y_1);(x_2,y_2)\} = \{(55,42);(435,410)\}$$

and the size range to be :

$$25 \text{ pixels} \leq \text{bubble area} \leq 7000 \text{ pixels}$$

so that all the bubbles within the AOI would be detected.

6.6.2 THE MANUAL SEGMENTATION OF SURFACE FROTH IMAGES

The manual segmentation involved visually identifying the individual bubble boundaries and manually tracing around them using a drawing package on the SUN workstation. The boundaries were drawn in white (255), as it was found that the maximum intensity never exceeded 230 and thus by thresholding the image at 254, the manually identified boundaries could be extracted and subsequently, the desired AOI analyzed to obtain an expected number

of bubbles. The analysis involved identifying and marking the centroids of all bubbles in the AOI lying within the specified size range and the manually segmented image is illustrated in Figure 6.30, with the centroids of the identified bubbles marked with a white plus.

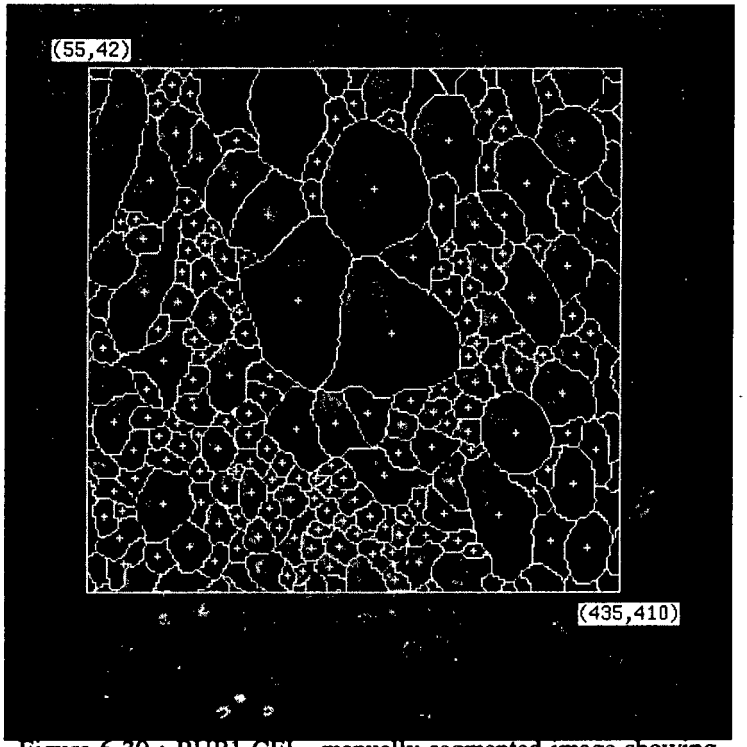


Figure 6.30 : BUB1.CFI - manually segmented image showing the identified bubbles within the AOI.

The analysis routine identified 173 bubbles for the manually segmented test image BUB1.CFI.

6.6.3 THE AUTOMATIC SEGMENTATION OF SURFACE FROTH IMAGES

BUB1.CFI was automatically segmented using DSSEs of radii 3 to 20 pixels and the individual segmented images were then analyzed to obtain the number of automatically detected bubbles for the corresponding SE size. The results obtained are tabulated in Table 6.1, together with the calculated efficiency values, which are defined in section 6.6.4 and the optimal SE values, which are defined in section 6.6.5. The sequence of individual images, illustrating the effectiveness of each SE size are given in Appendix A.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	25	14.45	1.00
4	62	35.84	2.48
5	97	56.07	3.88
6	119	68.79	4.76
7	134	77.46	5.36
8	147	84.97	5.88
9	146	84.39	5.84
10	150	86.71	6.00
11	155	89.60	6.20
12	155	89.60	6.20
13	146	84.39	5.84
14	145	83.82	5.80
15	140	80.92	5.60
16	136	78.61	5.44
17	132	76.30	5.28
18	122	70.52	4.88
19	128	73.99	5.12
20	124	71.68	4.96

Table 6.1 : Bubble detection as a function of SE radius, with calculated efficiency and optimal SE values.

6.6.4 THE EFFICIENCY OF THE SEGMENTATION TECHNIQUE

As has been shown, segmented surface froth images produce an interconnected network of lines, that give rise to blobs that are representative of individual bubbles. A measure of the efficiency (η) of the segmentation technique, can be obtained by computing the ratio of the number of blobs detected automatically in an AOI to the number of blobs detected manually in the same AOI, and can therefore be defined as :

$$\eta\ (\%) = \frac{\text{number of blobs detected with SE radius R}}{\text{number of blobs detected manually}} \times 100$$

By plotting a graph of η against increasing SE size, Figure 6.31, an efficiency curve is obtained.

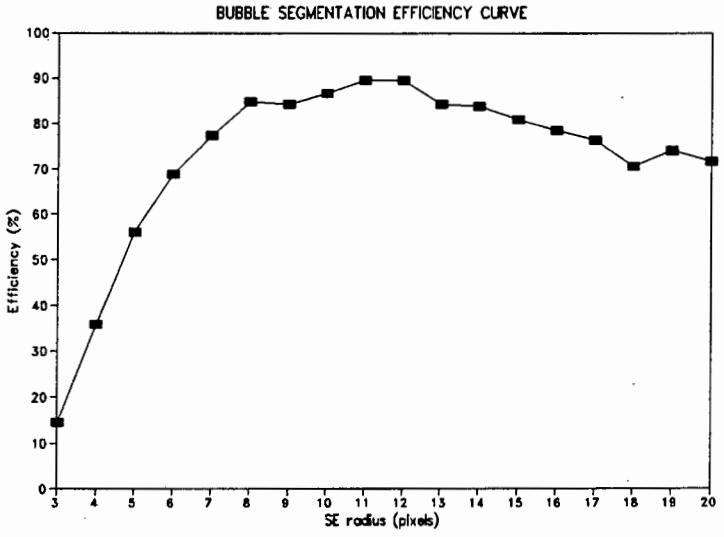


Figure 6.31 : Efficiency curve for BUB1.CFI.

The graph exhibits a definite characteristic shape, indicating that there is an optimum SE radius for the best possible segmentation. With reference to Figure 6.31, as the SE increases in radius, its ability to resolve the bubble boundaries for both the smaller and larger bubbles and thus to detect individual bubbles, improves rapidly. As a result the graph rises sharply to start with, but then begins to tapper off until optimal segmentation is achieved, which corresponds to a particular SE radius. As the SE increases beyond the optimal radius, the larger SE has two distinct, but counter-active affects on the segmentation :

- Firstly, the larger SE is unable to resolve the smaller bubbles and effectively "rolls over" them. The small neighbouring bubbles therefore coalesce and are incorrectly identified as larger bubbles. This merging effect results in a roll-off of the graph.
- Secondly, if a dual lighting source has been used, as was the case with the images that were processed, multiple highlights occur on the surfaces of the larger bubbles. The larger SE incorrectly identifies the valleys between these highlights as bubble boundaries, thereby resulting in the fragmentation of the larger bubbles into a number of smaller bubbles. This fragmentation of the larger bubbles counteracts the bubble merging, but does not neutralise it as from the images processed there are typically more smaller bubbles than larger bubbles and bubble coalescence is the

predominant factor resulting in the roll-off of the graph.

These two affects are clearly illustrated in Figures A.14 to A.18, in Appendix A.

6.6.5 DETERMINATION OF THE OPTIMAL STRUCTURING ELEMENT SIZE

A quantitative measure for the optimal SE is obtained by defining the quantity γ, which is given by the ratio of the number of automatically detected bubbles for increasing SE radius to the number of automatically detected bubbles, using the smallest possible SE and is therefore defined by :

$$\gamma = \frac{\text{number of blobs detected by SE of radius R}}{\text{number of blobs detected by smallest SE}}$$

A similar, but scaled graph to that given in Figure 6.31, is obtained by plotting γ against increasing SE radius, as illustrated in Figure 6.32. This illustrates that the ratio γ is indicative of the efficiency of the process and therefore provides an indirect measure of η.

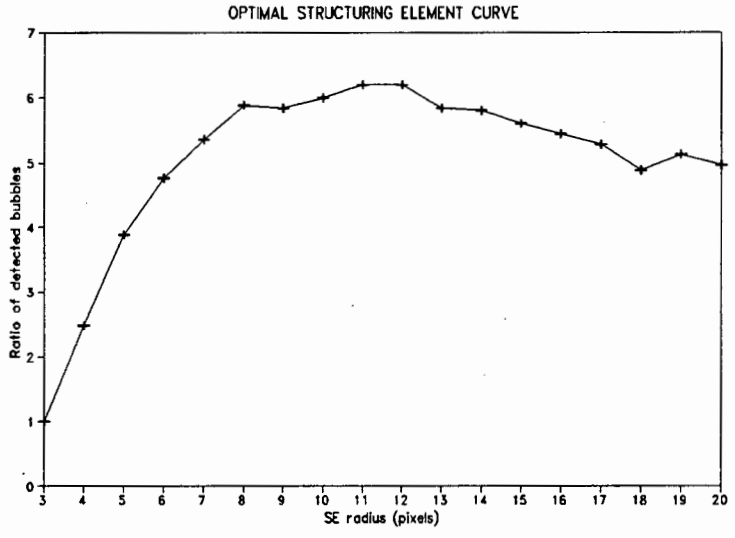


Figure 6.32 : Optimal SE curve for BUB1.CFI.

The segmentation process, as stated previously, is not error-free and particularly in the shadow regions, it may result in the spurious formation of bubbles that may not actually exist. This spurious formation of bubbles can be compensated for in part, by performing three point

averaging of the γ data, Figure 6.33, which smooths the shape of the graph.

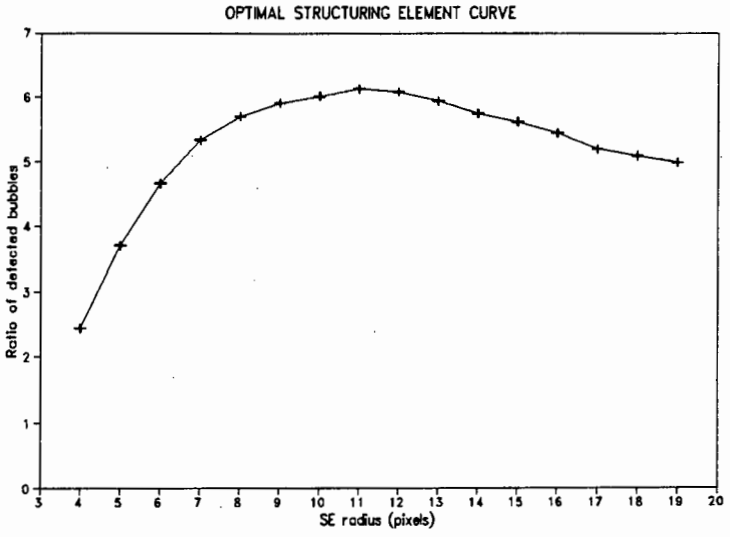


Figure 6.33 : BUB1.CFI optimal SE curve, after performing three point averaging of the data.

The smoothed graph reveals that the optimal SE that maximises the segmentation of surface froth images, corresponds to the highest point on the hump of the graph and furthermore, this point can be determined automatically. This means that a machine vision system implementing this segmentation technique for the characterisation of surface froth structures, by computing the optimal SE curve and subsequently locating the optimal SE the system will in principle be self-tuning. From Figure 6.33, the optimal SE for the test image BUB1.CFI, corresponds to a SE of radius 11 pixels.

As the hump is a function of SE radius one would intuitively expect the type of froth structure analyzed to affect the position of the hump. For a wet froth, that consists of small, spherically shaped bubbles, a smaller DSSE should effectively provide the best segmentation therefore shifting the hump to the left; and for a dry froth, that consists of larger, elongated bubbles, a similarly or larger sized DSSE should provide the best possible segmentation, thereby marginally shifting the hump to the right. This is confirmed by the results presented in Chapter 8, sections 8.2.1 to 8.2.11.

CHAPTER 7

THE ANALYSIS AND CHARACTERISATION OF SEGMENTED SURFACE FROTH IMAGES

"The analysis recognizes, locates the position and orientation, and provides a sufficiently detailed symbolic description or recognition of those imaged objects deemed to be of interest in the three-dimensional environment."

R.M. Haralick, Glossary of Computer Vision, Machine Vision International, Ann Arbor, Michigan 48104, USA, 1986.

Chapter 6 showed that the morphological processing of surface froth images, results in segmented images that are characterized by an interconnected network of lines which are the extracted bubble boundaries and which give rise to isolated regions or blobs that are representative of the individual bubbles.

As bubble size and bubble shape provide considerable information on the amount of drainage within the froth structure and hence on the grade/recovery operating point of a flotation cell (Chapter 2, sections 2.3.2 (A) and (B)), the individual blobs can be processed to provide statistical distributions of size and shape, which can be used to quantitatively characterise the surface froth.

7.1 SIZE DETERMINATION

For surface froth structures, since the boundary of a bubble completely specifies the surface area occupied by the bubble, a direct measure of bubble size can be obtained by computing the area of an individual blob.

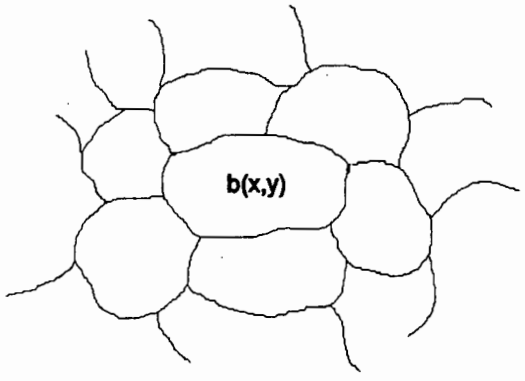


Figure 7.1 : Definition of the area of a blob b(x,y).

Thus for a discrete blob b(x,y), illustrated in Figure 7.1, the area of the blob is given by :

$$Area = \sum_x \sum_y (b(x,y) \cdot a_p) \qquad ...(7.1)$$

where a_p is a scale factor that specifies the area of a pixel.

7.2 SHAPE DETERMINATION

The shape of an object is a complex measurement to quantify precisely and considerable literature exists on various shape analysis techniques (Persoon and Fu, 1977), (Davis, 1977, I), (Davis, 1977, II), (Pavlidis, 1978) and (Shapiro, 1980).

As the bubbles that constitute surface froth structures are roughly circular or elliptical in shape, there is a degree of flexibility available in the analysis of the shape of the blobs in segmented surface froth images, as the blobs can be measured in terms of their circularity or their elipticity.

7.2.1 CIRCULARITY MEASURE

Circularity measures are so called because they are minimized by circular shapes (Castleman, 1979) and the most commonly used circularity measure is the dimensionless quantity given

by :

$$C = \frac{P^2}{4\Pi A} \qquad \ldots(7.2)$$

where P is the object perimeter and A the object area.

The measure is actually derived from the isoperimetric inequality (Rosenfeld and Kak, 1982, p. 265), that states for a shape in the real plane, $P^2/A \geq 4\pi$; with the equality holding iff the object is circular. The normalized measure $P^2/(4\pi A)$ is therefore frequently used as a measure of circularity, where $C = 1$ for circular shapes and $C > 1$ for less circular shapes.

Strictly speaking though, for discrete shapes, the quantity is not a true measure of circularity as a discrete circle never satisfies the equality, and it is in fact more a measure of the "dispersedness" of a shape (Rosenfeld and Kak, 1982, p. 265). Intuitively for a discrete shape S, that is relatively compact, its perimeter will be small and C will be close to unity; however, if S is elongated or irregular in shape, the perimeter will be larger and the shape will be more dispersed.

7.2.2 ELIPTICITY MEASURE

An alternate method for quantifying the circularity or elipticity of a shape, is to measure the ratio of the axis of elongation of the shape and the corresponding perpendicular axis. If the shape is almost circular, a value near unity will be obtained. Such a measure, can be derived using moments.

(A) MOMENTS

The theory of moments has its origin in the field of mechanics. For a continuous, bounded, two-dimensional function f(x,y), its infinite set of moments are defined by the equation :

$$m_{jk} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^j y^k f(x,y)\, dx\, dy \qquad \ldots(7.3)$$

where j, k ≥ 0 and the parameter j + k is called the **order of the moment.**

For grey level images, quantized using 8 bits, the function f(x,y) can be physically interpreted as point masses of magnitude 0-255, located at positions (x,y) in the image. It is common practice however, to threshold grey level images to give binary images, where the equivalent discrete equation for calculating the set of moments for a binary image b(x,y) is given by :

$$m_{jk} = \sum_x \sum_y x^j y^k b(x,y) \qquad \ldots(7.4)$$

These moments are unique for a particular object and can therefore be used to describe the shape of an object.

The simplest moment is the zero order moment m_{00} :

$$m_{00} = \sum_x \sum_y b(x,y) \qquad \ldots(7.5)$$

which is equivalent to the area of the object.

Furthermore, as will be shown, the first order moments (m_{10},m_{01}) and second order moments (m_{20},m_{11},m_{02}) are useful in deriving a robust elipticity measure.

(B) CENTRAL MOMENTS

The exact location of an object within an image is often required and it is standard practice to define this location using the centre of gravity ($\bar{x}$,$\bar{y}$) of the object, as illustrated in Figure 7.2.

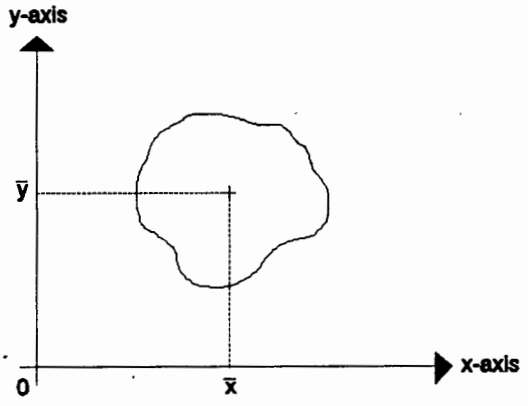


Figure 7.2 : Object location using centroids.

The centroid $(\bar{x},\bar{y})$ of an object, is calculated using the zeroth order moment m_{00} and the first order moments m_{10} and m_{01} as follows :

$$\bar{x} = \frac{\sum_x \sum_y x\, b(x,y)}{\sum_x \sum_y b(x,y)} = \frac{m_{10}}{m_{00}} \qquad \ldots(7.6)$$

$$\bar{y} = \frac{\sum_x \sum_y y\, b(x,y)}{\sum_x \sum_y b(x,y)} = \frac{m_{01}}{m_{00}} \qquad \ldots(7.7)$$

Using the centroid positions $(\bar{x},\bar{y})$ the axes can be translated such that the origin lies at the centroid of the object and a new set of position invariant moments, known as **central moments**, defined by :

$$\mu_{jk} = \sum_x \sum_y (x - \bar{x})^j (y - \bar{y})^k b(x,y) \qquad \ldots(7.8)$$

can be calculated with respect to the new origin.

As the second order central moments will be used in the derivation of an elipticity measure (see Appendix B), they will be derived in full.

$$\begin{aligned}
\mu_{11} &= \sum_x \sum_y (x - \bar{x})^1 (y - \bar{y})^1 b(x,y) \\
&= \sum_x \sum_y (xy - \bar{x}y - x\bar{y} + \bar{x}\bar{y}) b(x,y) \\
&= m_{11} - \bar{x}m_{01} - \bar{y}m_{10} + \bar{x}\bar{y}m_{00} \\
&= m_{11} - \left(\frac{m_{10}}{m_{00}}\right) m_{01} - \left(\frac{m_{01}}{m_{00}}\right) m_{10} + \left(\frac{m_{10} m_{01}}{m_{00}^2}\right) m_{00}
\end{aligned} \qquad \ldots(7.9)$$

Which simplifies to give :

$$\mu_{11} = m_{11} - \bar{y}m_{10} \qquad \ldots(7.10)$$

and :

$$\begin{aligned}\mu_{20} &= \sum_x \sum_y (x - \bar{x})^2 (y - \bar{y})^0 b(x,y) \\ &= \sum_x \sum_y (x^2 - 2x\bar{x} + \bar{x}^2) b(x,y) \\ &= m_{20} - 2\bar{x}m_{10} + \bar{x}^2 m_{00} \\ &= m_{20} - 2\left(\frac{m_{10}}{m_{00}}\right)m_{10} + \left(\frac{m_{10}}{m_{00}}\right)^2 m_{00}\end{aligned} \qquad \ldots(7.11)$$

Which simplifies to give :

$$\mu_{20} = m_{20} - \bar{x}m_{10} \qquad \ldots(7.12)$$

By similar expansion and simplification, μ_{02} can be shown to be :

$$\mu_{02} = m_{02} - \bar{y}m_{01} \qquad \ldots(7.13)$$

A note on standards here, the moment of inertia μ_{11} is commonly denoted as I_{xy}, μ_{20} as I_y, the moment of inertia with respect to the y axis and μ_{02} as I_x, the moment of inertia with respect to the x axis.

Using these central moments, a robust elipticity measure E, can be defined as (see Appendix B for a full derivation) :

$$E = \sqrt{\frac{I_a}{I_b}} \qquad \ldots(7.14)$$

Where I_a and I_b are the major and minor axes respectively of an equivalent ellipse and are given by :

$$I_a = \frac{1}{2}(\mu_{20} + \mu_{02}) + \sqrt{\mu_{11}^2 + \left(\frac{\mu_{20} - \mu_{02}}{2}\right)^2}$$

$$I_b = \frac{1}{2}(\mu_{20} + \mu_{02}) - \sqrt{\mu_{11}^2 + \left(\frac{\mu_{20} - \mu_{02}}{2}\right)^2} \qquad \ldots(7.15)$$

As mentioned in Chapter 2, section 2.6.1, these results differ from those presented by Woodburn *et. al.* and it is difficult to ascertain from the paper whether the disagreement is due to erroneous typing or incorrect derivation.

7.3 THE PARAMETERIZATION OF BLOBS

From the results presented in sections 7.2.1 and 7.2.2, segmented surface froth images can be characterised by computing the primary object parameters listed in Table 7.1, for the individual blobs.

Measured :	
Perimeter	P
Area	m_{00}
Sum of x values	m_{10}
Sum of y values	m_{01}
Sum of xy values	m_{11}
Calculated :	
x centroid	$\bar{x}$
y centroid	$\bar{y}$
Second central x moment	μ_{20}
Second central y moment	μ_{02}
Second central xy moment	μ_{11}
Major axis	I_a
Minor axis	I_b
Circularity measure	C
Elipticity measure	E

Table 7.1 : Primary object parameters for characterising the blobs of segmented surface froth images.

7.4 SEGMENTED SURFACE FROTH ANALYSIS

The analysis of segmented surface froth images must perform two functions :

- The individual blobs must be detected and labelled.
- The identified blobs must then be characterised using the object parameters listed in Table 7.1.

This can be achieved using boundary tracing or connected component analysis.

7.4.1 BOUNDARY TRACING

Boundary or contour tracing is a fundamental searching or traversing process by which the bounding contour of a blob can be detected. A typical boundary tracing algorithm, such as the one to be discussed here, will in practice perform two functions; blob location and blob analysis.

Blob Location : The objective here is to locate the blobs within an image. This is done by scanning the image from left to right, top to bottom, until a seed or starting pixel (x_1,y_1) of a blob is located. At this point, the function transfers control to the blob analysis module.

Blob Analysis : The objectives of this module are threefold. Firstly, the boundary of the detected blob must be traced and a list of valid boundary points generated, secondly, the list of boundary points must be sorted and thirdly, the object parameters P, m_{00}, m_{10}, m_{01}, m_{11} measured and their derivatives $\bar{x}$, $\bar{y}$, μ_{20}, μ_{02}, μ_{11}, I_a, I_b, C and E computed.

For this reason the function is usually divided into three stages. However, if the border points are sorted, as they are traced then only two stages are required.

Stage 1 : The seed pixel (x_1,y_1) is used as the starting point, and a trace direction code, Figure 7.3, is used to intelligently assist in the determination of the next border point of the object (Kulpa, 1977), (Chang and Leu, 1990) and (Liow, 1991). The trace direction consists of an eight neighbourhood code, labelled 0 to 7, which is used to indicate the previous search direction. The algorithm systematically follows the boundary of the object using the previous known search direction as a pointer for

searching for the next possible border point, using the eight neighbourhood. To assist in the computation of the object parameters, the identified border points are flagged as BLOCKED or UNBLOCKED, Figure 7.4, and singular border points such as a, i and m, are added to the list of border points twice, for reasons given by Chang and Leu (1990).

It is standard practice to compute the perimeter of the object, whilst tracing out its border.

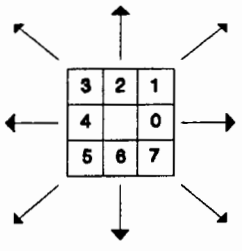


Figure 7.3 : Boundary trace direction codes.

Stage 2: The border points are sorted in increasing row and column coordinates, to assist the computation of the object parameters.

Stage 3: The sorted border points are used to "fill" the object and to compute the object parameters. Filling of the region is achieved by connecting all pixels between two border points that have been marked BLOCKED, Figure 7.4 and that lie on the same row. UNBLOCKED border points are ignored.

Table 7.2, illustrates how the described boundary tracing algorithm would analyze the blob illustrated in Figure 7.4, assuming a clockwise traversal of the border.

7.4.2 CONNECTED COMPONENT ANALYSIS

Connected component analysis or blob analysis uses a connected component operator, that takes as input a thresholded or binary image and produces aswutput an image in which the individual blobs have been identified and uniquely labelled. For each identified blob, the operator also calculates typical object parameters such as the area, perimeter, the number of

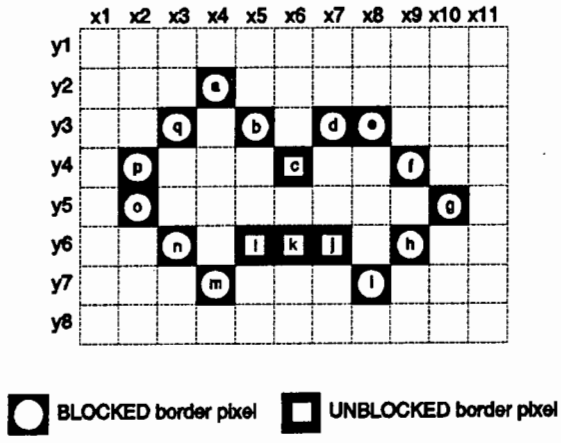


Figure 7.4 : Located blob, using boundary tracing.

Generated List	Sorted List	Region Filling
y_2, x_4 BLOCKED	y_2, x_4 BLOCKED	
y_2, x_4 BLOCKED	y_2, x_4 BLOCKED	a → a
y_3, x_5 BLOCKED	y_3, x_3 BLOCKED	
y_4, x_6 UNBLOCKED	y_3, x_5 BLOCKED	q → b
y_3, x_7 BLOCKED	y_3, x_7 BLOCKED	
y_3, x_8 BLOCKED	y_3, x_8 BLOCKED	d → e
y_4, x_9 BLOCKED	y_4, x_2 BLOCKED	
y_5, x_{10} BLOCKED	y_4, x_6 UNBLOCKED	
y_6, x_9 BLOCKED	y_4, x_9 BLOCKED	p → f
y_7, x_8 BLOCKED	y_5, x_2 BLOCKED	
y_7, x_8 BLOCKED	y_5, x_{10} BLOCKED	o → g
y_6, x_7 UNBLOCKED	y_6, x_3 BLOCKED	
y_6, x_6 UNBLOCKED	y_6, x_5 UNBLOCKED	
y_6, x_5 UNBLOCKED	y_6, x_6 UNBLOCKED	
y_7, x_4 BLOCKED	y_6, x_7 UNBLOCKED	
y_7, x_4 BLOCKED	y_6, x_9 BLOCKED	n → h
y_6, x_3 BLOCKED	y_7, x_4 BLOCKED	
y_5, x_2 BLOCKED	y_7, x_4 BLOCKED	m → m
y_4, x_2 BLOCKED	y_7, x_8 BLOCKED	
y_3, x_3 BLOCKED	y_7, x_8 BLOCKED	i → i

Table 7.2 : Processed blob.

holes in a blob, the bounding rectangle, the extremal (x,y) points of a blob, the second moments, the blob orientation and the major and minor axes of the blob.

Unlike a typical boundary tracing algorithm, such as the one discussed in section 7.4.1, a connected component analysis algorithm requires no form of sorting and is therefore considerably quicker. Typically, such algorithms use run-length coding to describe as well as compress the binary image and then generate the blobs by searching for overlapwng runs.

Thus a typical connected component algorithm consists of two functions :

- A function to generate the run-lengths for each row in a binary image.
- A function that implements a connected component operator that generates the blobs using the predetermined runs and also measures the primary object parameters P, m_{00}, m_{10}, m_{01} and m_{11}.

A connected component algorithm is generally a more efficient and elegant way of realizing blob analysis and for reasons given later, this type of algorithm was used to analyze segmented surface froth images.

Run-length coding

Run-length coding is a frequently used coding technique for compressing binary images and exploits the fact that for any particular row in a binary image, there will be long runs of zeros and ones (Horn, 1986, p. 58) and thus the data can be fully specified by recording the length of the runs of zeros and ones.

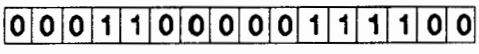


Figure 7.5 : An image line for illustrating run-length coding.

For the image line given in Figure 7.5, the run-length code could be represented by :

{pixel value, length of run, flag to indicate the end of a run}

which would give :

{0,3,#,1,2,#,0,5,#,1,4,#,0,2,#}

where # indicates the end of a run.

Run-length coding allows for the rapid and easy computation of the primary object parameters P, m_{00}, m_{10}, m_{01} and m_{11}, and a comprehensive treatment on the computation of these parameters is given by Horn (1986, pp. 58-61) and will thus not be covered here.

Connectivity analysis

Once the run codes have been generated for each row in the image, the individual blobs within the image must be generated using the runs. This is achieved by "connectivity analysis" which groups the overlapping runs from adjacent rows into a data structure called a "blob descriptor" that keeps track of the primary blob parameters, which are typically updated at the end of each overlapping run.

As the connectivity analysis algorithm processes each row of run codes, it uses an "active blob" list to keep track of all the current blobs and as each row of runs is processed, the connectivity operator performs one of four tasks; blob insertion, continuation, merging and deletion, as illustrated in Figure 7.6.

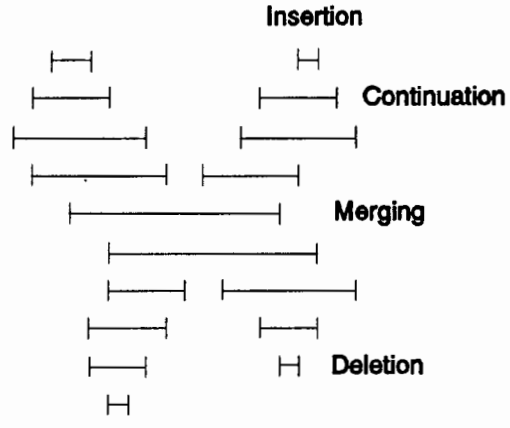


Figure 7.6 : Located blob, using connectivity analysis.

Insertion : When a run occurs which does not overlap with an existing blob, a new blob descriptor must be created. This new blob descriptor is then inserted into the active blob list.

Continuation : When a run overlaps with the run on a previous row of an existing blob, using 4-neighbour connectivity (see Appendix C), then the run must be added to the blob descriptor.

Deletion : If the process passes the columns for an existing blob without finding a run, then

the blob descriptor is moved from the list of active blobs and stored.

Merging : When a run is found to overlap with two distinct blobs, then these blobs must be merged into a single blob descriptor.

Segmented surface froth images were analyzed using connected component analysis, for two reasons :

- Firstly and most importantly, existing code was available that required minor modifications in order to process 512x512 images, of the type used.
- Secondly, as illustrated in the sequence of images given in Appendix A, the segmentation of surface froth images often results in erroneous detections in the form of loose or unconnected branches. Instead of trying to eliminate these branches during the postprocessing of the segmented images, it was decided that it would be simpler to leave them and process them in the analysis stage. As is illustrated in Figure 7.7, connectivity analysis of a blob with an unconnected branch will result in the identification of two separate blobs, labelled 1 and 2. However, these blobs will later be merged into a single blob, labelled 3, with continued processing.

Figure 7.7 : The processing of erroneous boundary detections.

Thus unconnected branches are efficiently handled, but they will however tend to bias the computed circularity measure, because, as can be seen from Figure 7.7, the perimeter of the branch will be included in the overall perimeter of the merged blob. As a result, it would be reasonable to expect the elipticity measure to provide a more reliable and unbiased measurement of the shape of the blob. This is in fact the case, and is substantiated by the results presented in Chapter 8, sections 8.2.1 to 8.2.11.

Implementation

The modified code for the connectivity algorithm, quickly and efficiently analyzed the blobs in segmented surface froth images. Figure 7.8, illustrates the run-times obtained for the full processing of surface froth images, viz. the preprocessing, segmentation, postprocessing and the analysis, together with the run-times for the morphological segmentation alone, for DSSEs of radius 3 to 20 pixels. It is clear that the automatic processing will be optimized by optimizing the morphological segmentation of the images.

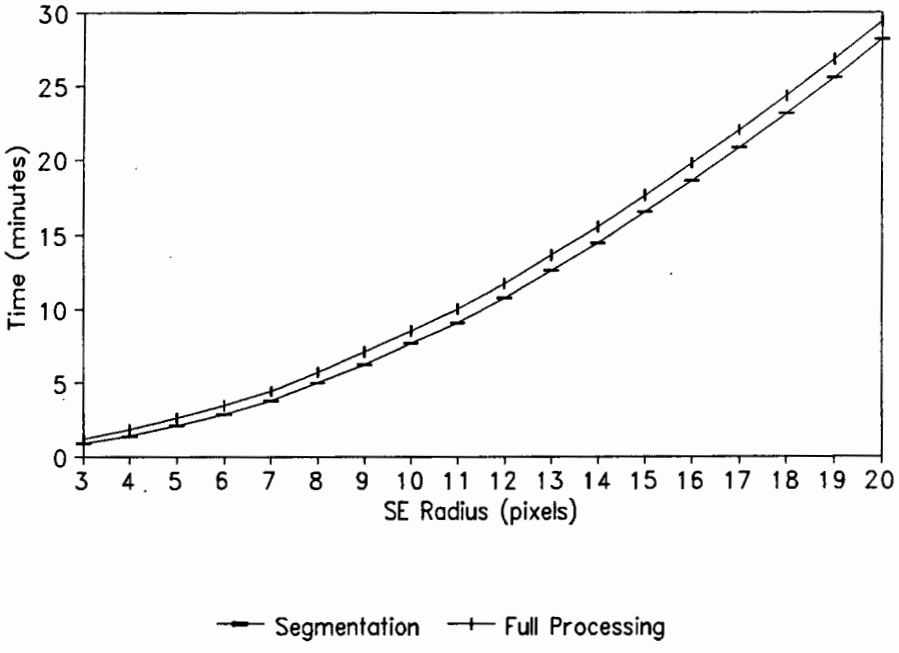


Figure 7.8 : Run-times of the full processing of surface froth images, for DSSEs of radii 3 to 20 pixels.

Figure 7.9 is the result of analyzing the optimally segmented test image BUB1.CFI, using a DSSE of radius 11, as discussed in section 6.6.5, and Figure 7.10 illustrates the accuracy of the segmentation technique, by overlaying the optimally segmented image (Figure 7.9) on the original image. Blobs with areas in the range 25 to 7000 pixels, have had their centroids marked with black pluses (+) in Figure 7.9 and with white pluses in Figure 7.10.

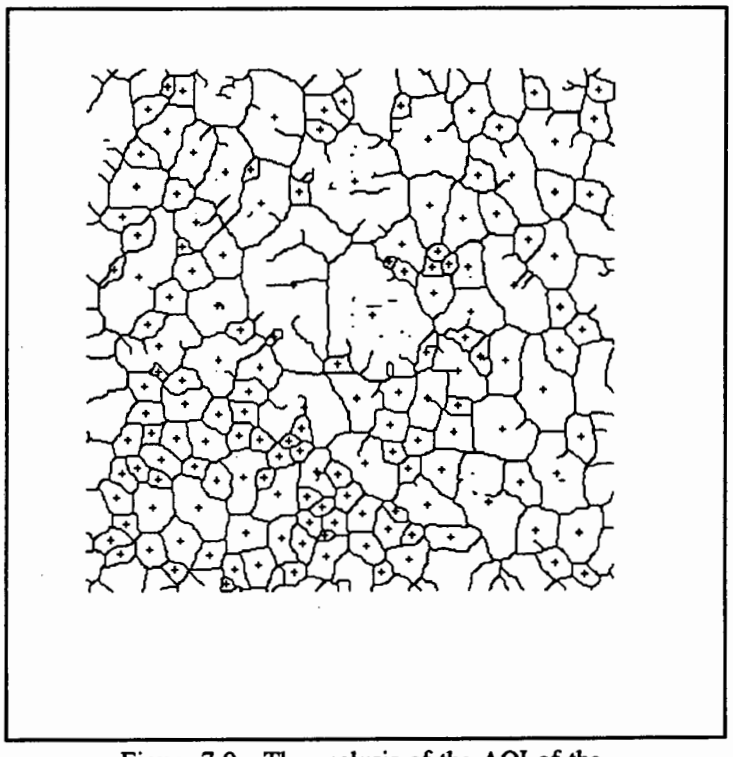


Figure 7.9 : The analysis of the AOI of the optimally segmented BUB1.CFI.

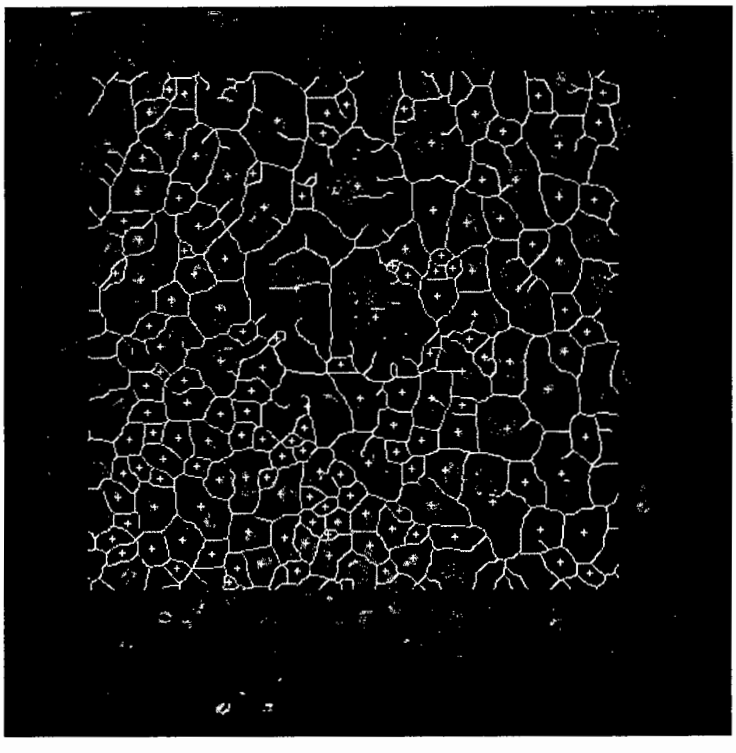


Figure 7.10 : Figure 7.9 overlayed on the original image.

7.5 THE STATISTICAL CHARACTERISATION OF SURFACE FROTHS

The connected component analysis of segmented surface froth images provides a measure of the area (size), circularity (compactness) and elipticity (elongation) of the individual blobs and a graphical representation of the required distributions can be obtained by plotting the individual histograms.

7.5.1 STATISTICAL ANALYSIS

The histogram is a frequently used graphical technique for presenting experimental data and is generated by sorting the data values into a number of equal-width bins. In practice, the data must be divided into as many bins as are statistically meaningful and the problem often arises as to how to select the desired bin width.

If the data values are more or less normally distributed, it can be shown that the optimum bin width Δx is approximately given by (Heald, 1984) :

$$\Delta x \sim \sigma\left(\frac{20}{N}\right)^{\frac{1}{6}} \qquad ..(7.16)$$

where σ is the standard deviation of the data and N is the number of data values.

For data that is skewed, that is when the shape of the distribution of values is markedly asymmetrical with either the right-hand tail extended (positively skewed) or the left-hand tail extended (negatively skewed); the selection of an optimum bin width is a matter of subjective judgement.

Once the histogram for a set of experimental data has been generated, the mean, median, mode, standard deviation and skewness are the most frequently used parameters with which to characterise the distribution (Guttman, Wilks and Hunter, 1982).

Mean

The mean or average, denoted by μ, is the first moment of the data and is a measure of the central tendency of the data values. It is computed by :

$$\mu = \frac{1}{N}\sum_{i=1}^{N} d_i \qquad \ldots(7.17)$$

where N is the number of data values and d_i is the i^{th} value of the set of data values $\{D\} = \{d_1, d_2, \ldots, d_N\}$.

Median

The median or middle value, denoted by Q_{50}, is an alternative measure of the central tendency of the data and is defined as that value which has as many values above it as below it.

Mode

The mode is defined as the most frequently occurring value in a set of data values {D}.

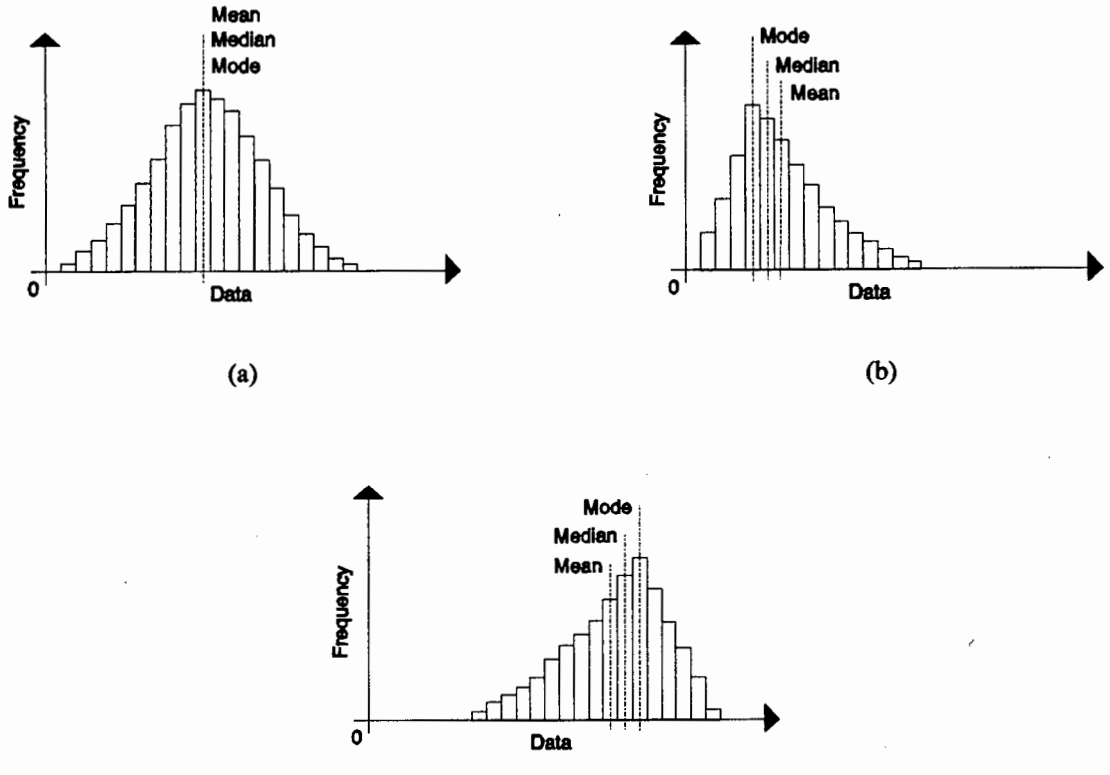


Figure 7.11 : The mean, median and mode for
(a) Normally distributed data.
(b) Positively skewed data.
(c) Negatively skewed data.

When a distribution of data values is fairly symmetrical, Figure 7.11 (a), the mean, median and mode are about equal and any one of these three measurements can be used to describe the central tendency of the data.

For distributions that are asymmetrical, Figures 7.11 (b) and (c), the effect of the skewness is to separate out the three measures of central tendency. For such distributions, the mean is sensitive to the extreme values and thus gives a misleading value for the central tendency. The median though, still retains the characteristic of being a "typical" or central value and for strongly skewed data, it is therefore the preferable measure of central tendency.

Standard Deviation

The standard deviation, denoted by σ, is computed from the second moment of the data and describes the spread or dispersion of the data values on either side of the central value μ. It is given by :

$$\sigma(d_1, d_2, \ldots, d_N) = \sqrt{Var(d_1, d_2, \ldots, d_N)} \qquad \ldots(7.18)$$

where

$$Var(d_1, d_2, \ldots, d_N) = \frac{1}{N-1} \sum_{i=1}^{N} (d_i - \mu)^2 \qquad \ldots(7.19)$$

For skewed distributions, it is important to note that the standard deviation losses its statistical meaning, due to the asymmetry of the distribution.

Skewness

The skewness is the third moment of the data and is a measure of the asymmetry of a distribution. It is given by :

$$Skew(d_1, d_2, \ldots, d_N) = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{d_i - \mu}{\sigma} \right]^3 \qquad \ldots(7.20)$$

A positive value of skewness signifies a distribution with an asymmetric tail extending out towards more positive x, as illustrated in Figure 7.11 (b); a negative value signifies a distribution whose tail extends out towards more negative x, as illustrated in Figure 7.11 (c).

(A) SIZE AND SHAPE DISTRIBUTIONS

The analysis results, which were saved as ASCII files, were statistically processed using the commercial package Statgraphics V5.0, for which the University of Cape Town has site licence. In all a total of six frequency histograms for the test image BUB1.CFI were plotted; these being the expected and observed[1] size, circularity and elipticity distributions. The individual histograms were then statistically characterised by computing the previously discussed parameters. The results are illustrated in Figures 7.12 (a) and (b) to 7.14 (a) and (b).

A Note on the Selection of Bin Width Sizes

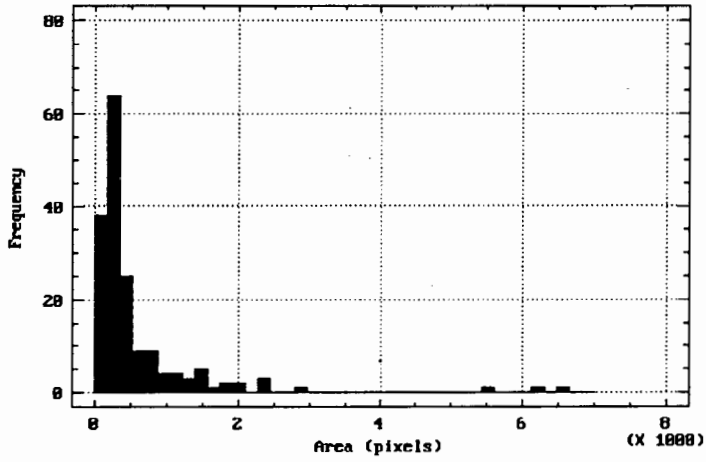
As all the histograms were positively skewed, there was no hard and fast rule for determining the optimal bin width size and the choice was a matter of subjective judgement. The chosen histogram parameters are listed in Table 7.3.

	SIZE (pixels)	CIRCULARITY	ELIPTICITY
Range	1 to 7000	1 to 10	1 to 4
# of Bins	40	40	20
Bin width	175	0.225	0.15

Table 7.3 : Chosen histogram parameters for BUB1.CFI.

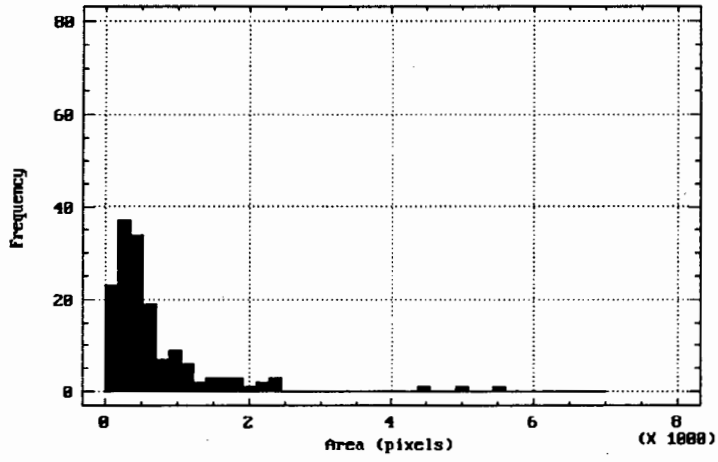
In each case the most suitable number of divisions were arrived at by plotting the data, using Statgraphics, until the chosen number of subdivisions gave a statistically meaningful result. This was done for all of the data that was analyzed using Statgraphics.

[1] Expected refers to the distributions obtained from the manually segmented test image. Observed refers to the distributions obtained from the automatically segmented test image, using the optimal DSSE.



(a)

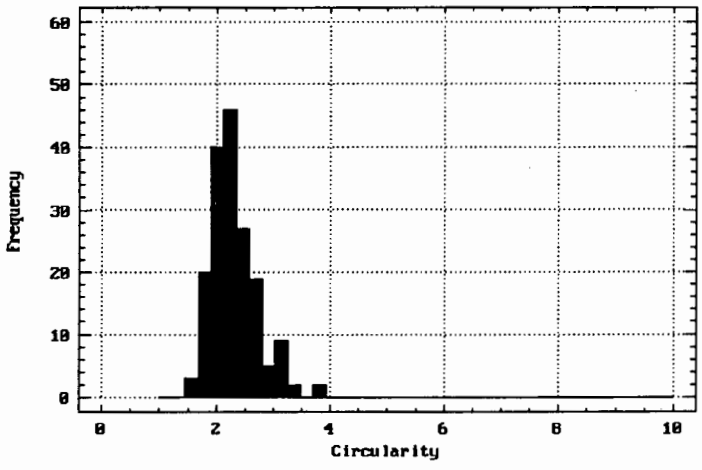
μ = 587.79, σ = 898.42, skewness = 4.47,
Q_{50} = 302, mode = 131.



(b)

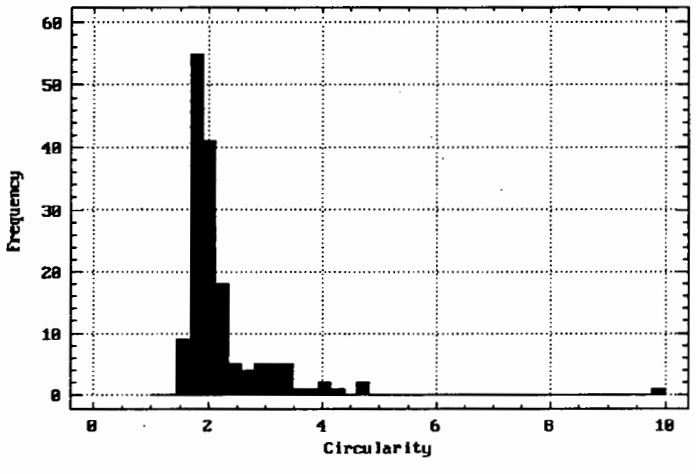
μ = 671.48, σ = 802.74, skewness = 3.48
Q_{50} = 414, mode = 219.

Figure 7.12 : Size distributions for the test image BUB1.CFI :
(a) Expected distribution and (b) Observed distribution.



(a)

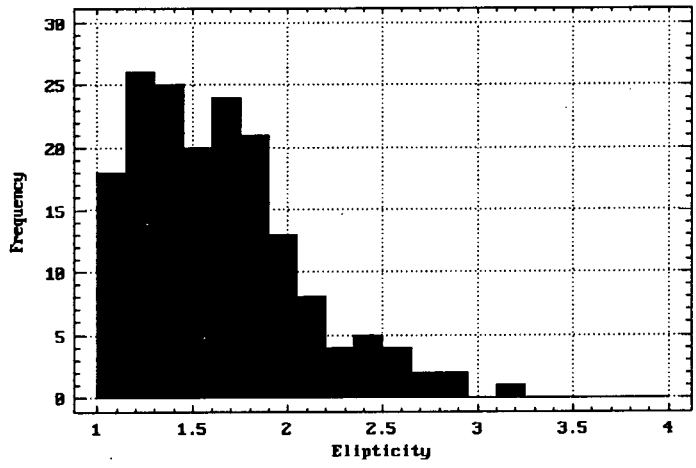
μ = 2.31, σ = 0.40, skewness = 1.09,
Q_{50} = 2.25, mode = 2.03.



(b)

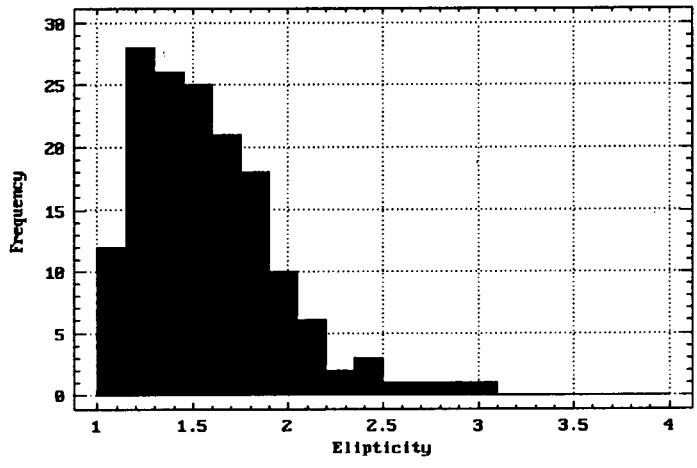
μ = 2.22, σ = 0.86, skewness = 5.09,
Q_{50} = 1.95, mode = 1.78.

Figure 7.13 : Circularity distributions for the test image BUB1.CFI :
(a) Expected distribution and (b) Observed distribution.



(a)

μ = 1.64, σ = 0.43, skewness = 1.00,
Q_{50} = 1.59, mode = 1.35.



(b)

μ = 1.58, σ = 0.37, skewness = 1.13,
Q_{50} = 1.53, mode = 1.56.

Figure 7. 14 : Elipticity distributions for the test image BUB1.CFI :
(a) Expected distribution and (b) Observed distribution.

(B) CHI-SQUARE TEST AS A MEASURE OF "GOODNESS-OF-FIT"

A visual comparison of the expected and observed distributions, reveals that they are similar in shape and that the calculated statistics are reasonably close. It is desirable however, to be able to quantitatively measure the similarity or "goodness-of-fit" between the expected and observed distributions and this can be achieved by applying the chi-square (χ^2) test.

The chi-square test is a frequently used test statistic for comparing the fit of experimentally observed data to theoretically expected values and is defined as :

$$\chi^2 = \sum_{j=1}^{r} \frac{(O_j - E_j)^2}{E_j} \quad \textit{with } f \textit{ degrees of freedom} \qquad \ldots(7.21)$$

where : r is the total number of cells.

O_j are the number of observations occurring in cell j.

E_j are the expected number of observations for cell j, based on a preconceived distribution.

f is the number of degrees of freedom for the test, which in general will be equal to (r-1) minus the number of statistical quantities on which the E_j's are based.

For segmented surface froth structures, as previously stated, the expected (E_j) values are the results obtained by analyzing the manually segmented images and the observed (O_j) values are the results obtained by analyzing the images that have been automatically segmented using the optimal DSSE.

Using the null hypothesis :

H_0 : the observed distribution is similar to the expected or theoretical distribution.

and the alternate hypothesis :

H_1 : the observed distribution is <u>not</u> similar to the expected or theoretical distribution.

and a stated significance level (α), which is typically taken as 0.05 (or 5%), it is possible to statistically interpret the results using tabulated χ^2 values, where :

H_0 is rejected (or H_1 is accepted) if the calculated sample χ^2 > tabled χ^2

H_0 is accepted (or H_1 is rejected) if the calculated sample χ^2 < tabled χ^2

A practical note of importance is that the number of expected or observed counts within each category should be roughly 5 or more. If an E_j or O_j count is significantly less than 5, the adjacent cells are combined, until the value exceeds 5. Furthermore, the chi-square tests will be given in the standard format : $\chi^2_{f;\alpha}$, where f is the number of degrees of freedom for the test and α is the chosen significance level.

A Note on Applying the Chi-square Test to Surface Froth Distributions.

Rewriting equation (7.21) gives :

$$\chi^2 = \sum_{j=1}^{r} \frac{(O_j^2 - 2E_jO_j + E_j^2)}{E_j}$$

$$= \sum_{j=1}^{r} \frac{O_j^2}{E_j} - 2\sum_{j=1}^{r} O_j + \sum_{j=1}^{r} E_j$$

But $\Sigma O_j = \Sigma E_j = N$, the sample size. This therefore simplifies the expression to :

$$\chi^2 = \sum_{j=1}^{r} \frac{O_j^2}{E_j} - N \qquad \ldots(7.22)$$

There is however a problem with the sample size N, when applying the χ^2 test to the expected and observed distributions. An explanation of this problem follows.

As a means of testing the performance of the machine vision system, it was assumed that by manually segmenting the test image, the best possible, "unbiased" results would be obtained. The resultant distributions and statistical measurements could then be used as the theoretically expected results. For a specific AOI, if the number of bubbles detected manually are denoted by N_E, because the machine vision system is incapable of directly mimicking the human visual process, for the same AOI it will detect N_O bubbles automatically, using the optimal DSSE. The problem is that $N_E \neq N_O$ and it is necessary to modify the expected (E_j) values according to the number of bubbles that are detected automatically and this is done as follows.

For the manually determined histogram, the probability p_j of there being e_j bubbles in bin j, is given by :

$$p_j = \frac{e_j}{N_E} \qquad \ldots(7.23)$$

Thus the modified expected value E_j, for N_O bubbles, is given by :

$$E_j = N_O \cdot p_j = \left(\frac{N_O}{N_E}\right) \cdot e_j \qquad \ldots(7.24)$$

The χ^2 test is then performed using this modified E_j value.

The following results were obtained on applying the χ^2 test to the size, circularity and elipticity distributions for the test image BUB1.CFI where N_E = 173 and N_O = 155.

The goodness-of-fit of the size distributions.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	175	23	38	34.05	3.59
176	350	37	64	57.34	7.22
351	525	34	25	22.40	6.00
526	700	19	9	8.06	14.83
701	875	7	9	8.06	0.14
876	1225	15	8	7.17	8.56
1226	1575	5	8	7.17	0.66
1576	2100	7	5	4.48	1.42
2101	7000	8	7	6.27	0.48
		155	173	155.00	42.90

Table 7.4 : χ^2 test for the size distributions.

Calculated : $\chi^2_8 = 42.90$

Tabulated : $\chi^2_{8,0.05} = 15.51$

Conclusion : Reject H_0.

The goodness-of-fit of the circularity distributions.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.000	1.674	10	4	3.58	11.51
1.675	1.899	54	19	17.02	80.35
1.900	2.124	41	41	36.74	0.49
2.125	2.349	18	46	41.22	13.08
2.350	2.574	5	26	23.30	14.37
2.575	2.799	4	19	17.02	9.96
2.800	3.024	5	5	4.48	0.06
3.025	3.249	5	8	7.17	0.66
3.250	10.000	13	5	4.48	16.20
		155	173	155.01	146.68

Table 7.5 : χ^2 test for the circularity distributions.

Calculated : $\chi^2_8 = 146.68$

Tabulated : $\chi^2_{8,0.05} = 15.51$

Conclusion : Reject H_0.

The goodness-of-fit of the elipticity distributions.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.14	12	17	15.23	0.69
1.15	1.29	26	27	24.19	0.14
1.30	1.44	29	24	21.50	2.62
1.45	1.59	24	21	18.82	1.43
1.60	1.74	20	23	20.61	0.02
1.75	1.89	19	22	19.71	0.03
1.90	2.04	9	13	11.65	0.60
2.05	2.19	7	8	7.17	0.01
2.20	4.00	9	18	16.13	3.15
		155	173	155.01	8.69

Table 7.6 : χ^2 test for the elipticity distributions.

Calculated : $\chi^2_8 = 8.69$

Tabulated : $\chi^2_{8,0.05} = 15.51$

Conclusion : Accept H_0.

(C) MODELLING THE SIZE DISTRIBUTIONS

It is possible to describe the particle size distributions, from many size reduction operations, in terms of the log normal and Rosin-Rammler distributions and to a lesser extent by the normal distribution (Popplewell and Peleg, 1989). Early investigative work into modelling the bubble size distributions, showed that it was not possible using the binomial and Poisson

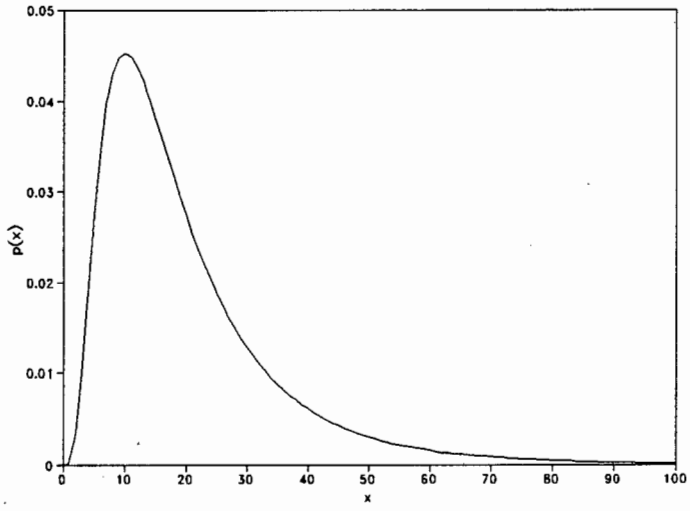


Figure 7.15 : Log normal distribution, with $X_m = 10$ and $\sigma_g = 2$.

distributions. However, since the shape of the size distributions illustrated in Figures 7.12 (a) and (b) are similar to that illustrated in Figure 7.15, which is a log normal distribution, it was thought that a log normal distribution would provide a suitable model. The log normal probability distribution p(x), is given by :

$$p(x) = \frac{1}{X_m \ln \sigma_g \sqrt{2\pi}} \exp\left[\frac{\ln^2 \sigma_g}{2}\right] \cdot \exp\left[\frac{(\ln x - \ln X_m)^2}{2\ln^2 \sigma_g}\right] \qquad \ldots(7.25)$$

where X_m is the mode and σ_g is a characteristic constant. The mean (μ) and variance (σ^2) of the log normal distribution are given by (Popplewell and Peleg, 1989) :

$$\mu = \exp(\theta + \tfrac{1}{2}s^2)$$
$$\sigma^2 = W(W - 1)\exp(2\theta) \qquad \ldots(7.26)$$

Where :

$$s = \ln \sigma_g, \quad \theta = \ln X_m + \ln^2 \sigma_g, \quad W = \exp(s^2)$$

The log normal modelling of the expected and observed size distributions was accomplished using Statgraphics and the following results were obtained.

Modelling the expected size distribution.

Fitted parameters : $\mu = 527.67$

$\sigma = 580.38$

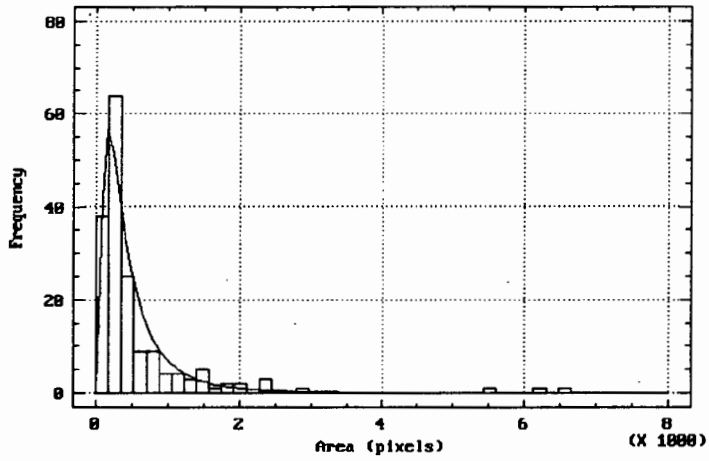


Figure 7.16 : Log normal fit to the expected size distribution, with the best fit parameters $\mu = 527.67$ and $\sigma = 580.38$.

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	175	38	36.9	0.03
176	350	64	48.5	4.98
351	525	25	30.5	0.98
526	700	9	18.6	4.93
701	875	9	11.7	0.60
876	1050	4	7.6	1.70
1051	1225	4	5.1	0.24
1226	1575	8	6.0	0.63
1576	7000	12	8.2	1.81
		173	173.1	15.90

Table 7.7 : χ^2 test for the log normal fit to the expected size distribution.

Calculated : $\chi^2_6 = 15.90$

Tabulated : $\chi^2_{6,0.05} = 12.59$

Conclusion : Reject H_0.

Modelling the observed size distributions.

Fitted parameters : $\mu = 659.16$

$\sigma = 759.69$

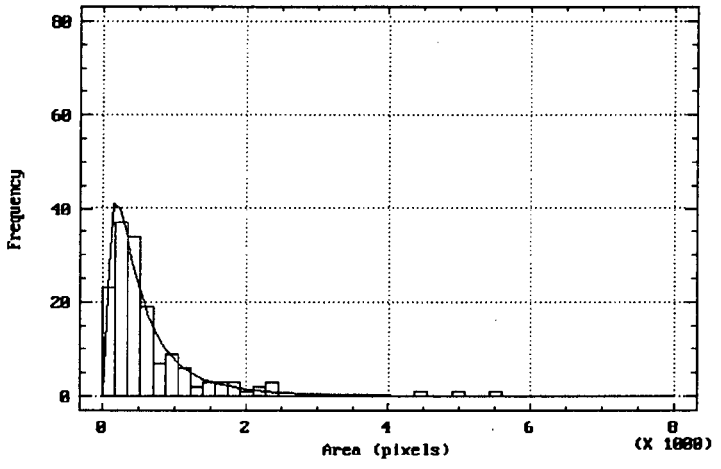


Figure 7.17 : Log normal fit to the observed size distribution, with the best fit parameters $\mu = 659.16$ and $\sigma = 759.69$.

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	175	23	25.2	0.20
176	350	37	38.2	0.04
351	525	34	27.1	1.78
526	700	19	18.0	0.05
701	875	7	12.2	2.19
876	1050	9	8.4	0.05
1051	1225	6	6.0	0
1226	1575	5	7.6	0.86
1576	2100	7	5.7	0.28
2101	7000	8	6.6	0.29
		155	155.0	5.74

Table 7.8 : χ^2 test for the log normal fit to the observed size distribution.

Calculated : $\chi^2_7 = 5.74$

Tabulated : $\chi^2_{7,0.05} = 14.07$

Conclusion : Accept H_0.

CHAPTER 8

THE RESULTS OF CHARACTERISING VARIOUS SURFACE FROTH IMAGES

In addition to the test image BUB1.CFI, a further eleven different surface froth images BUB2.CFI to BUB12.CFI (see Appendix D for photographs of these images), were processed to provide a comparative measure for the overall performance of the methodology presented in Chapters 6 and 7, for segmenting and analyzing surface froth images.

Each image was segmented both manually and automatically and due to excessive shadow regions in most of the images processed, only a specific AOI of the segmented images were analyzed to characterise the surface froth. The manually segmented images provided a set of expected size, circularity and elipticity distributions and the automatically segmented images, obtained using the corresponding optimal DSSE, provided a set of observed size, circularity and elipticity distributions.

Having both the expected and observed distributions, the objectives of the statistical analysis of the data were twofold.

- Firstly, to measure the goodness-of-fit of the expected and observed distributions, to ascertain the performance of the system.
- Secondly, to model both the expected and the observed size distributions using best fitting log normal distributions.

8.1 FORMAT OF THE RESULTS

The results for the processed images are presented in sections 8.2.1 to 8.2.11 and are given in the following format :

- Original image.
- Manually segmented image, overlayed on the original image, with the processed AOI and the centroids of the detected bubbles shown.
- Automatically segmented image, obtained using the optimal DSSE, overlayed on the original image, with the processed AOI and the centroids of the detected bubbles shown.
- Efficiency curve for the automatic segmentation of the image, using DSSEs of radii 3 to 20 pixels.
- Three point averaged optimal DSSE curve.
- Size distributions for (a) the manually segmented image and (b) the automatically segmented image.
- Log normal fits for (a) the size distribution determined from the manually segmented image and (b) the size distribution determined from the automatically segmented image.
- Circularity distributions for (a) the manually segmented image and (b) the automatically segmented image.
- Elipticity distributions for (a) the manually segmented image and (b) the automatically segmented image.
- The corresponding χ^2 tests.

The results are discussed in section 8.3 and a control concept, that illustrates the capability of the system to differentiate between various surface froth structures, using bubble size and bubble shape as the measured control parameters, is presented in section 8.4.

In each case, the distributions determined from the manually segmented images are referred to as the **expected** (E) distributions and the distributions determined from the automatically segmented images are referred to as the **observed** (O) distributions. Furthermore, the chi-square test values for the distributions are tabulated as illustrated in Table 8.1, where f is the

Figure	χ^2 tests	Conclusion
Figure Number	Calculated : χ^2_f Tabulated : $\chi^2_{f;\alpha}$	Accept/Reject H_0

Table 8.1 : Example table for displaying the χ^2 tests.

number of degrees of freedom for the test and α is the significance level of the test. The null hypothesis H_0, given in Chapter 7, section 7.5.1 (B) is accepted if χ^2 calculated $<$ χ^2 tabulated for a 5% significance level, otherwise it is rejected. All relevant data for the individual tests is tabulated in Appendix E.

8.2 PROCESSED IMAGES

For the automatically segmented images, the optimal DSSE used to segment the images, was taken as the DSSE corresponding to the maximum point on the optimal DSSE curve, that was generated for every processed image.

For each distribution, the mean, standard deviation, skewness, median and mode values are given. For the size distributions, all the statistics except for the skewness have units of pixels, with the skewness having units of pixels^3 (see Chapter 7, section 7.5.1). Since the circularity and elipticity measurements are dimensionless quantities, the corresponding statistics have no units.

8.2.1 BUB2.CFI

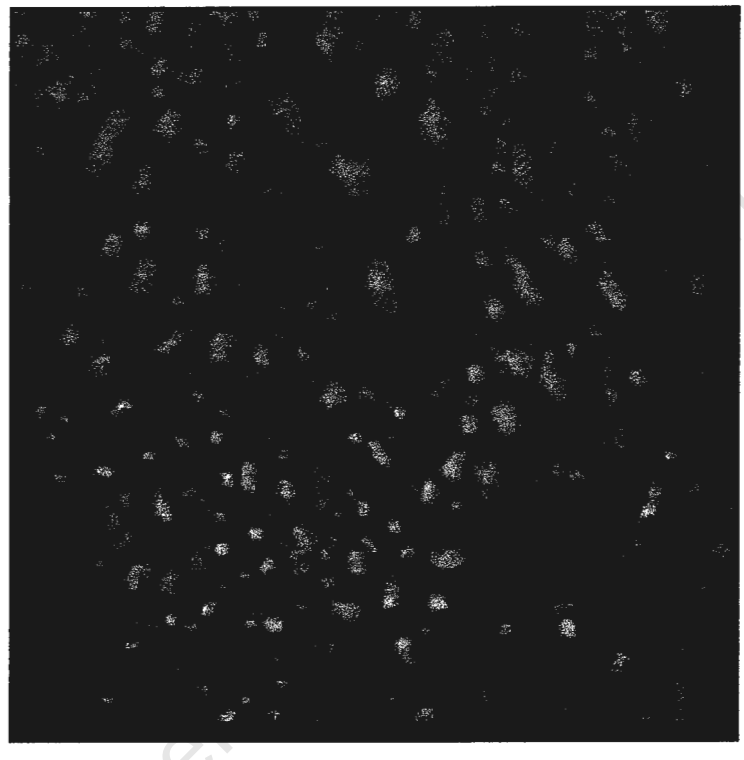


Figure 8.1 : BUB2.CFI - original image.

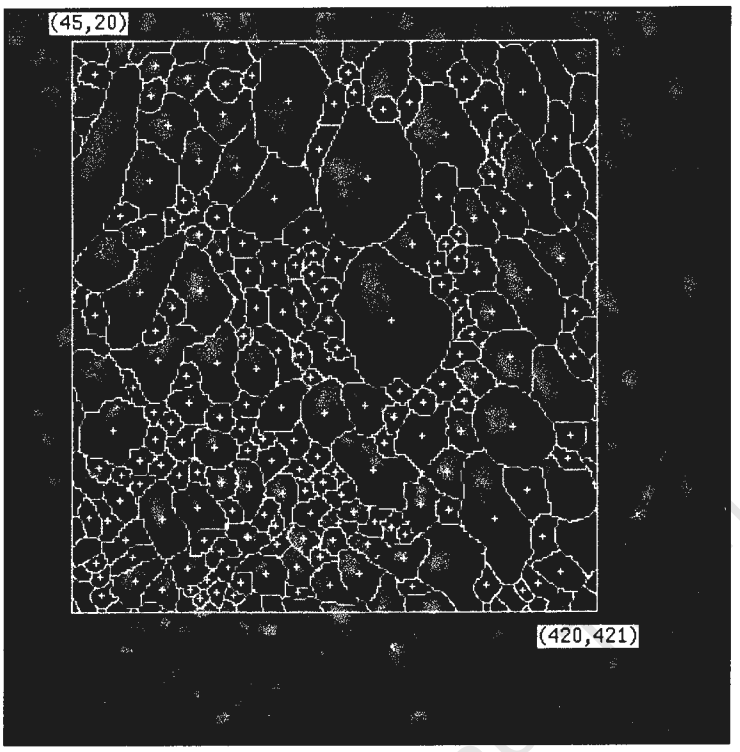


Figure 8.2 : Manually segmented image.

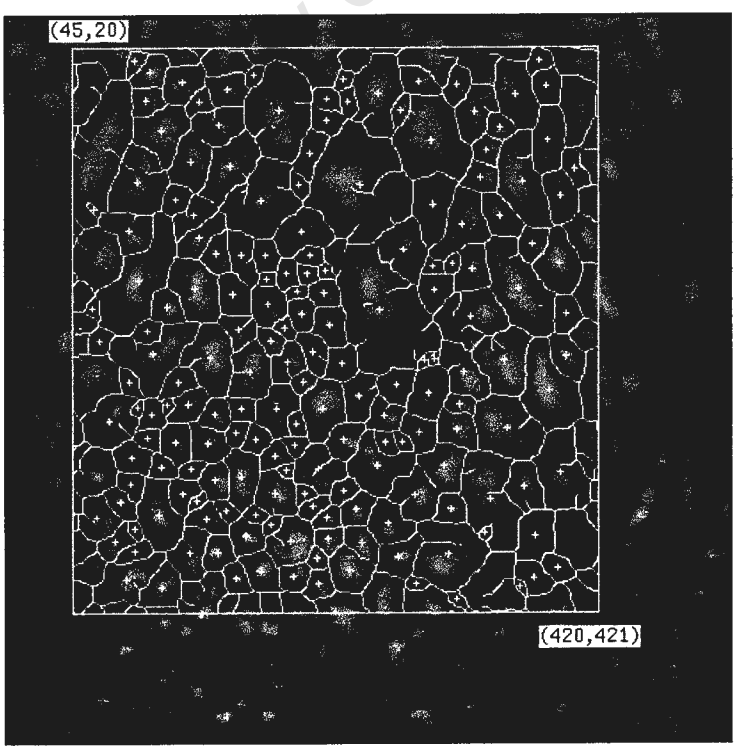


Figure 8.3 : Automatically segmented image, using the optimal DSSE of radius 10 pixels.

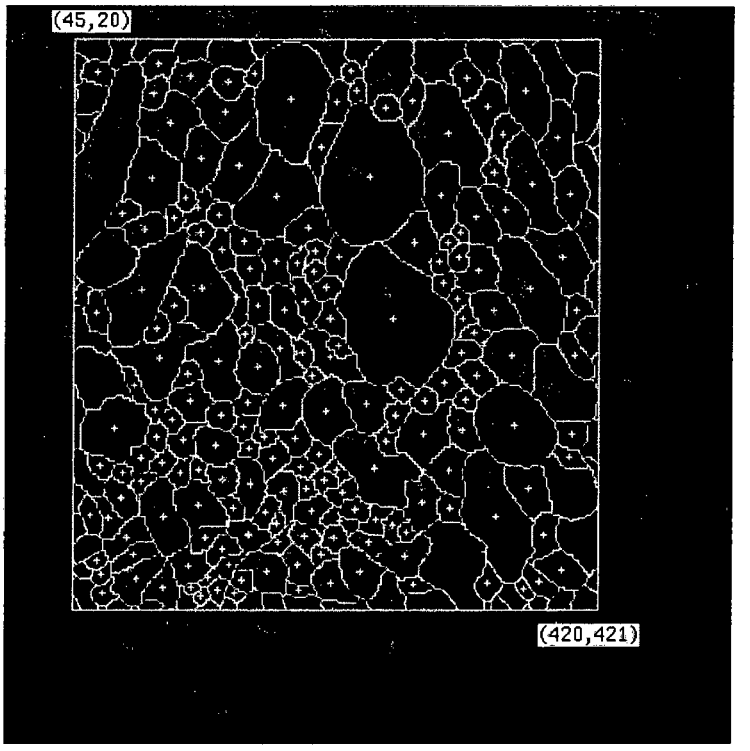


Figure 8.2 : Manually segmented image.

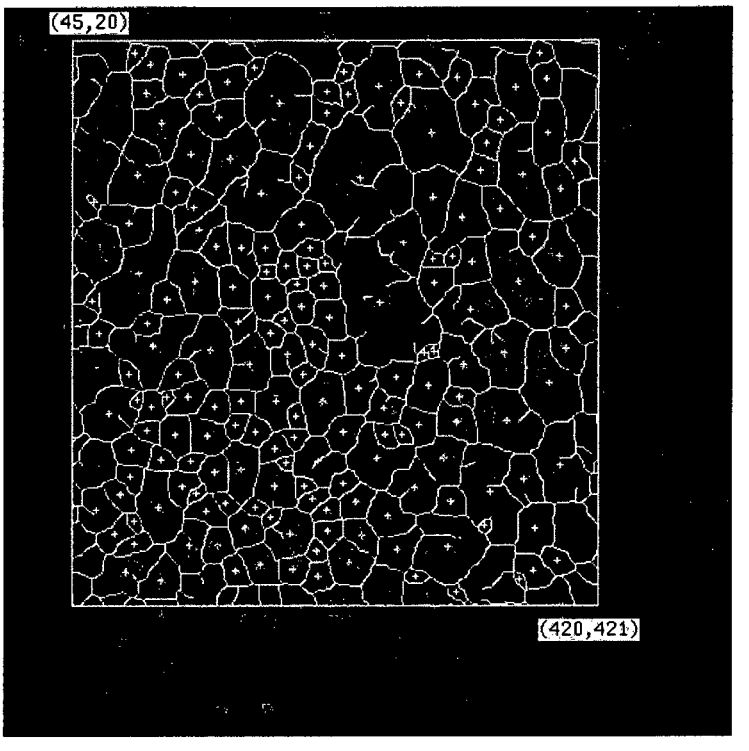


Figure 8.3 : Automatically segmented image, using the optimal DSSE of radius 10 pixels.

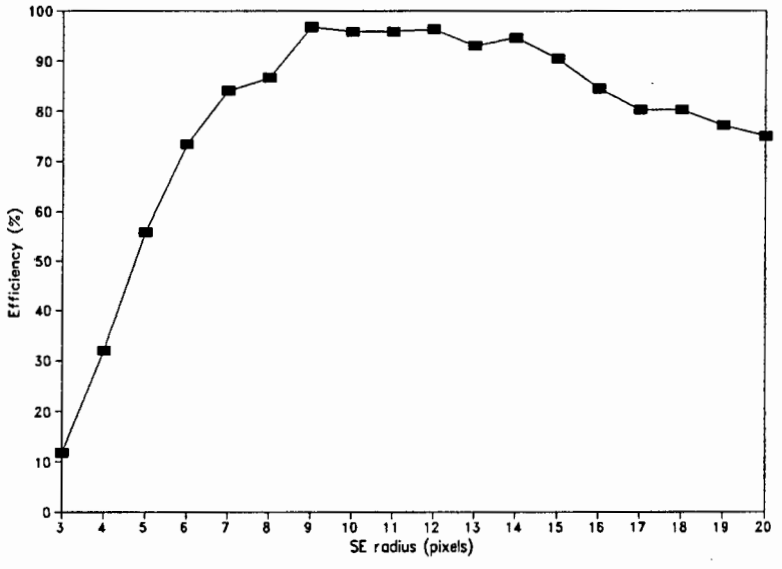


Figure 8.4 : Segmentation efficiency curve.

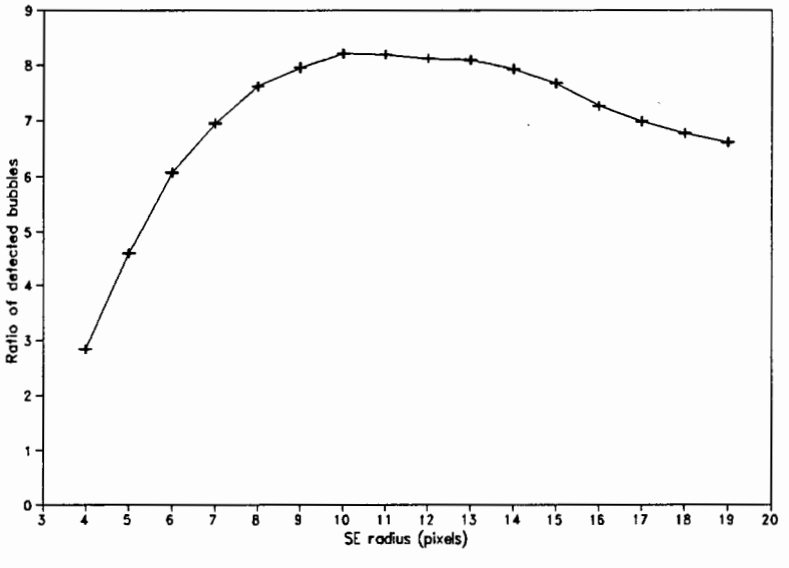
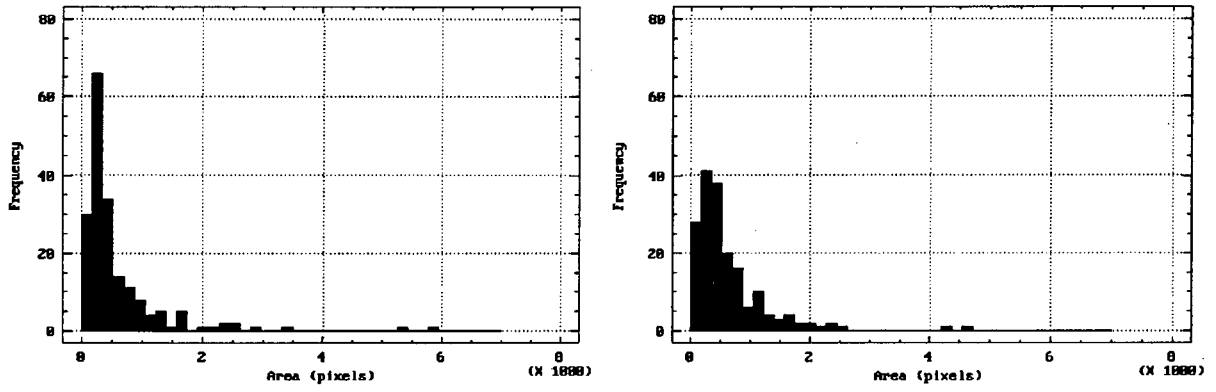
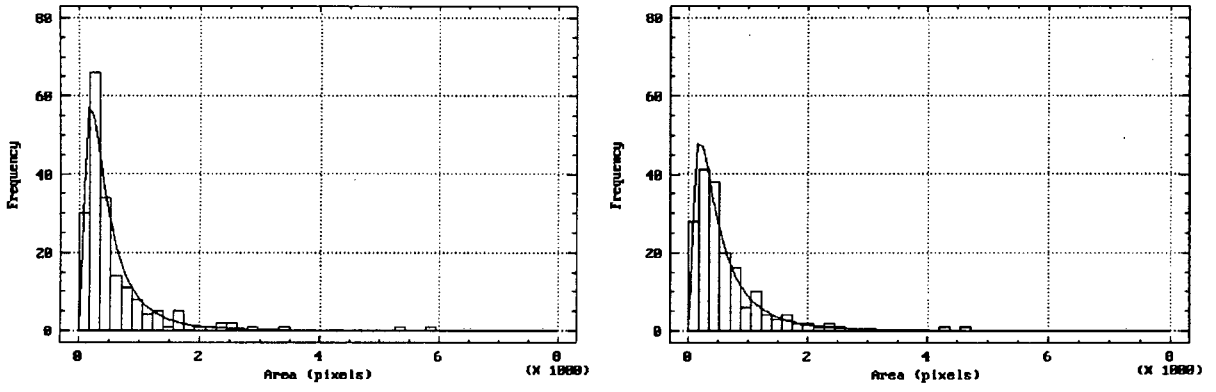


Figure 8.5 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.6 : (a) Expected size distribution with μ = 590.21, σ = 763.48, skewness = 3.99, Q_{50} = 335.00 and mode = 163.00.
(b) Observed size distribution with μ = 632.43, σ = 645.49, skewness = 3.06, Q_{50} = 432.50 and mode = 416.00.

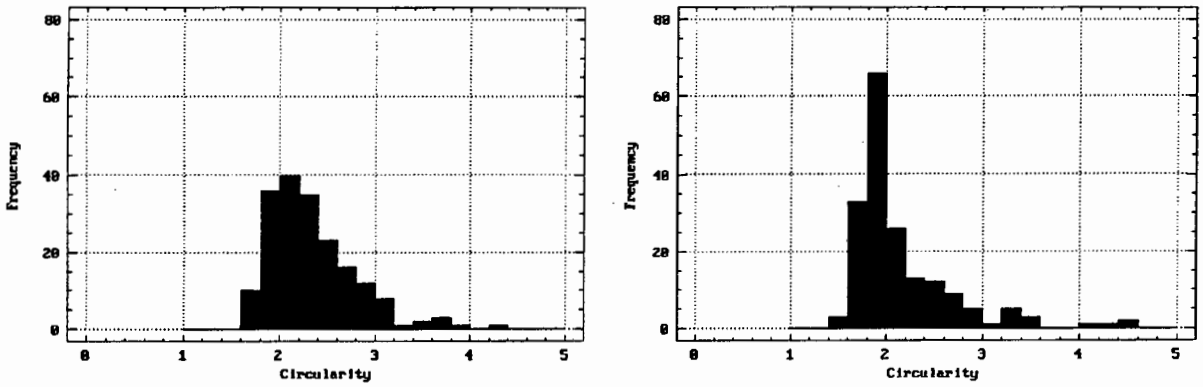


(a) (b)

Figure 8.7 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 552.44 and σ = 574.81.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 645.10 and σ = 724.40.

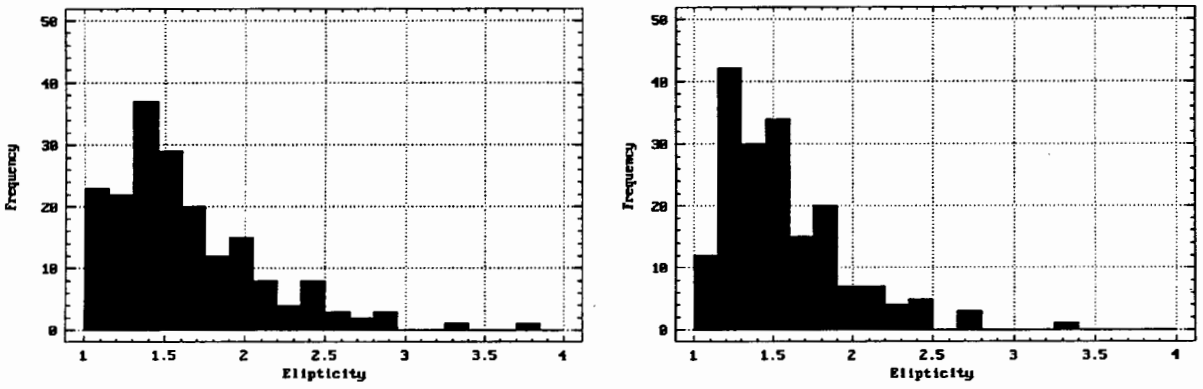
Figure	χ^2 tests	Conclusion
8.6	Calculated : χ^2_8 = 25.08 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Reject H_0
8.7 (a)	Calculated : χ^2_6 = 11.80 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Accept H_0
8.7 (b)	Calculated : χ^2_7 = 5.56 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Accept H_0

Table 8.2 : χ^2 tests for Figures 8.6, 8.7 (a) and (b).



(a) (b)

Figure 8.8 : (a) Expected circularity distribution with μ = 2.35, σ = 0.46, skewness = 1.35, Q_{50} = 2.25 and mode = 2.08.
(b) Observed circularity distribution with μ = 2.16, σ = 0.53, skewness = 2.13, Q_{50} = 1.98 and mode = 1.99.



(a) (b)

Figure 8.9 : (a) Expected elipticity distribution with μ = 1.63, σ = 0.47, skewness = 1.38, Q_{50} = 1.52 and mode = 1.34.
(b) Observed elipticity distribution with μ = 1.56, σ = 0.38, skewness = 1.40, Q_{50} = 1.47 and mode = 1.29.

Figure	χ^2 tests	Conclusion
8.8	Calculated : χ^2_8 = 130.00 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Reject H_0
8.9	Calculated : χ^2_9 = 42.46 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Reject H_0

Table 8.3 : χ^2 tests for Figures 8.8 and 8.9.

8.2.2 BUB3.CFI

Figure 8.10 : BUB3.CFI - original image.

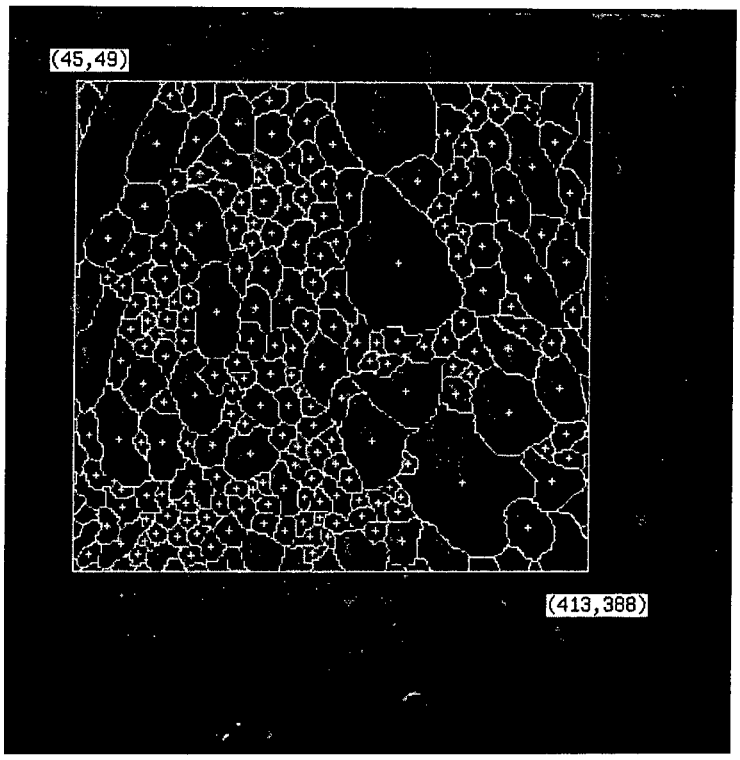


Figure 8.11 : Manually segmented image.

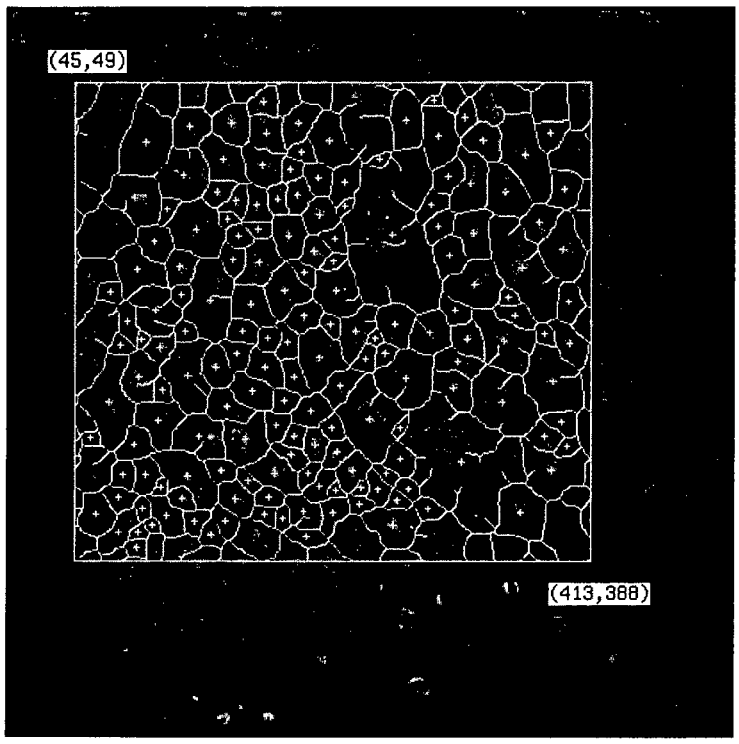


Figure 8.12 : Automatically segmented image, using the optimal DSSE of radius 11 pixels.

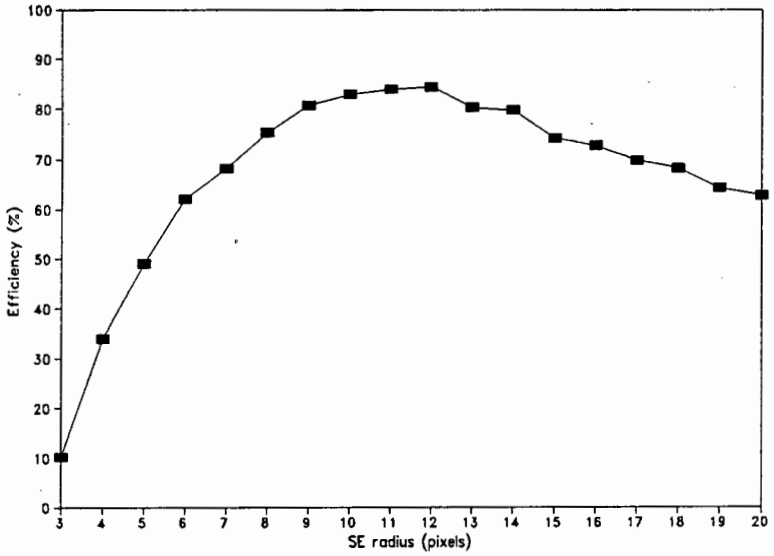


Figure 8.13 : Segmentation efficiency curve.

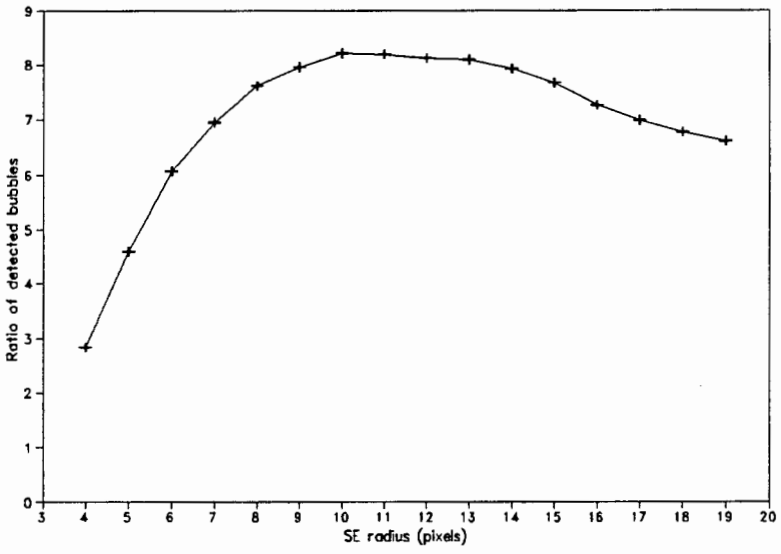
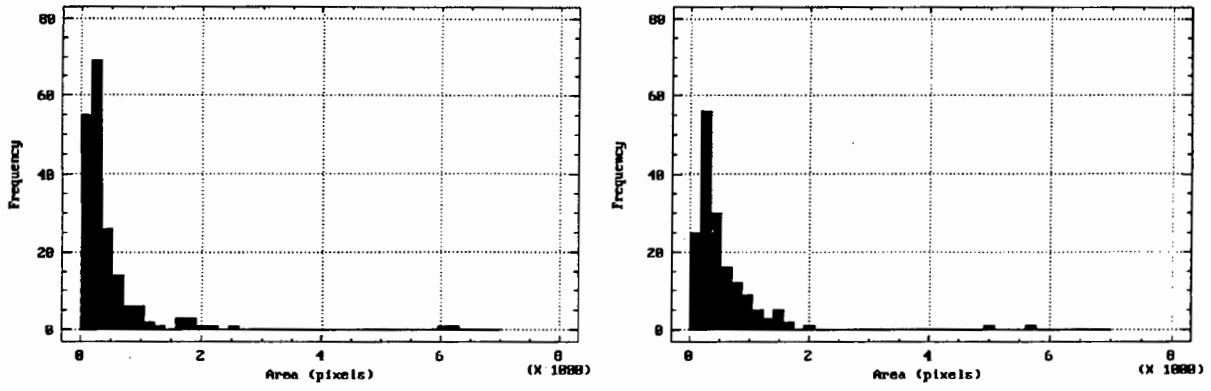
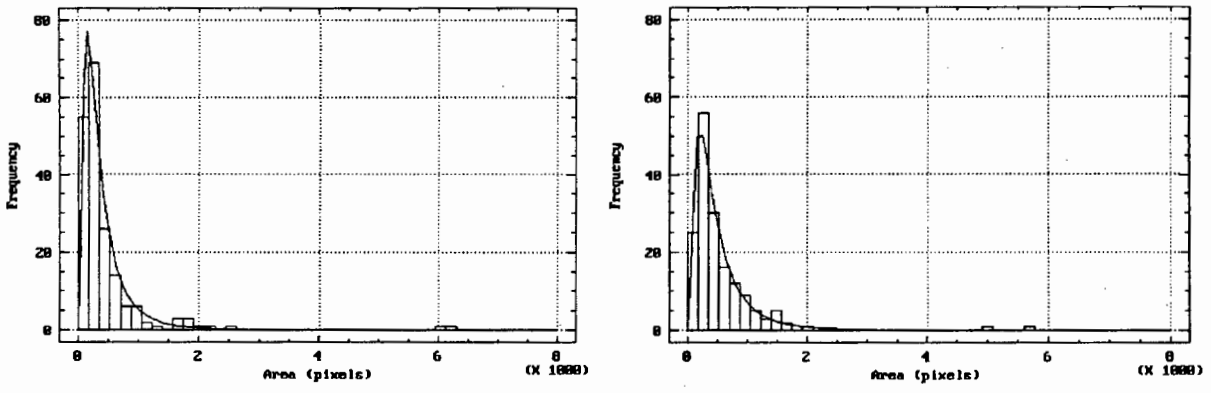


Figure 8.14 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.15 : (a) Expected size distribution with μ = 464.23, σ = 706.26, skewness = 5.70, Q_{50} = 252.00 and mode = 200.00.
(b) Observed size distribution with μ = 551.93, σ = 657.10, skewness = 5.09, Q_{50} = 357.50 and mode = 86.00.

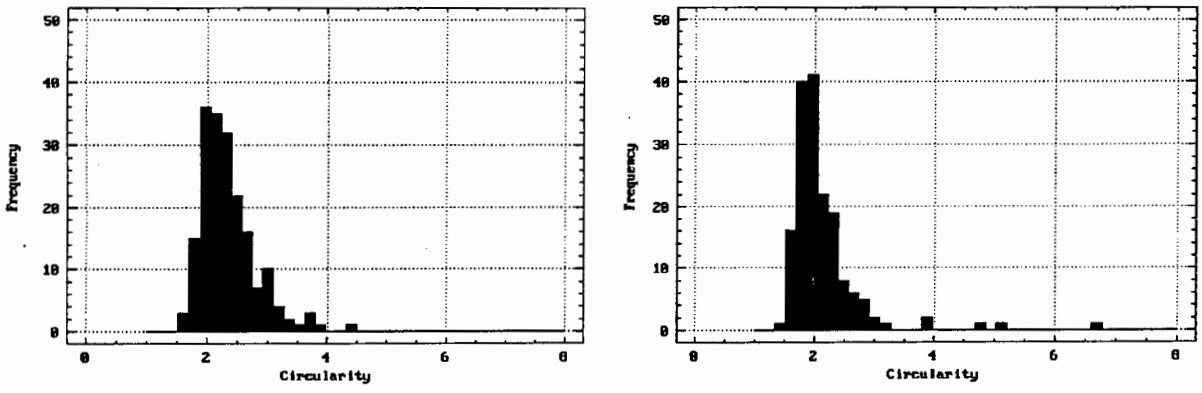


(a) (b)

Figure 8.16 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 423.04 and σ = 431.73.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 535.77 and σ = 517.09.

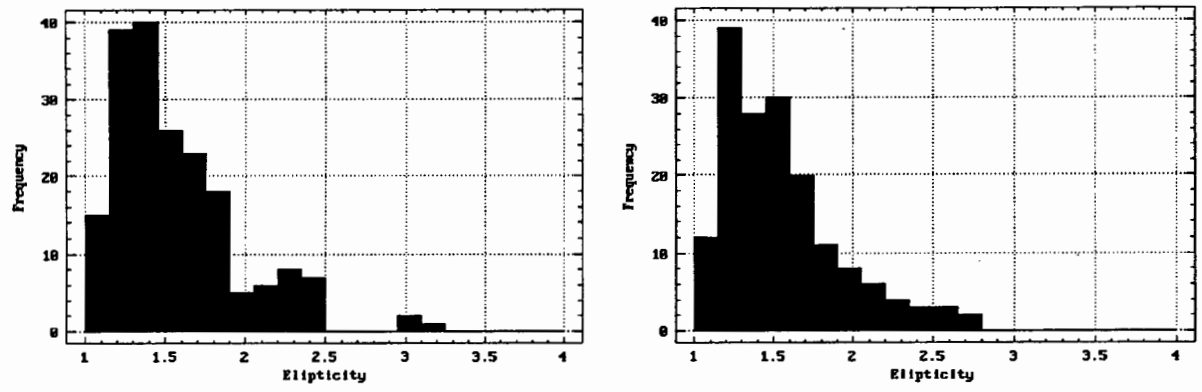
Figure	χ^2 tests	Conclusion
8.15	Calculated : χ^2_6 = 25.73 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Reject H_0
8.16 (a)	Calculated : χ^2_5 = 9.45 Tabulated : $\chi^2_{5;0.05}$ = 11.07	Accept H_0
8.16 (b)	Calculated : χ^2_6 = 4.08 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Accept H_0

Table 8.4 : χ^2 tests for Figures 8.15, 8.16 (a) and (b).



(a) (b)

Figure 8.17 : (a) Expected circularity distribution with μ = 2.27, σ = 0.35, skewness = 0.81, Q_{50} = 2.22 and mode = 2.06.
(b) Observed circularity distribution with μ = 2.13, σ = 0.61, skewness = 4.15, Q_{50} = 1.98 and mode = 1.80.



(a) (b)

Figure 8.18 : (a) Expected elipticity distribution with μ = 1.58, σ = 0.40, skewness = 1.37, Q_{50} = 1.47 and mode = 1.35.
(b) Observed elipticity distribution with μ = 1.55, σ = 0.37, skewness = 1.14, Q_{50} = 1.47 and mode = 1.25.

Figure	χ^2 tests	Conclusion
8.17	Calculated : χ^2_7 = 136.70 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Reject H_0
8.18	Calculated : χ^2_9 = 9.15 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Accept H_0

Table 8.5 : χ^2 tests for Figures 8.17 and 8.18.

8.2.3 BUB4.CFI

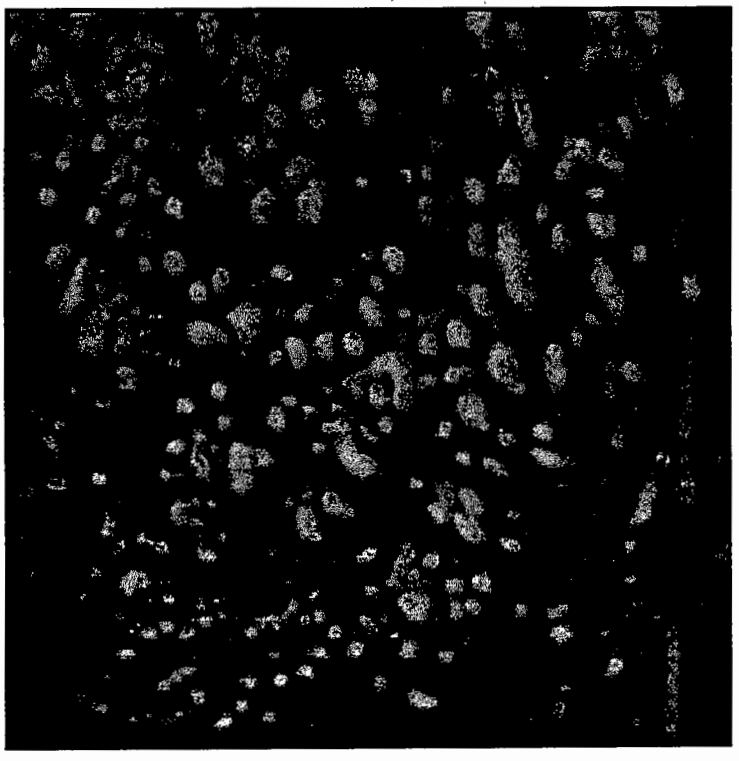


Figure 8.19 : BUB4.CFI - original image.

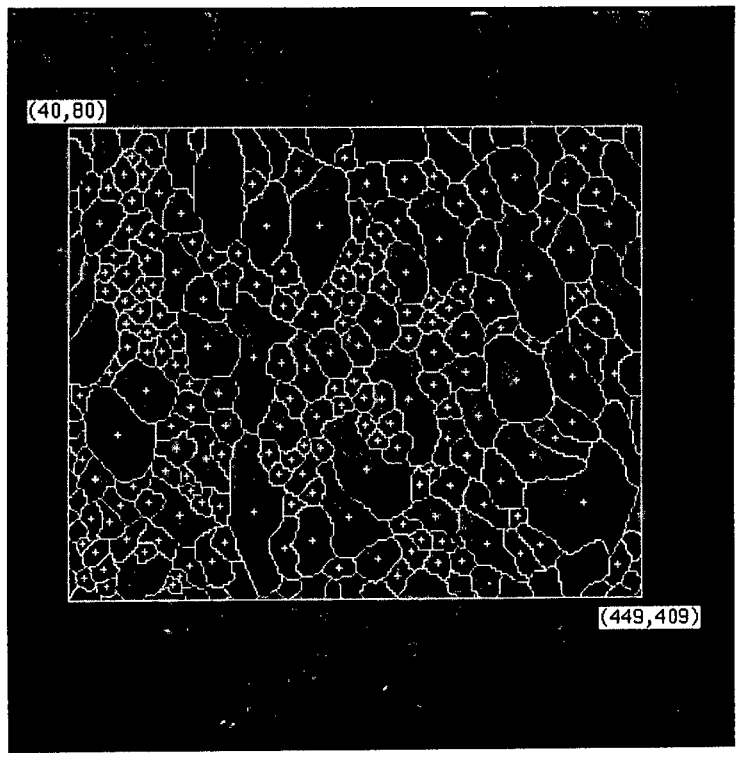


Figure 8.20 : Manually segmented image.

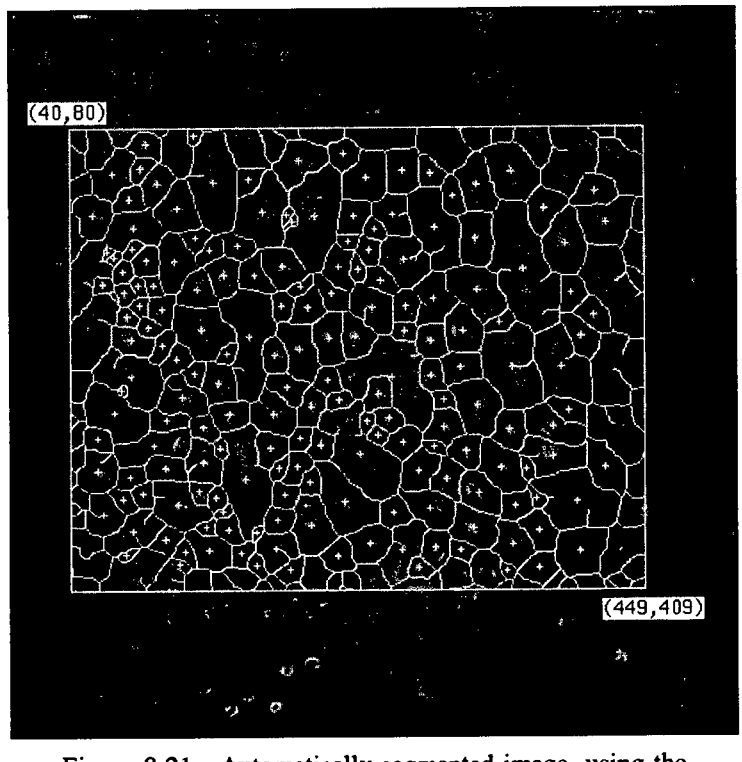


Figure 8.21 : Automatically segmented image, using the optimal DSSE of radius 10 pixels.

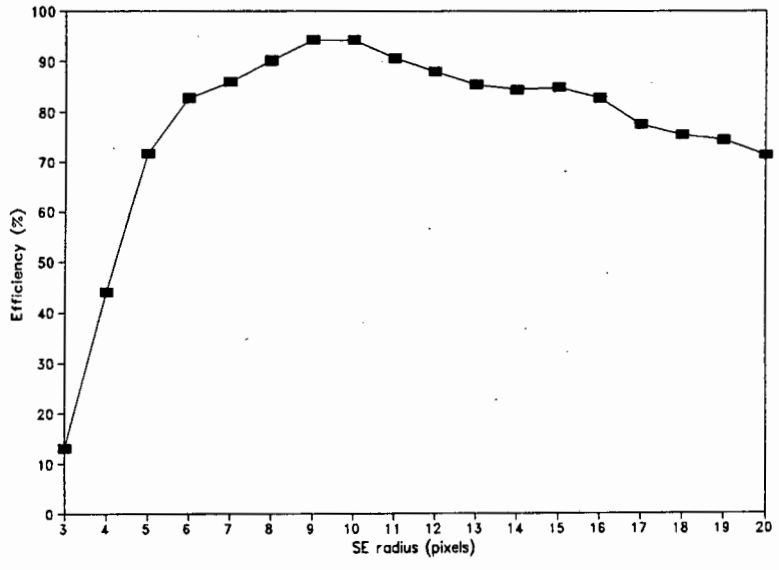


Figure 8.22 : Segmentation efficiency curve.

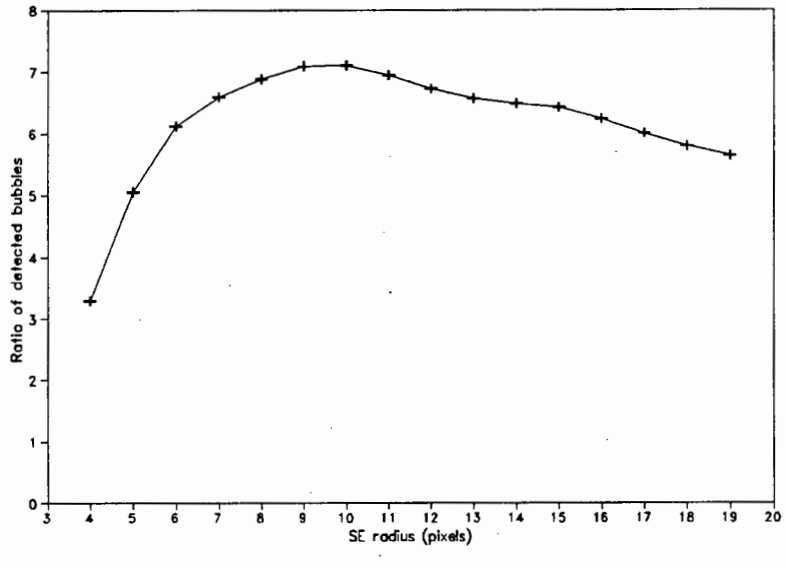
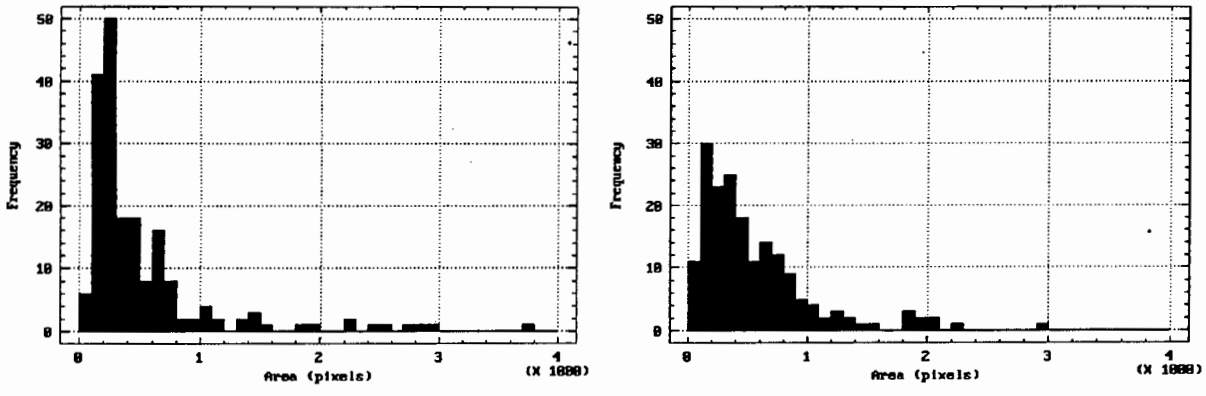
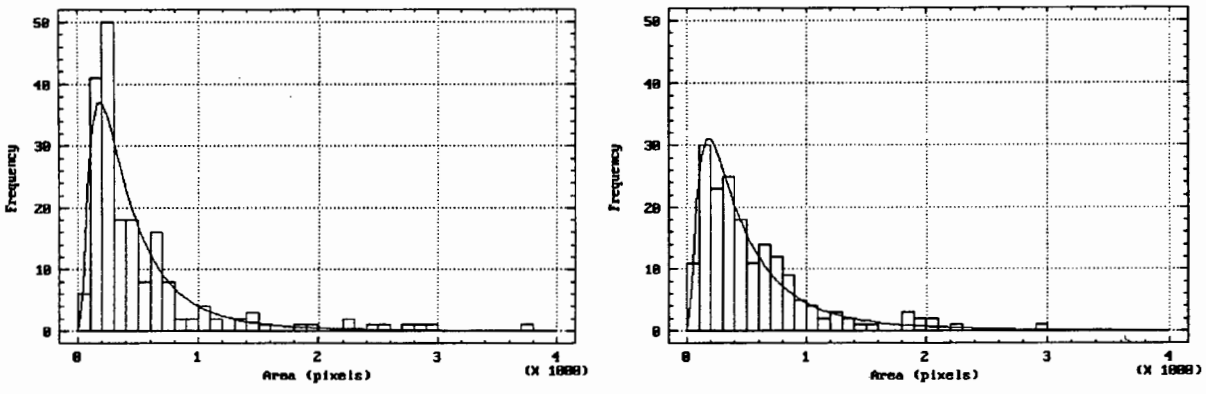


Figure 8.23 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.24 : (a) Expected size distribution with μ = 513.51, σ = 582.31, skewness = 2.91, Q_{50} = 295.00 and mode = 219.00.
(b) Observed size distribution with μ = 548.16, σ = 483.36, skewness = 1.99, Q_{50} = 410.50 and mode = 235.00.

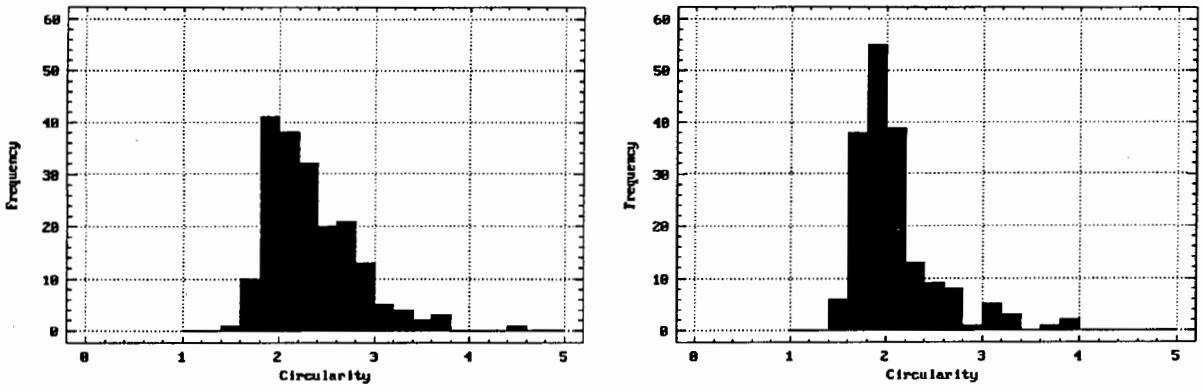


(a) (b)

Figure 8.25 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 488.10 and σ = 476.49.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 568.95 and σ = 616.70.

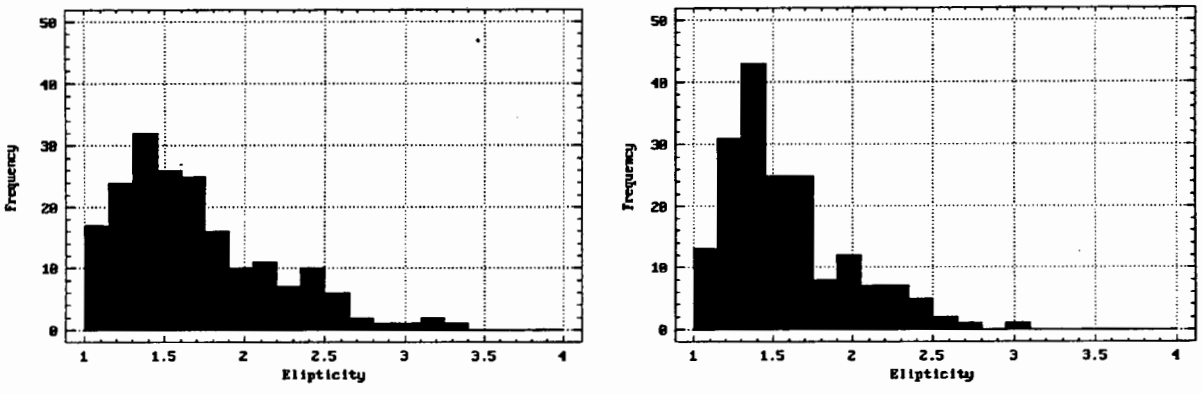
Figure	χ^2 tests	Conclusion
8.24	Calculated : χ^2_{10} = 42.44 Tabulated : $\chi^2_{10;0.05}$ = 18.31	Reject H_0
8.25 (a)	Calculated : χ^2_{10} = 29.88 Tabulated : $\chi^2_{10;0.05}$ = 18.31	Reject H_0
8.25 (b)	Calculated : χ^2_{11} = 6.56 Tabulated : $\chi^2_{11;0.05}$ = 19.68	Accept H_0

Table 8.6 : χ^2 tests for Figures 8.24, 8.25 (a) and (b).



(a) (b)

Figure 8.26 : (a) Expected circularity distribution with μ = 2.34, σ = 0.46, skewness = 1.27, Q_{50} = 2.23 and mode = 1.86.
(b) Observed circularity distribution with μ = 2.07, σ = 0.43, skewness = 1.93, Q_{50} = 1.97 and mode = 1.84.



(a) (b)

Figure 8.27 : (a) Expected elipticity distribution with μ = 1.69, σ = 0.48, skewness = 1.04, Q_{50} = 1.57 and mode = 1.19.
(b) Observed elipticity distribution with μ = 1.58, σ = 0.38, skewness = 1.13, Q_{50} = 1.46 and mode = 1.37.

Figure	χ^2 tests	Conclusion
8.26	Calculated : χ^2_7 = 146.97 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Reject H_0
8.27	Calculated : χ^2_9 = 21.88 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Reject H_0

Table 8.7 : χ^2 tests for Figures 8.26 and 8.27.

8.2.4 BUB5.CFI

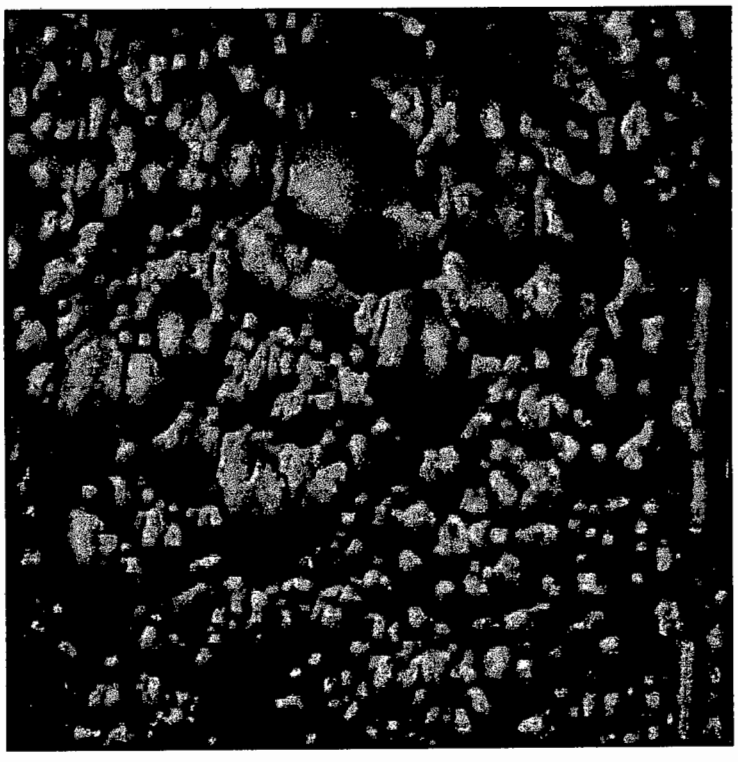


Figure 8.28 : BUB5.CFI - original image.

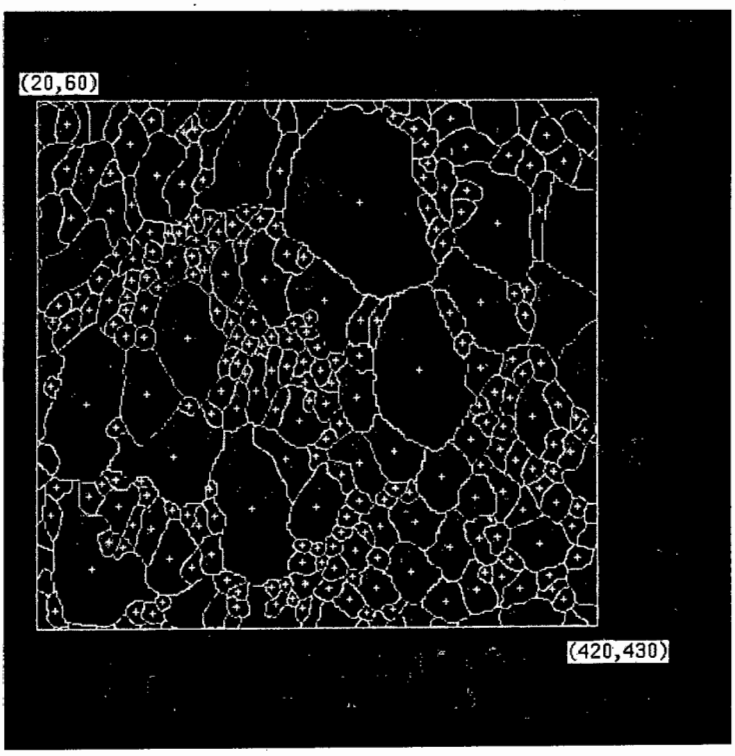


Figure 8.29 : Manually segmented image.

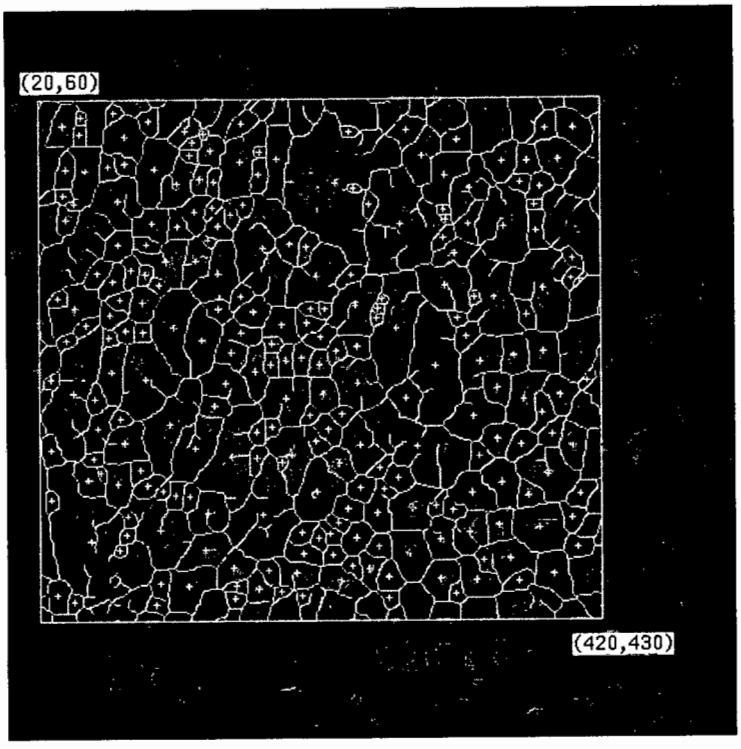


Figure 8.30 : Automatically segmented image, using the optimal DSSE of radius 10 pixels.

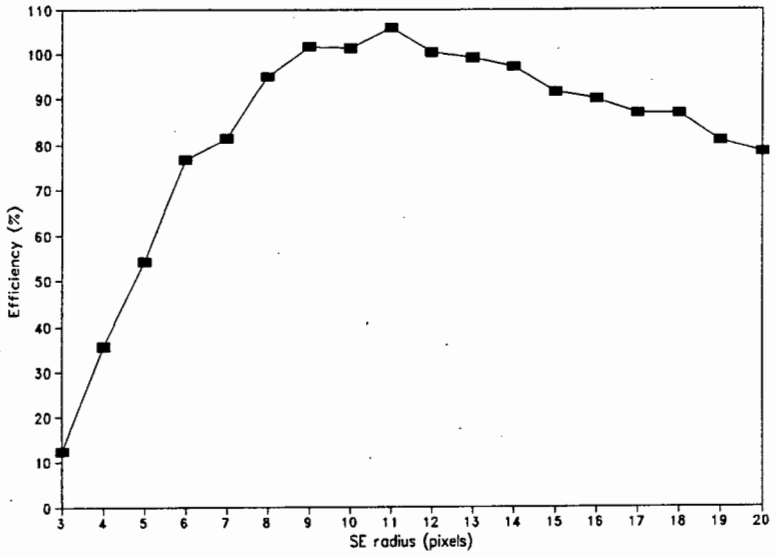


Figure 8.31 : Segmentation efficiency curve.

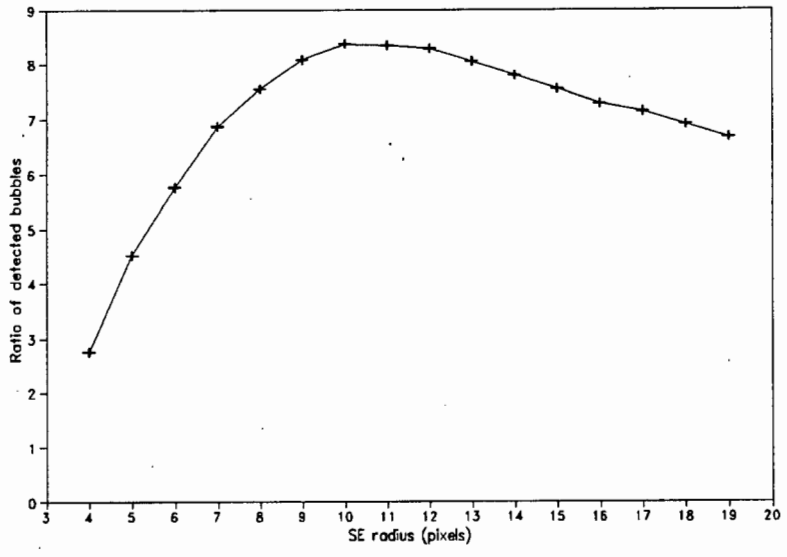
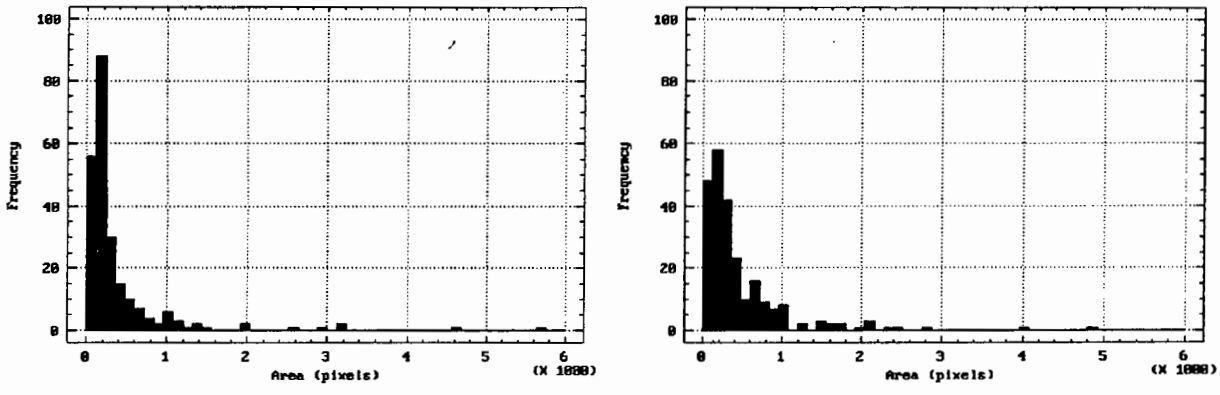
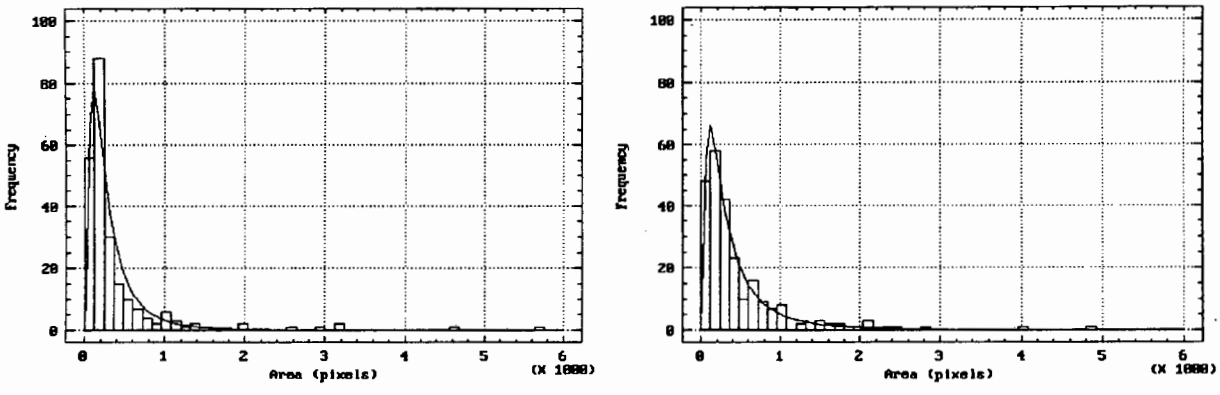


Figure 8.32 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.33 : (a) Expected size distribution with μ = 467.44, σ = 973.27, skewness = 6.25, Q_{50} = 201.00 and mode = 112.00.
(b) Observed size distribution with μ = 469.33, σ = 600.40, skewness = 3.70, Q_{50} = 261.00 and mode = 110.00.

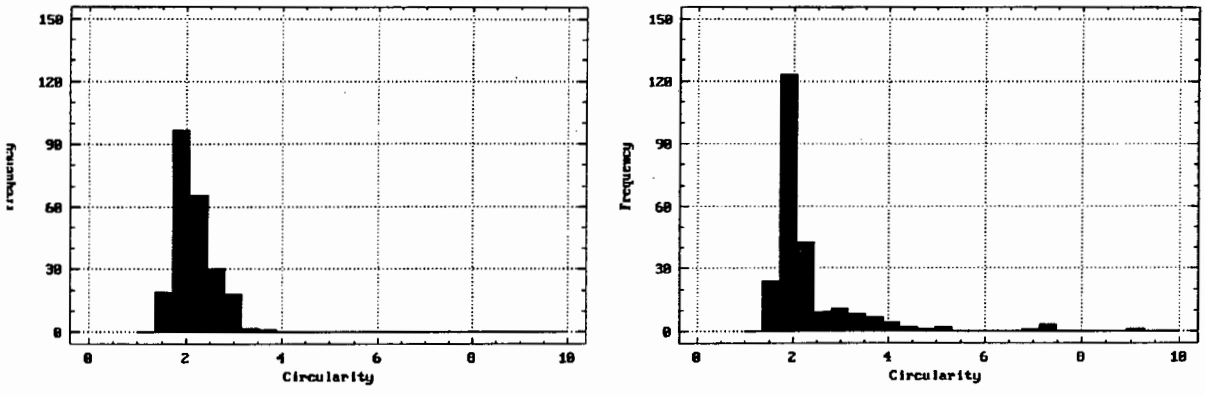


(a) (b)

Figure 8.34 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 383.14 and σ = 469.55.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 459.64 and σ = 584.17.

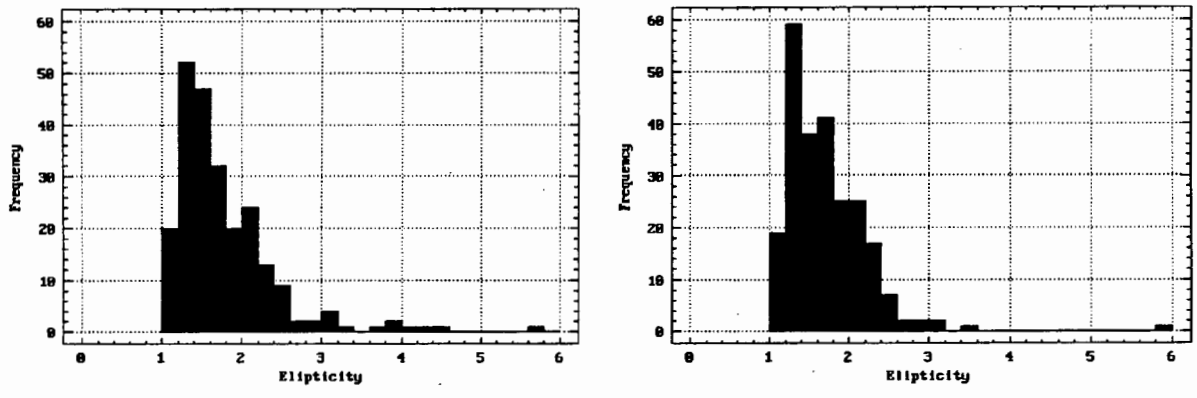
Figure	χ^2 tests	Conclusion
8.33	Calculated : χ^2_9 = 49.75 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Reject H_0
8.34 (a)	Calculated : χ^2_8 = 27.47 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Reject H_0
8.34 (b)	Calculated : χ^2_9 = 8.89 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Accept H_0

Table 8.8 : χ^2 tests for Figures 8.33, 8.34 (a) and (b).



(a) (b)

Figure 8.35 : (a) Expected circularity distribution with μ = 2.18, σ = 0.39, skewness = 0.88, Q_{50} = 2.11 and mode = 1.93.
(b) Observed circularity distribution with μ = 2.33, σ = 1.02, skewness = 3.50, Q_{50} = 1.99 and mode = 1.78.



(a) (b)

Figure 8.36 : (a) Expected elipticity distribution with μ = 1.77, σ = 0.64, skewness = 2.42, Q_{50} = 1.59 and mode = 1.27.
(b) Observed elipticity distribution with μ = 1.70, σ = 0.51, skewness = 2.90, Q_{50} = 1.62 and mode = 1.32.

Figure	χ^2 tests	Conclusion
8.35	Calculated : χ^2_4 = 46.72 Tabulated : $\chi^2_{4;0.05}$ = 9.49	Reject H_0
8.36	Calculated : χ^2_8 = 12.21 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Accept H_0

Table 8.9 : χ^2 tests for Figures 8.35 and 8.36.

8.2.5 BUB6.CFI

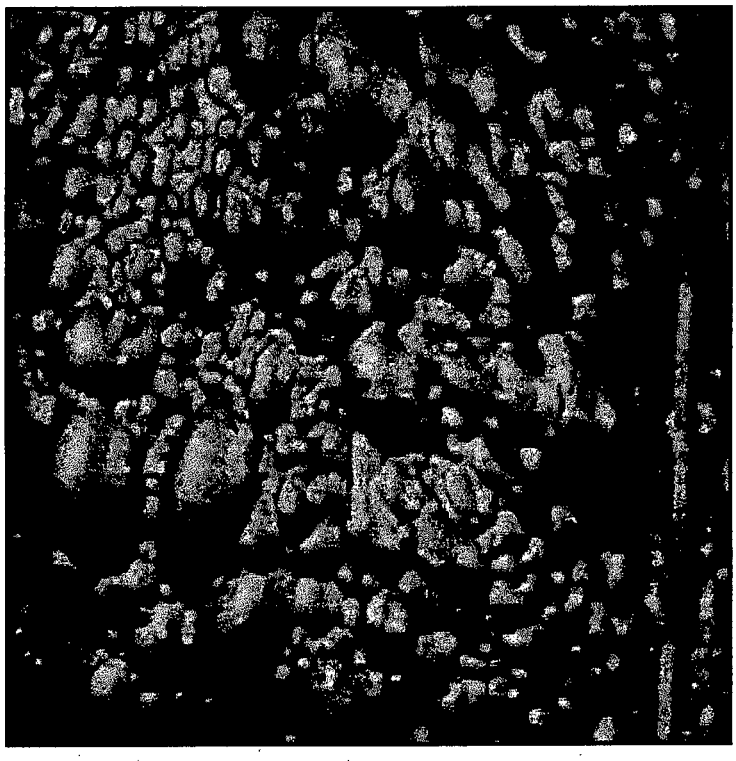


Figure 8.37 : BUB6.CFI - original image.

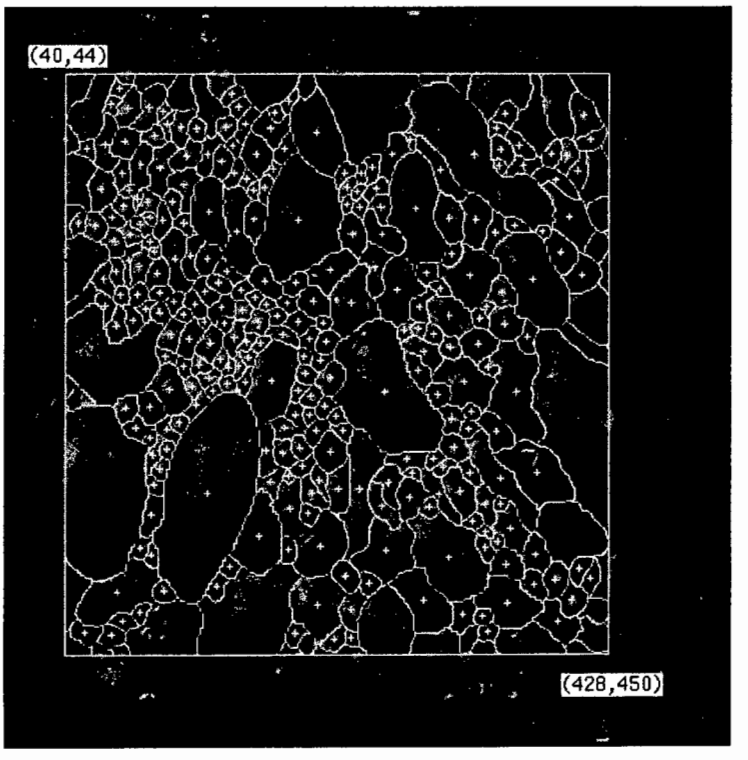


Figure 8.38 : Manually segmented image.

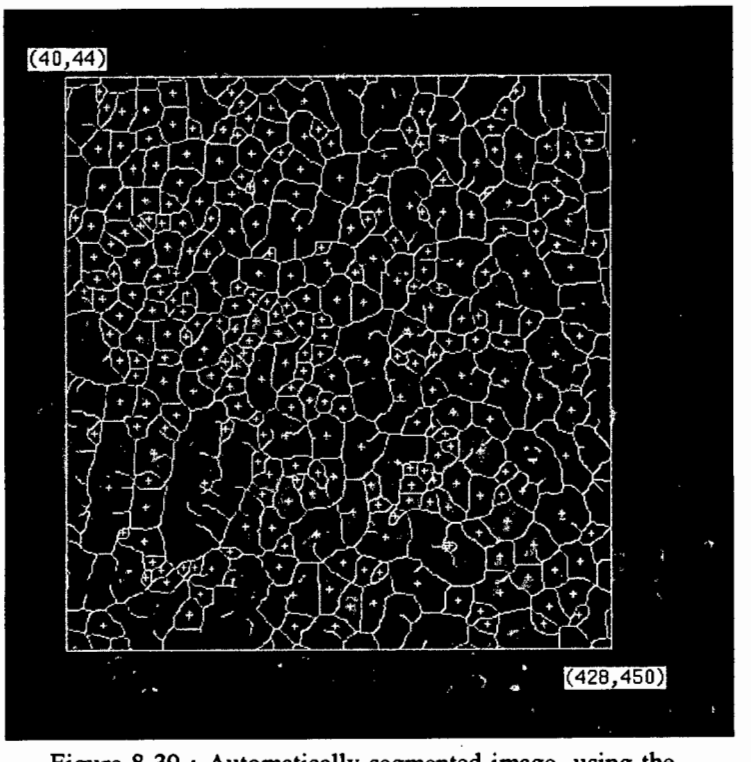


Figure 8.39 : Automatically segmented image, using the optimal DSSE of radius 10 pixels.

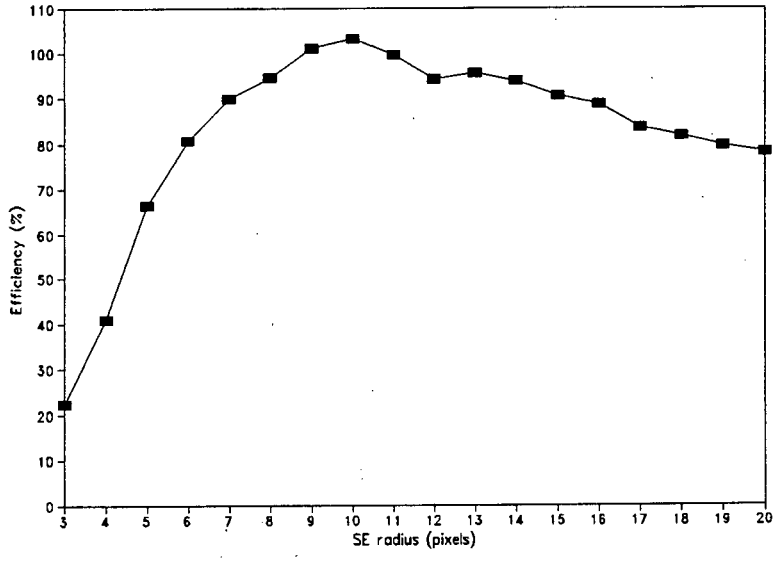


Figure 8.40 : Segmentation efficiency curve.

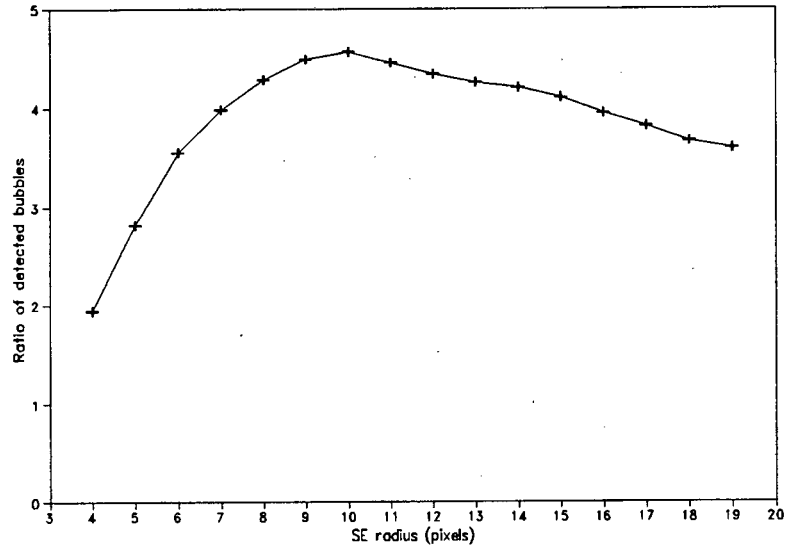
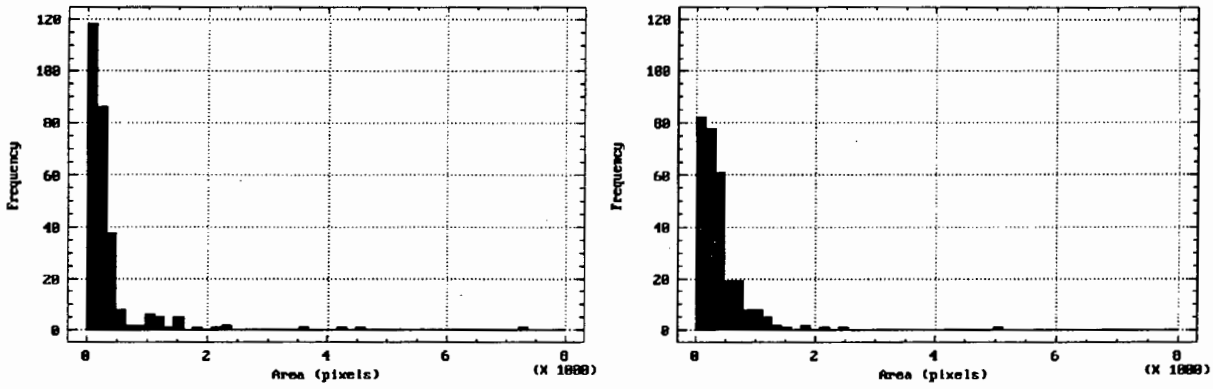
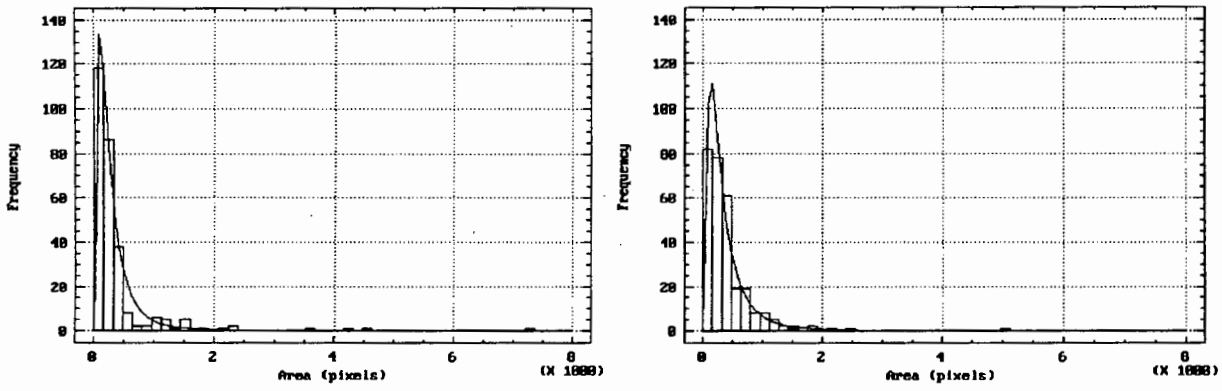


Figure 8.41 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.42 : (a) Expected size distribution with μ = 372.13, σ = 677.51, skewness = 6.09, Q_{50} = 181.00 and mode = 100.00.
(b) Observed size distribution with μ = 391.61, σ = 441.22, skewness = 5.01, Q_{50} = 275.00 and mode = 68.00.

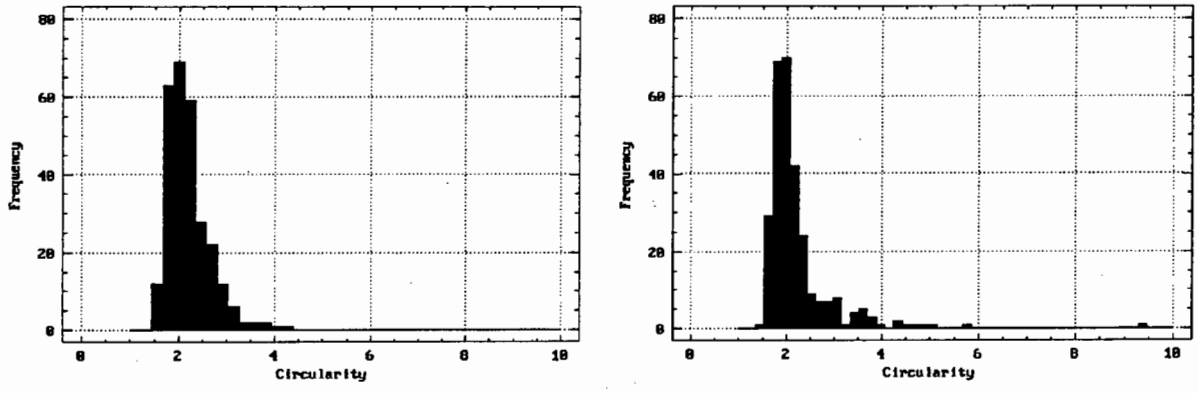


(a) (b)

Figure 8.43 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 320.43 and σ = 349.60.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 390.03 and σ = 416.34.

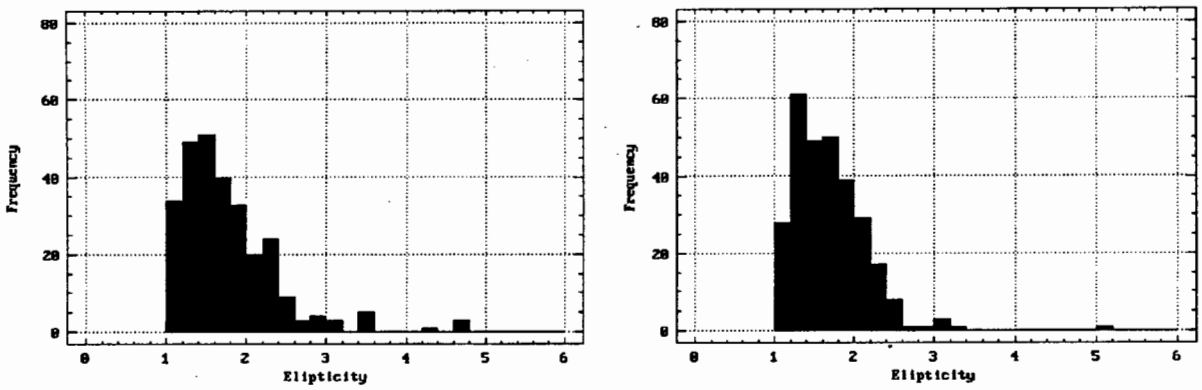
Figure	χ^2 tests	Conclusion
8.42	Calculated : χ^2_6 = 102.47 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Reject H_0
8.43 (a)	Calculated : χ^2_5 = 33.74 Tabulated : $\chi^2_{5;0.05}$ = 11.07	Reject H_0
8.43 (b)	Calculated : χ^2_6 = 8.70 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Accept H_0

Table 8.10 : χ^2 tests for Figures 8.42, 8.43 (a) and (b).



(a) (b)

Figure 8.44 : (a) Expected circularity distribution with μ = 2.20, σ = 0.44, skewness = 1.51, Q_{50} = 2.11 and mode = 1.97.
(b) Observed circularity distribution with μ = 2.22, σ = 0.75, skewness = 4.38, Q_{50} = 2.03 and mode = 1.86.



(a) (b)

Figure 8.45 : (a) Expected elipticity distribution with μ = 1.77, σ = 0.61, skewness = 1.99, Q_{50} = 1.63 and mode = 1.19.
(b) Observed elipticity distribution with μ = 1.69, σ = 0.46, skewness = 2.12, Q_{50} = 1.64 and mode = 1.49.

Figure	χ^2 tests	Conclusion
8.44	Calculated : χ^2_8 = 51.08 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Reject H_0
8.45	Calculated : χ^2_8 = 20.55 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Reject H_0

Table 8.11 : χ^2 tests for Figures 8.44 and 8.45.

8.2.6 BUB7.CFI

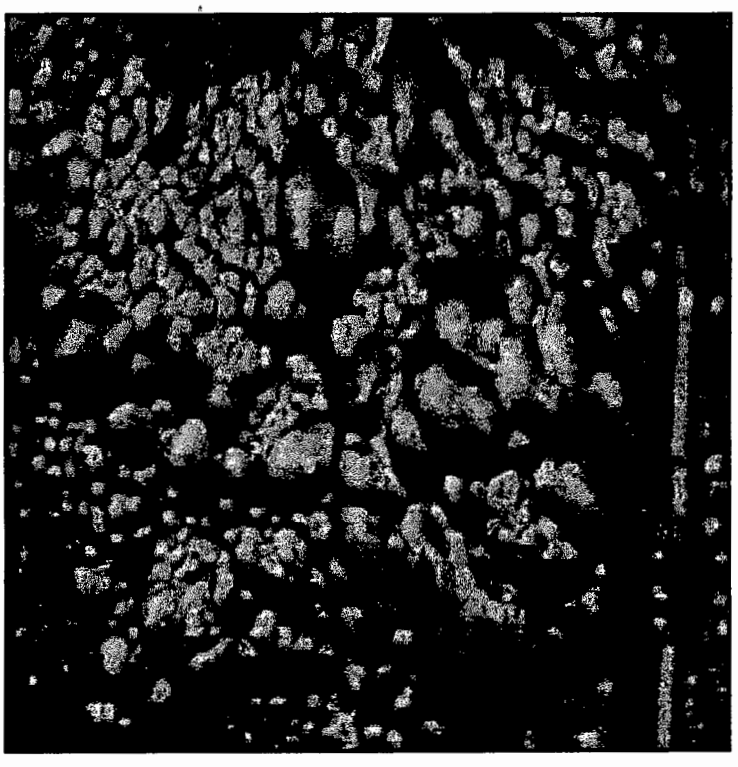


Figure 8.46 : BUB7.CFI - original image.

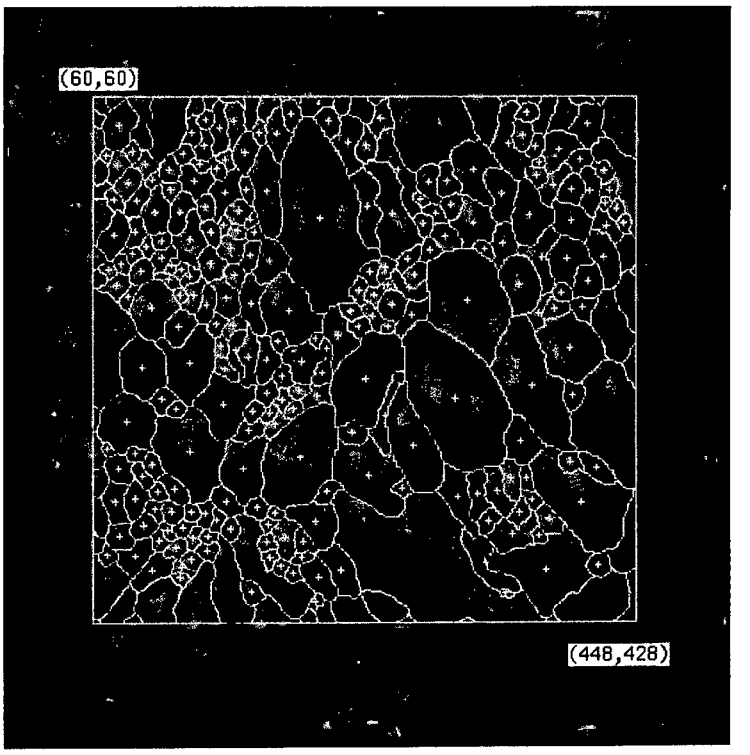


Figure 8.47 : Manually segmented image.

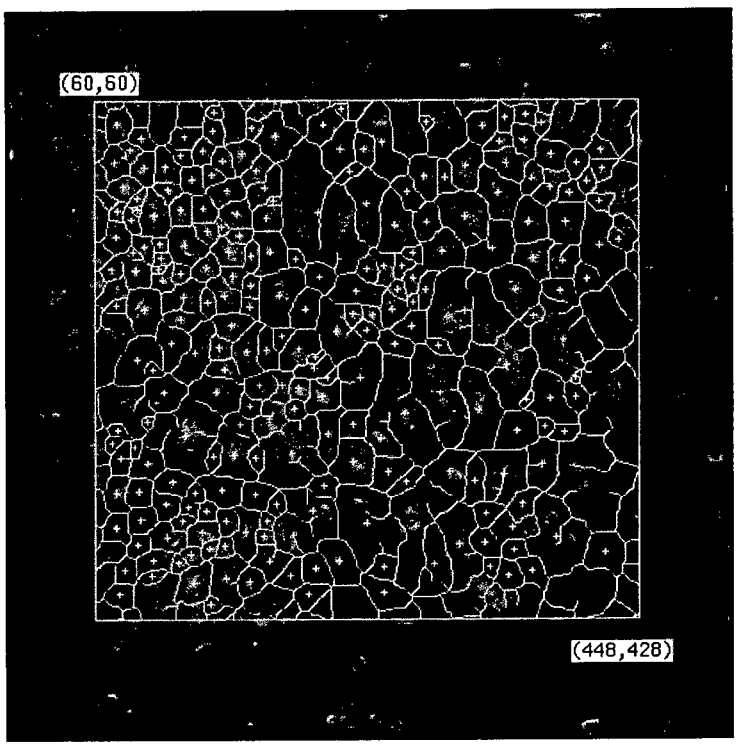


Figure 8.48 : Automatically segmented image, using the optimal DSSE of radius 12 pixels.

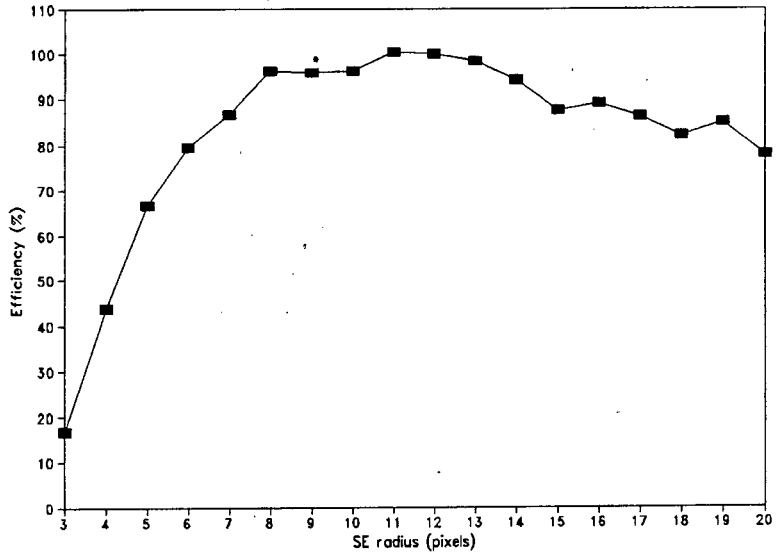


Figure 8.49 : Segmentation efficiency curve.

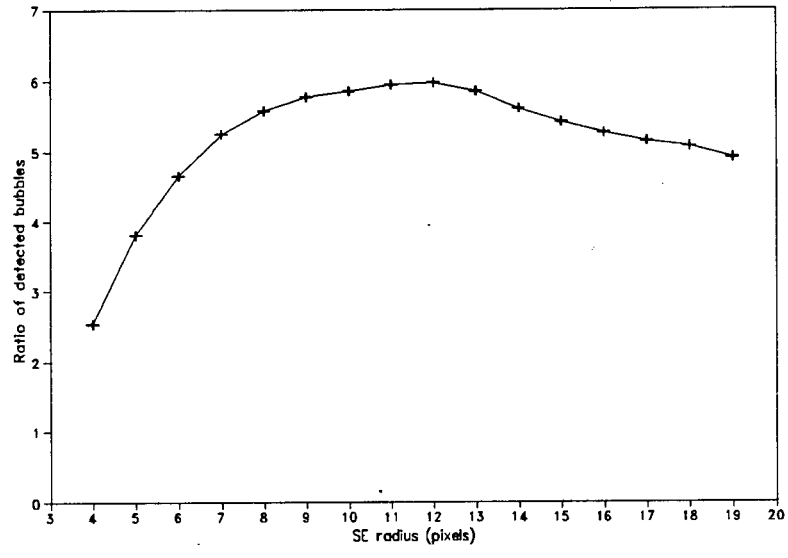
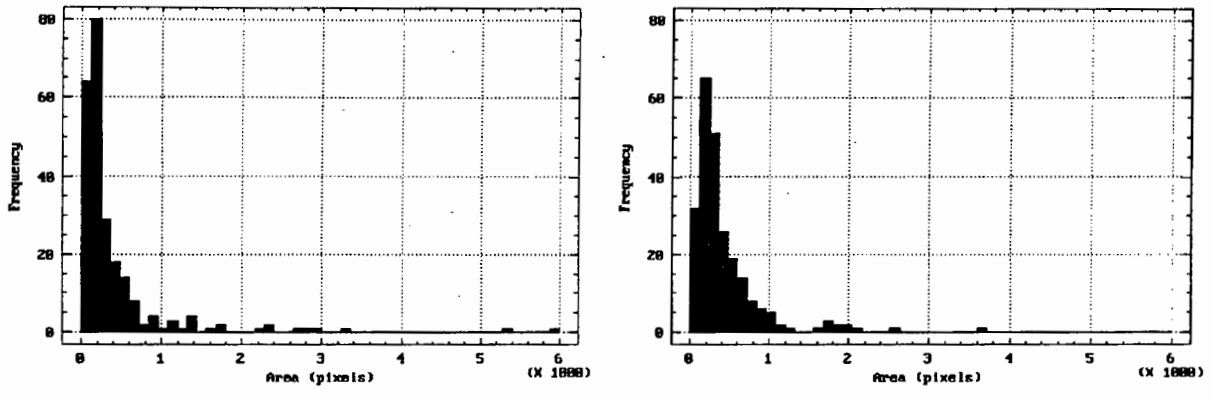
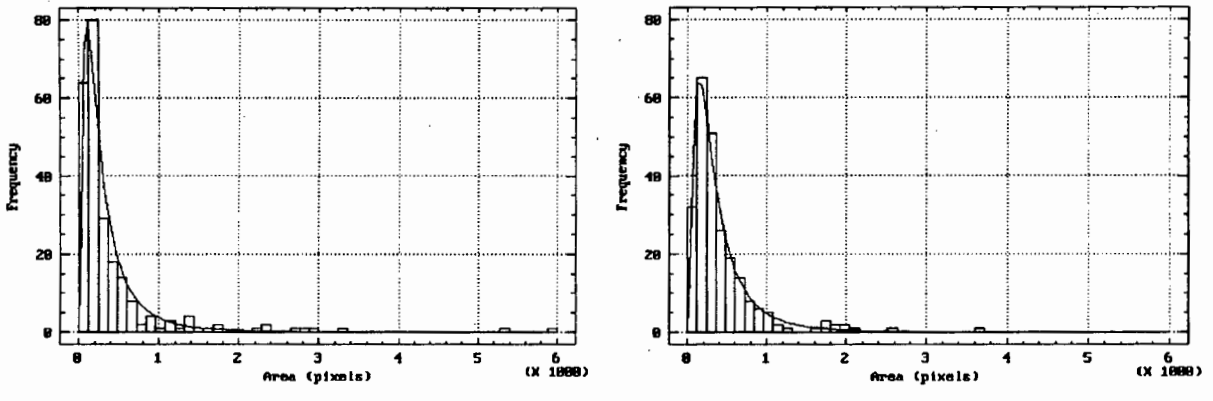


Figure 8.50 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.51 : (a) Expected size distribution with μ = 414.73, σ = 695.67, skewness = 4.75, Q_{50} = 195.00 and mode = 112.00.
(b) Observed size distribution with μ = 428.45, σ = 450.09, skewness = 3.17, Q_{50} = 298.00 and mode = 158.00.

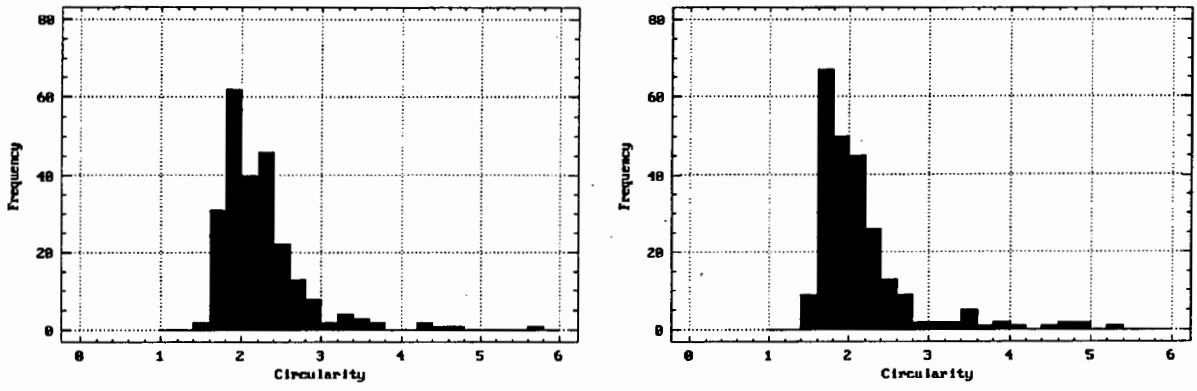


(a) (b)

Figure 8.52 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 364.54 and σ = 436.58.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 425.05 and σ = 437.26.

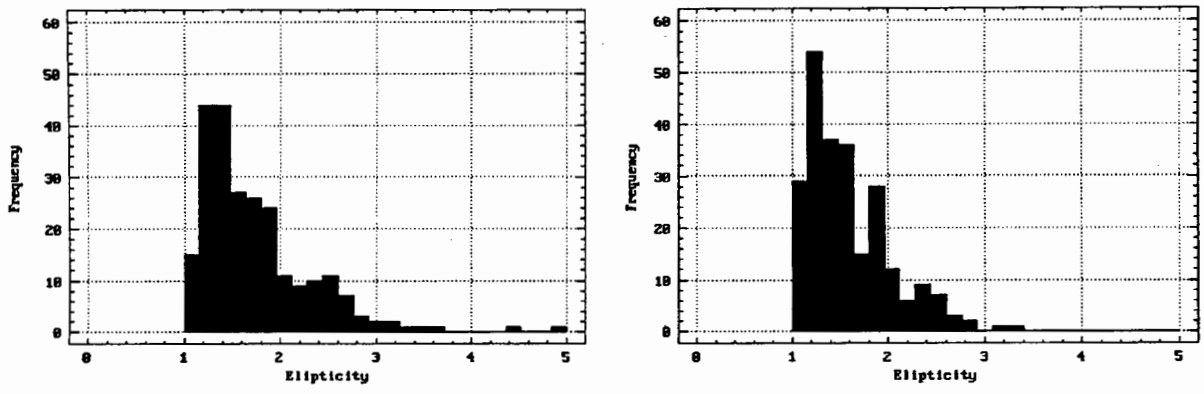
Figure	χ^2 tests	Conclusion
8.51	Calculated : χ^2_8 = 56.21 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Reject H_0
8.52 (a)	Calculated : χ^2_7 = 19.24 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Reject H_0
8.52 (b)	Calculated : χ^2_8 = 9.90 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Accept H_0

Table 8.12 : χ^2 tests for Figures 8.51, 8.52 (a) and (b).



(a) (b)

Figure 8.53 : (a) Expected circularity distribution with μ = 2.24, σ = 0.55, skewness = 2.55, Q_{50} = 2.16 and mode = 2.28.
(b) Observed circularity distribution with μ = 2.15, σ = 0.63, skewness = 2.57, Q_{50} = 1.97 and mode = 1.74.



(a) (b)

Figure 8.54 : (a) Expected elipticity distribution with μ = 1.74, σ = 0.59, skewness = 1.83, Q_{50} = 1.60 and mode = 1.43.
(b) Observed elipticity distribution with μ = 1.60, σ = 0.43, skewness = 1.16, Q_{50} = 1.49 and mode = 1.31.

Figure	χ^2 tests	Conclusion
8.53	Calculated : χ^2_7 = 79.66 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Reject H_0
8.54	Calculated : χ^2_{10} = 35.28 Tabulated : $\chi^2_{10;0.05}$ = 18.31	Reject H_0

Table 8.13 : χ^2 tests for Figures 8.53 and 8.54.

8.2.7 BUB8.CFI

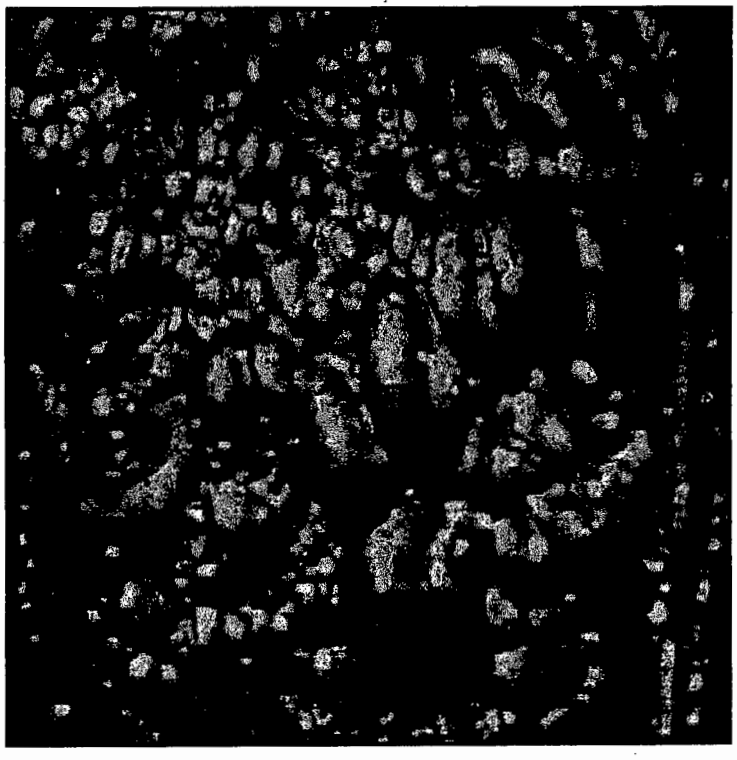


Figure 8.55 : BUB8.CFI - original image.

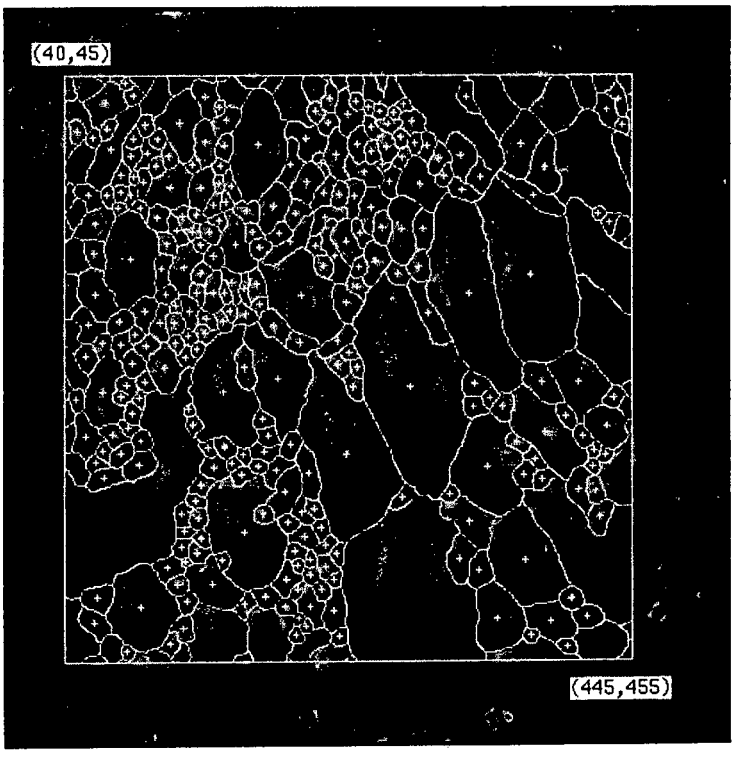


Figure 8.56 : Manually segmented image.

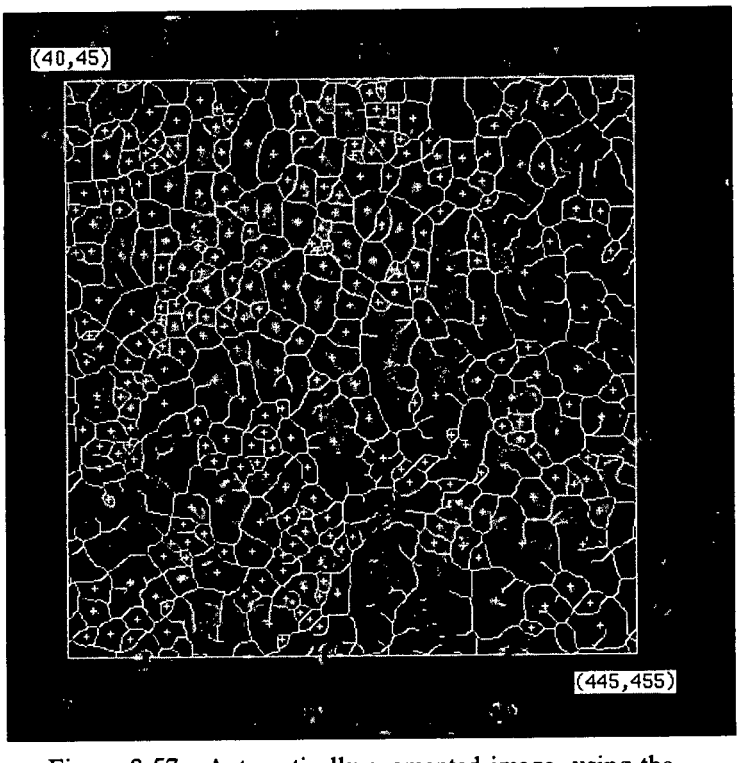


Figure 8.57 : Automatically segmented image, using the optimal DSSE of radius 11 pixels.

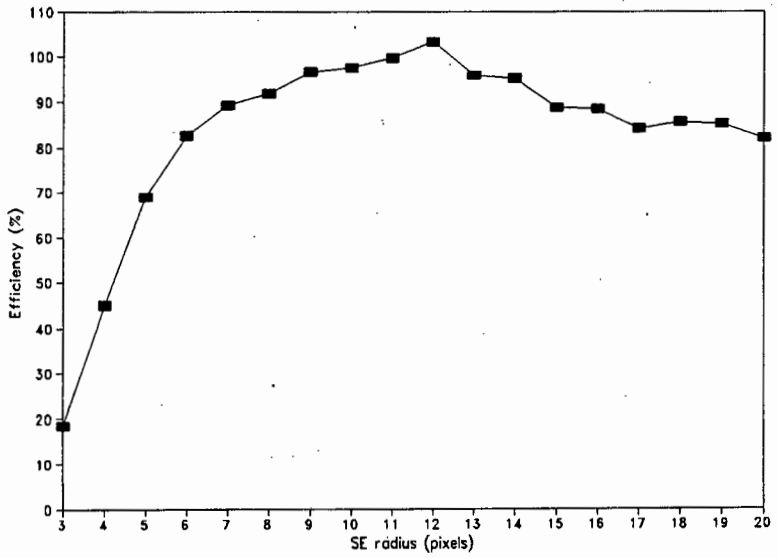


Figure 8.58 : Segmentation efficiency curve.

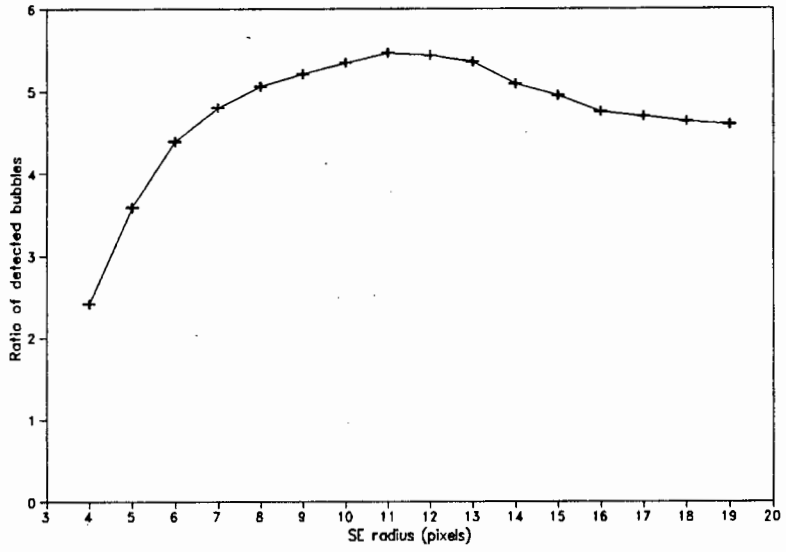
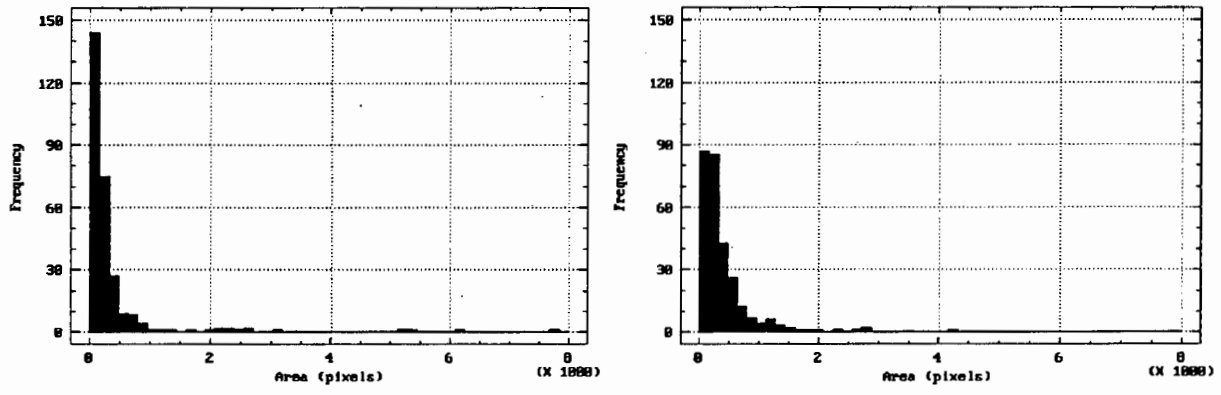
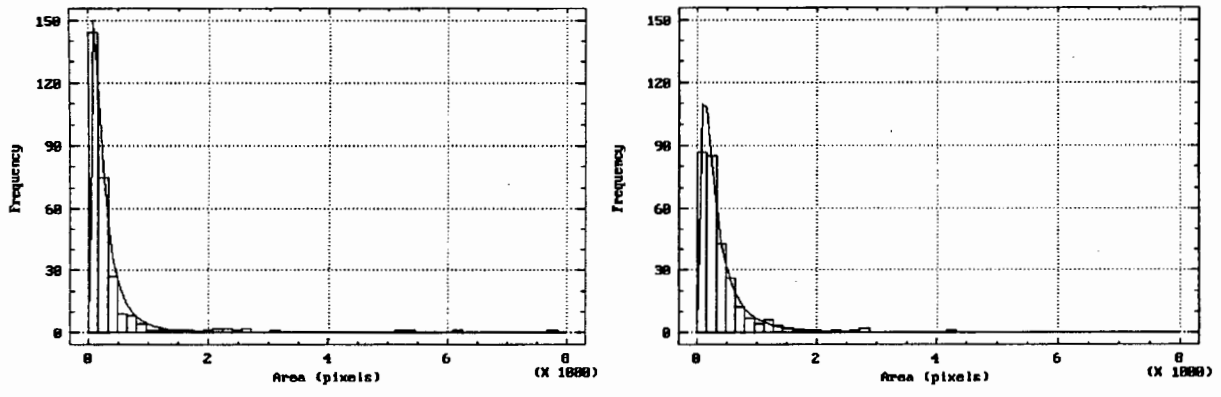


Figure 8.59 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.60 : (a) Expected size distribution with μ = 378.54, σ = 823.55, skewness = 5.81, Q_{50} = 160.00 and mode = 91.00.
(b) Observed size distribution with μ = 399.73, σ = 485.24, skewness = 3.74, Q_{50} = 255.00 and mode = 145.00.

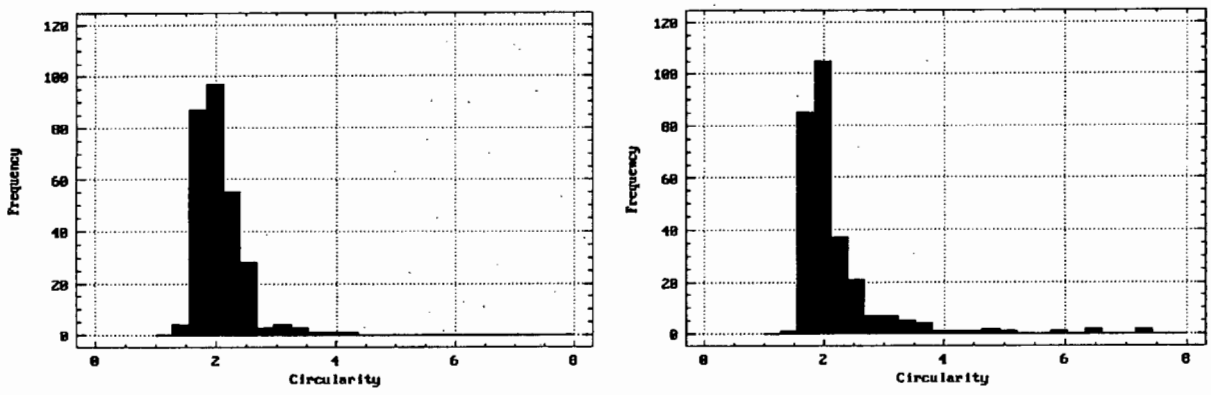


(a) (b)

Figure 8.61 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 300.34 and σ = 348.56.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 389.82 and σ = 445.23.

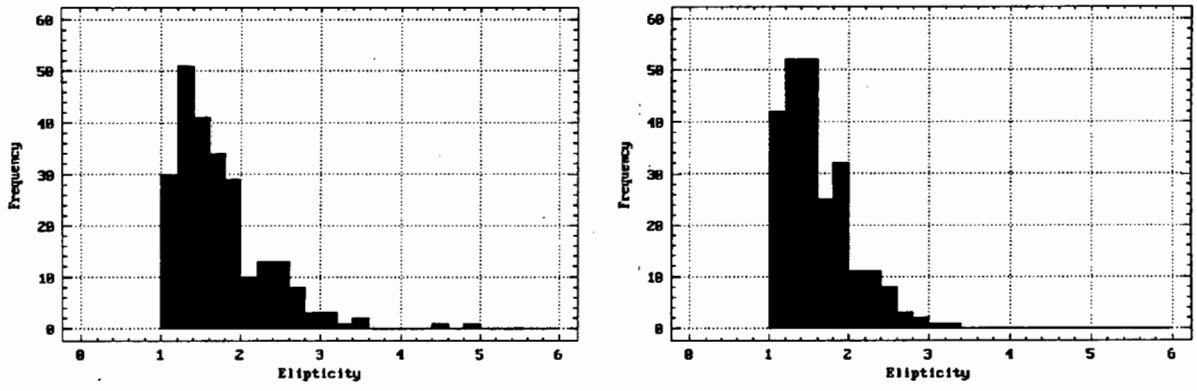
Figure	χ^2 tests	Conclusion
8.60	Calculated : χ^2_6 = 103.25 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Reject H_0
8.61 (a)	Calculated : χ^2_5 = 32.13 Tabulated : $\chi^2_{5;0.05}$ = 11.07	Reject H_0
8.61 (b)	Calculated : χ^2_6 = 2.78 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Accept H_0

Table 8.14 : χ^2 tests for Figures 8.60, 8.61 (a) and (b).



(a) (b)

Figure 8.62 : (a) Expected circularity distribution with μ = 2.05, σ = 0.39, skewness = 1.98, Q_{50} = 1.98 and mode = 1.98.
(b) Observed circularity distribution with μ = 2.22, σ = 0.81, skewness = 3.78, Q_{50} = 1.99 and mode = 1.87.



(a) (b)

Figure 8.63 : (a) Expected elipticity distribution with μ = 1.67, σ = 0.56, skewness = 2.35, Q_{50} = 1.52 and mode = 1.23.
(b) Observed elipticity distribution with μ = 1.67, σ = 0.42, skewness = 1.07, Q_{50} = 1.59 and mode = 1.92.

Figure	χ^2 tests	Conclusion
8.62	Calculated : χ^2_5 = 48.24 Tabulated : $\chi^2_{5;0.05}$ = 11.07	Reject H_0
8.63	Calculated : χ^2_8 = 11.01 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Accept H_0

Table 8.15 : χ^2 tests for Figures 8.62 and 8.63.

8.2.8 BUB9.CFI

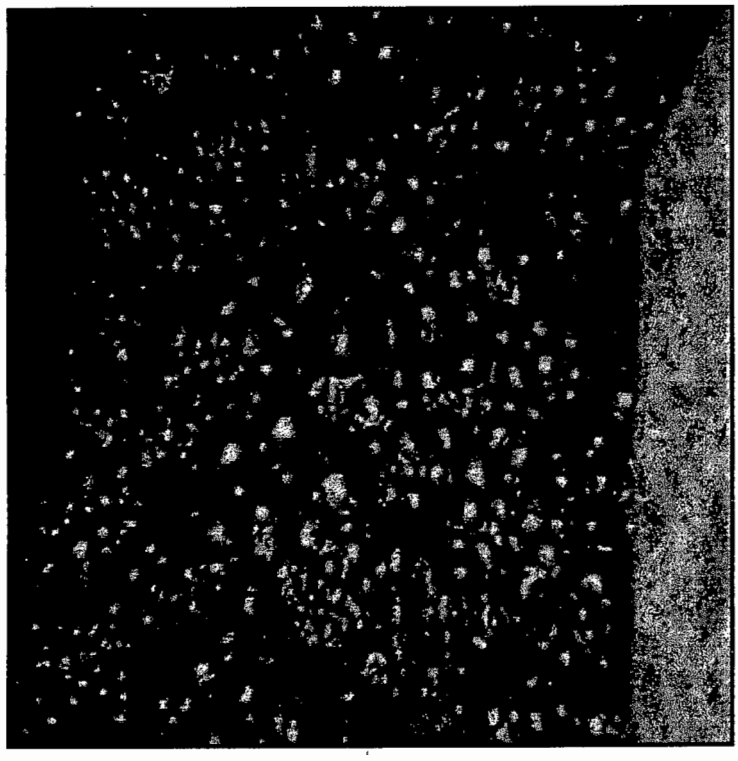


Figure 8.64 : BUB9.CFI - original image.

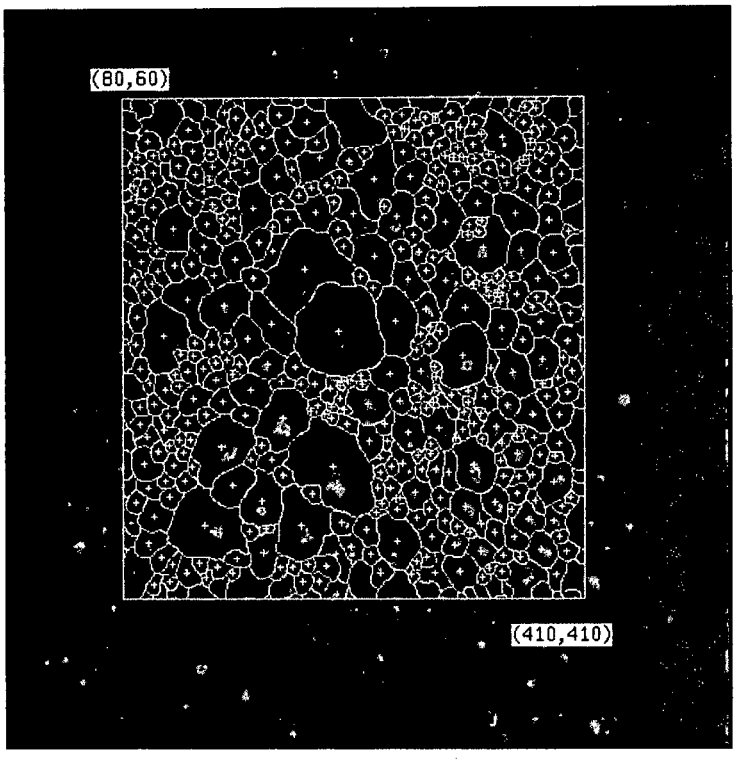


Figure 8.65 : Manually segmented image.

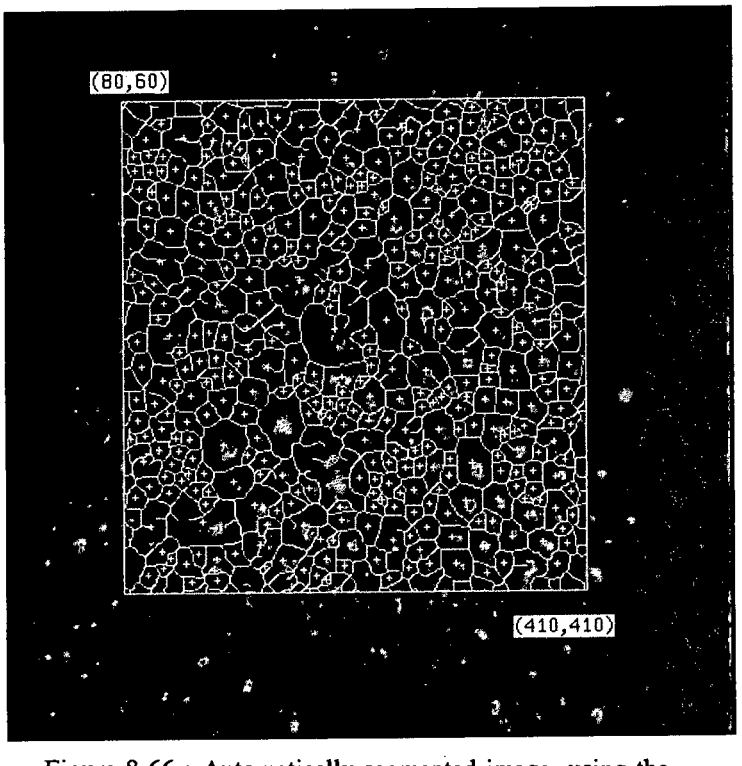


Figure 8.66 : Automatically segmented image, using the optimal DSSE of radius 8 pixels.

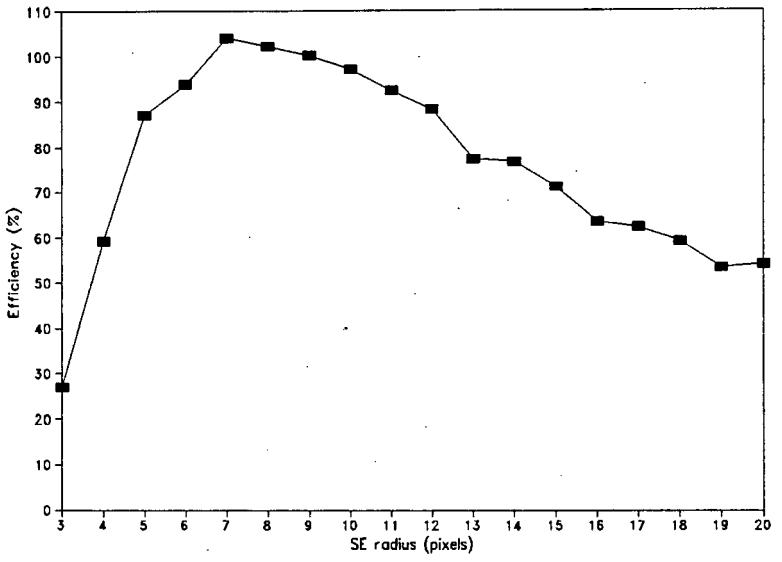


Figure 8.67 : Segmentation efficiency curve.

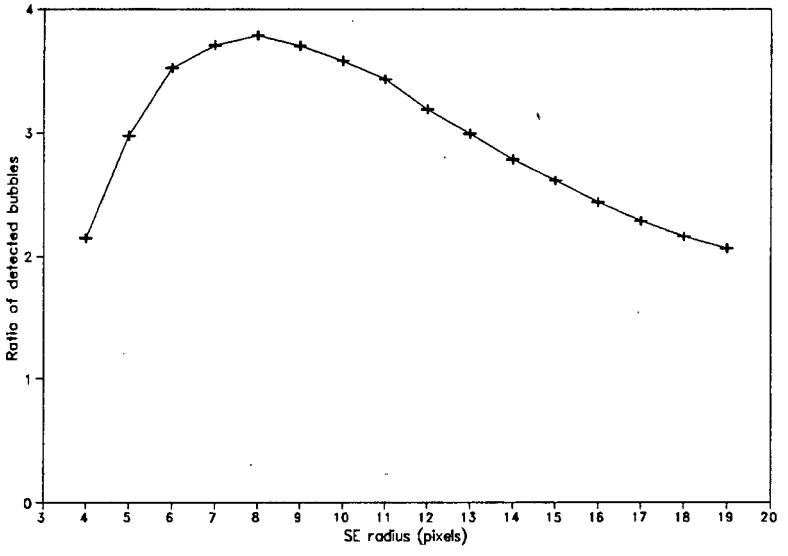
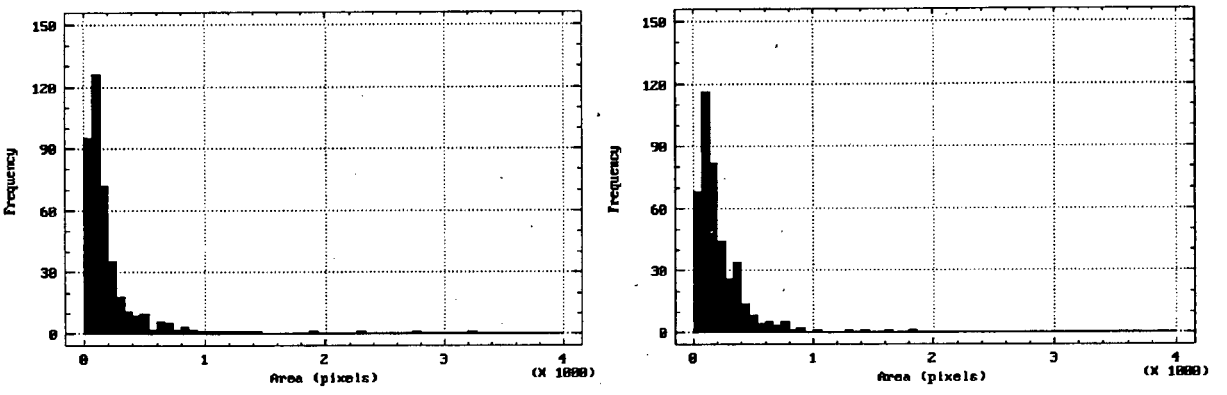
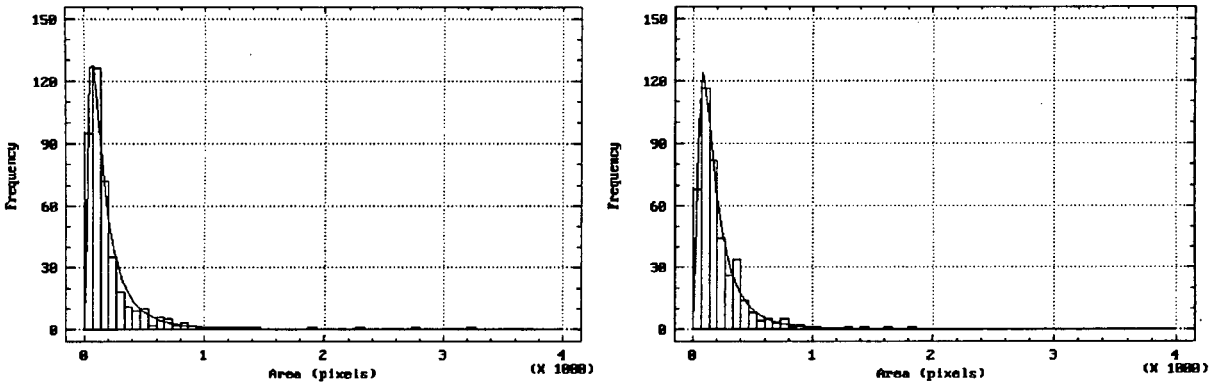


Figure 8.68 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.69 : (a) Expected size distribution with μ = 217.71, σ = 321.69, skewness = 5.03, Q_{50} = 123.50 and mode = 51.00.
(b) Observed size distribution with μ = 215.22, σ = 214.47, skewness = 3.29, Q_{50} = 150.00 and mode = 54.00.

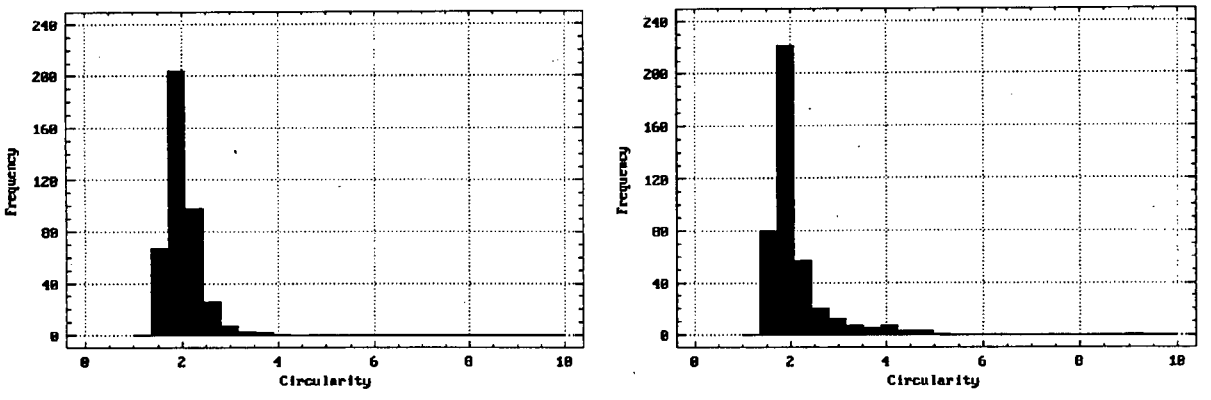


(a) (b)

Figure 8.70 : (a) Log normal fit of the expected size distribution, with the best fit parameters μ = 202.30 and σ = 233.81.
(b) Log normal fit of the observed size distribution, with the best fit parameters μ = 213.83 and σ = 209.70.

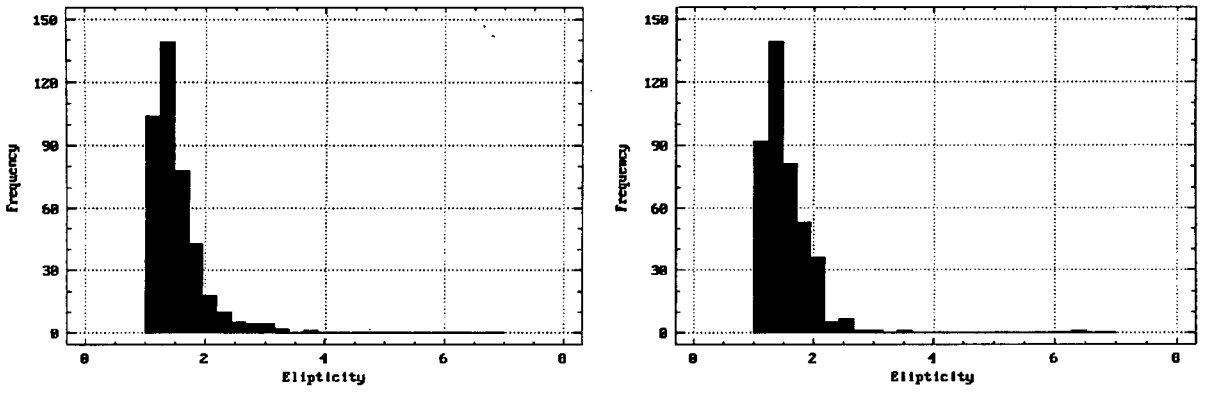
Figure	χ^2 tests	Conclusion
8.69	Calculated : χ^2_{10} = 70.42 Tabulated : $\chi^2_{10;0.05}$ = 18.31	Reject H_0
8.70 (a)	Calculated : χ^2_9 = 17.46 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Reject H_0
8.70 (b)	Calculated : χ^2_9 = 12.63 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Accept H_0

Table 8.16 : χ^2 tests for Figures 8.69, 8.70 (a) and (b).



(a) (b)

Figure 8.71 : (a) Expected circularity distribution with μ = 2.02, σ = 0.35, skewness = 1.85, Q_{50} = 1.94 and mode = 1.74.
(b) Observed circularity distribution with μ = 2.09, σ = 0.68, skewness = 4.39, Q_{50} = 1.89 and mode = 1.99.



(a) (b)

Figure 8.72 : (a) Expected elipticity distribution with μ = 1.52, σ = 0.41, skewness = 1.94, Q_{50} = 1.41 and mode = 1.33.
(b) Observed elipticity distribution with μ = 1.53, σ = 0.43, skewness = 4.32, Q_{50} = 1.44 and mode = 1.30.

Figure	χ^2 tests	Conclusion
8.71	Calculated : χ^2_5 = 97.40 Tabulated : $\chi^2_{5;0.05}$ = 11.07	Reject H_0
8.72	Calculated : χ^2_6 = 25.18 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Reject H_0

Table 8.17 : χ^2 tests for Figures 8.71 and 8.72.

8.2.9 BUB10.CFI

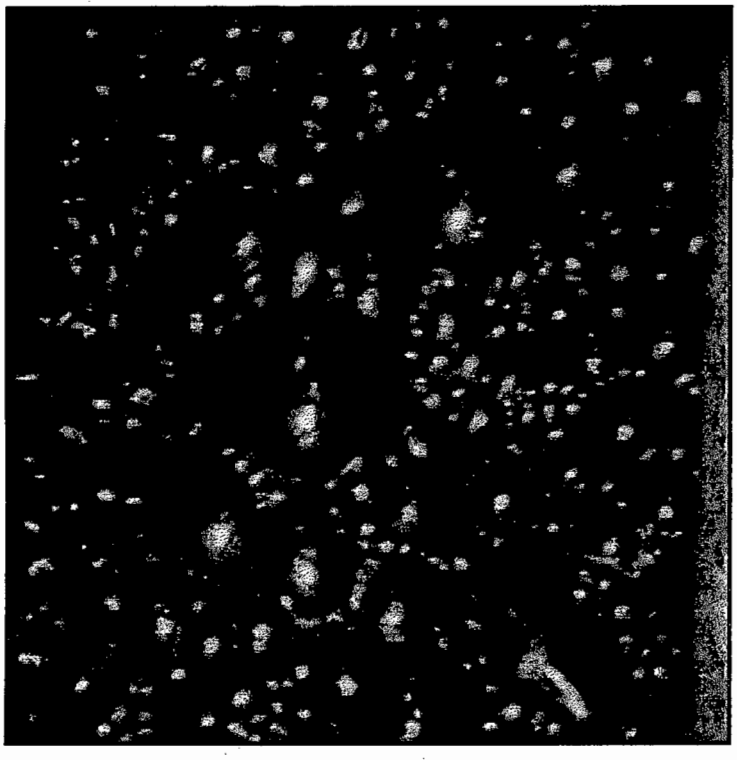


Figure 8.73 : BUB10.CFI - original image.

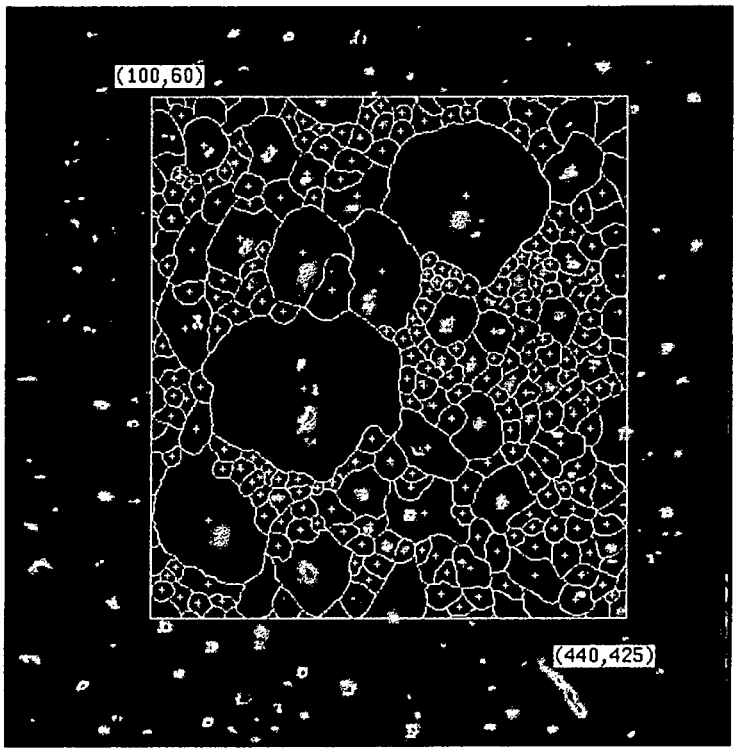


Figure 8.74 : Manually segmented image.

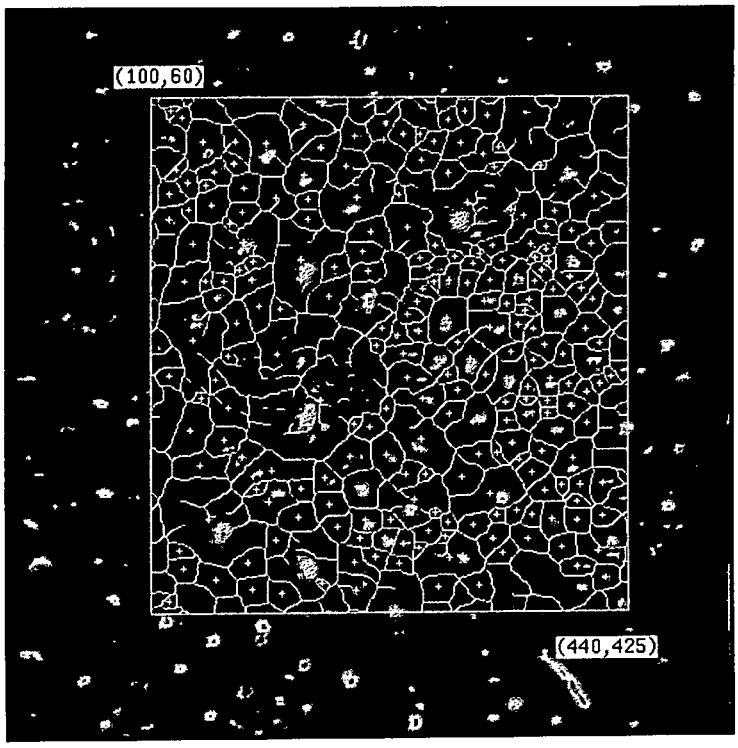
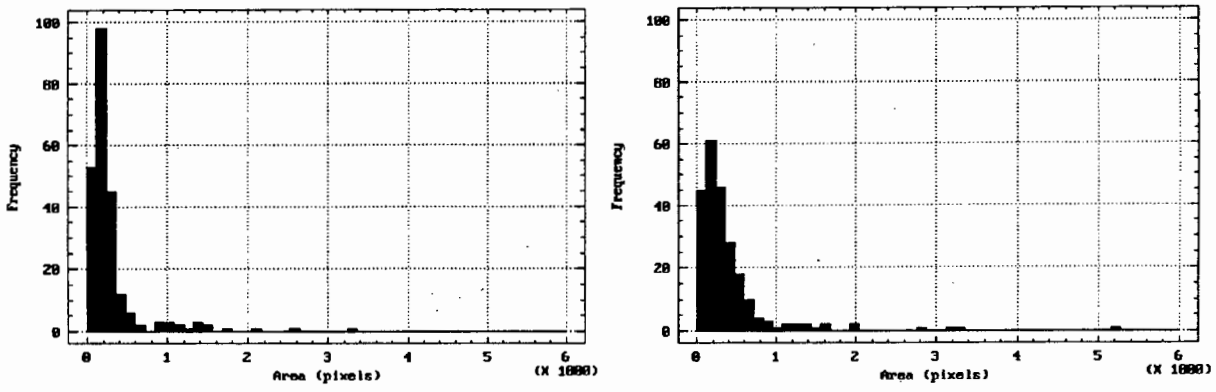
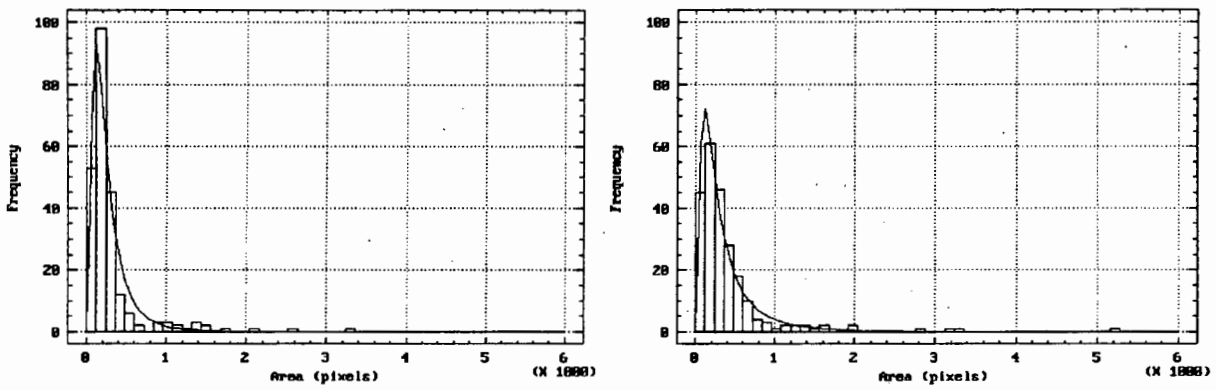


Figure 8.75 : Automatically segmented image, using the optimal DSSE of radius 11 pixels.



(a) (b)

Figure 8.78 : (a) Expected size distribution with μ = 408.85, σ = 1092.94, skewness = 8.38, Q_{50} = 197.00 and mode = 192.00.
(b) Observed size distribution with μ = 398.99, σ = 545.99, skewness = 4.87, Q_{50} = 269.00 and mode = 183.00.

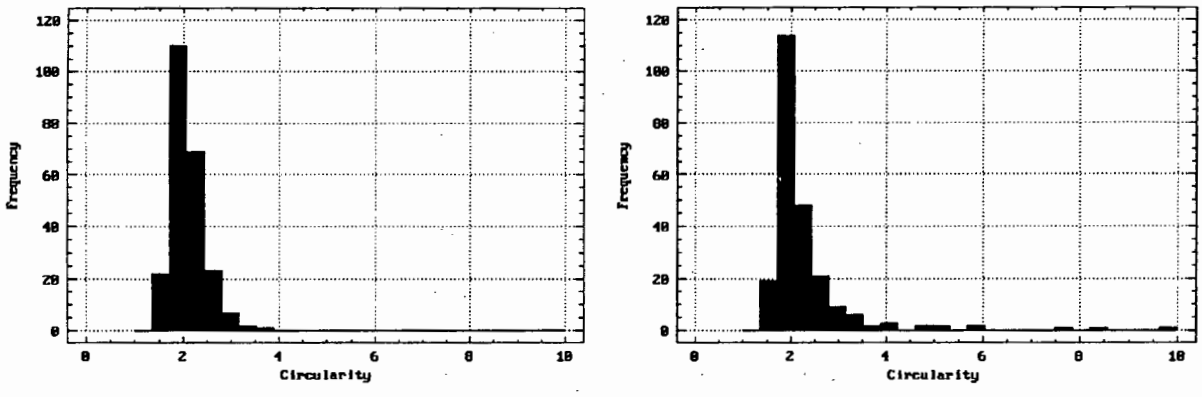


(a) (b)

Figure 8.79 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 316.42 and σ = 340.18.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 388.83 and σ = 458.81.

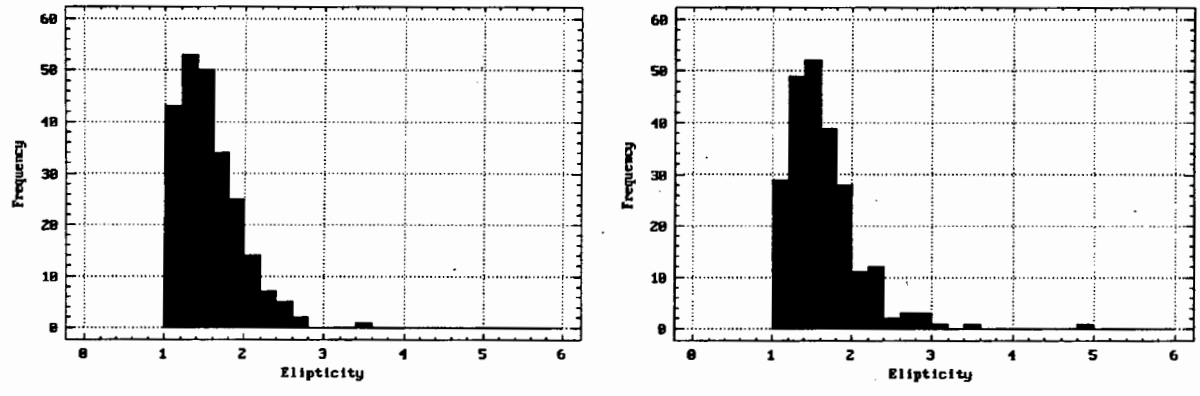
Figure	χ^2 tests	Conclusion
8.78	Calculated : χ^2_7 = 90.85 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Reject H_0
8.79 (a)	Calculated : χ^2_5 = 37.26 Tabulated : $\chi^2_{5;0.05}$ = 11.07	Reject H_0
8.79 (b)	Calculated : χ^2_8 = 7.64 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Accept H_0

Table 8.18 : χ^2 tests for Figures 8.78, 8.79 (a) and (b).



(a) (b)

Figure 8.80 : (a) Expected circularity distribution with μ = 2.10, σ = 0.34, skewness = 1.25, Q_{50} = 2.03 and mode = 2.10.
(b) Observed circularity distribution with μ = 2.31, σ = 0.99, skewness = 4.25, Q_{50} = 2.02 and mode = 1.96.



(a) (b)

Figure 8.81 : (a) Expected elipticity distribution with μ = 1.55, σ = 0.38, skewness = 1.30, Q_{50} = 1.47 and mode = 1.36.
(b) Observed elipticity distribution with μ = 1.64, σ = 0.47, skewness = 2.45, Q_{50} = 1.55 and mode = 1.31.

Figure	χ^2 tests	Conclusion
8.80	Calculated : χ^2_4 = 45.43 Tabulated : $\chi^2_{4;0.05}$ = 9.49	Reject H_0
8.81	Calculated : χ^2_7 = 12.25 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Accept H_0

Table 8.19 : χ^2 tests for Figures 8.80 and 8.81.

8.2.10 BUB11.CFI

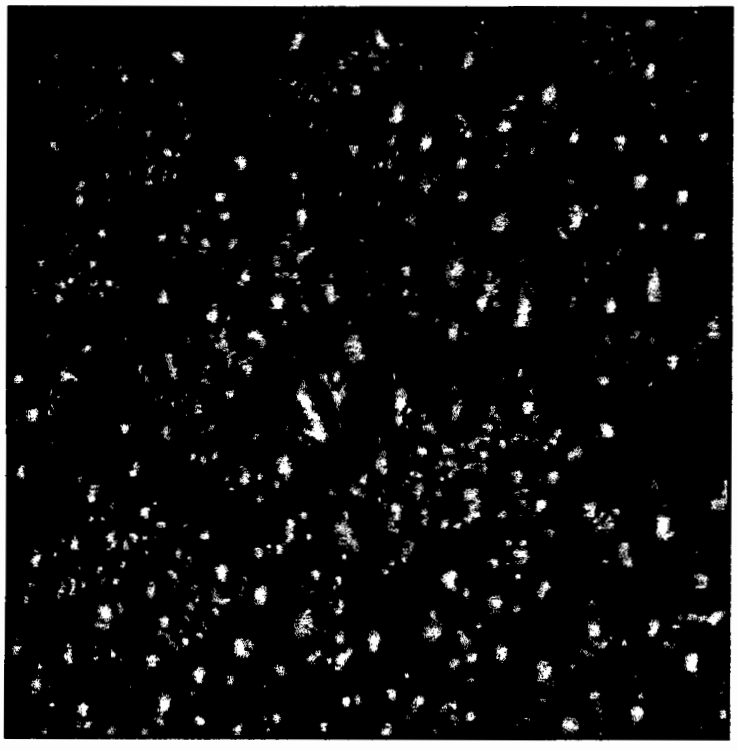


Figure 8.82 : BUB11.CFI - original image.

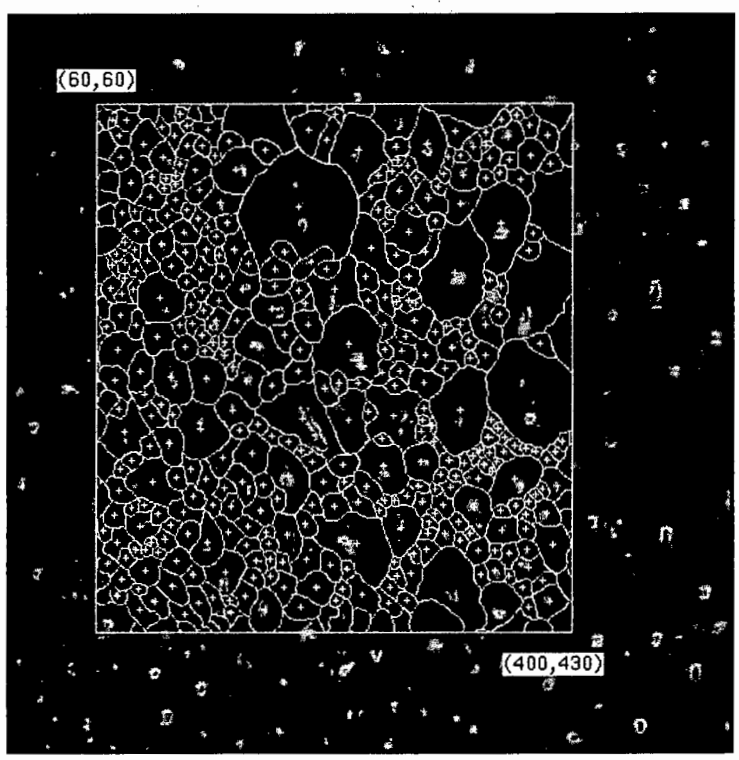


Figure 8.83 : Manually segmented image.

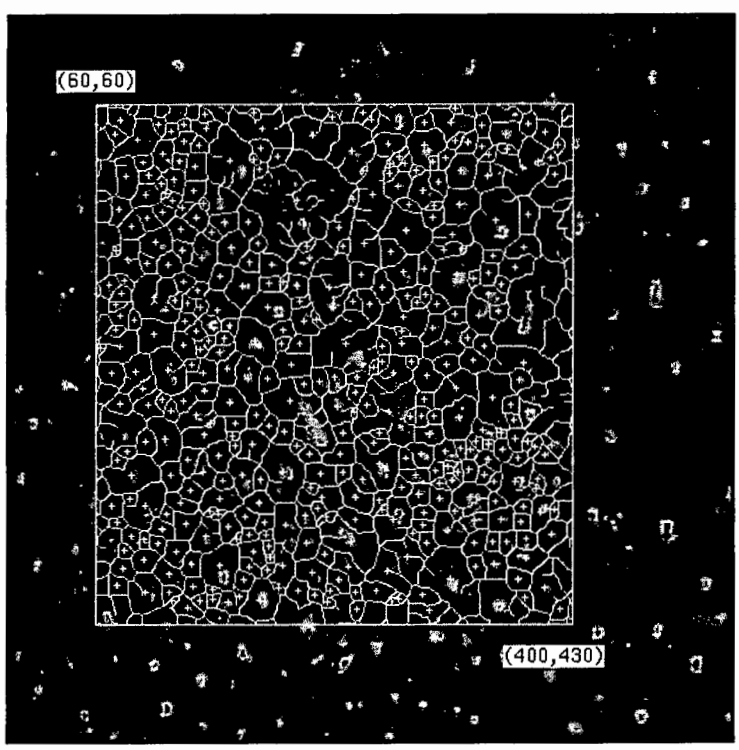


Figure 8.84 : Automatically segmented image, using the optimal DSSE of radius 8 pixels.

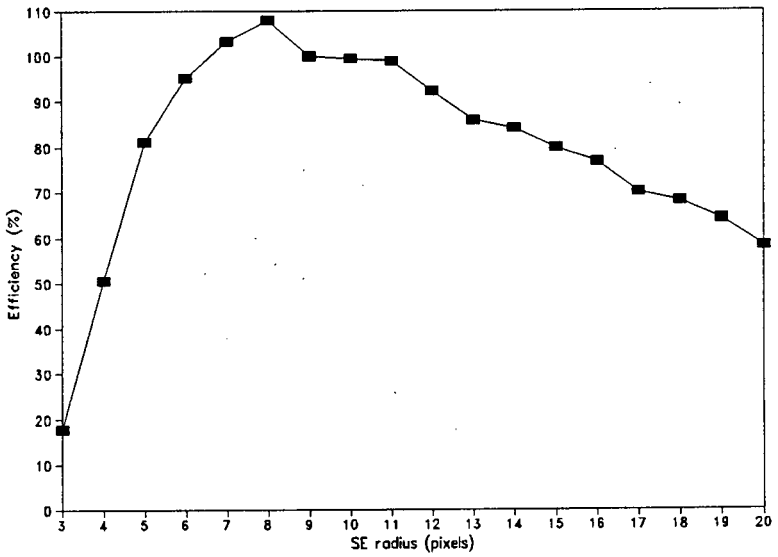


Figure 8.85 : Segmentation efficiency curve.

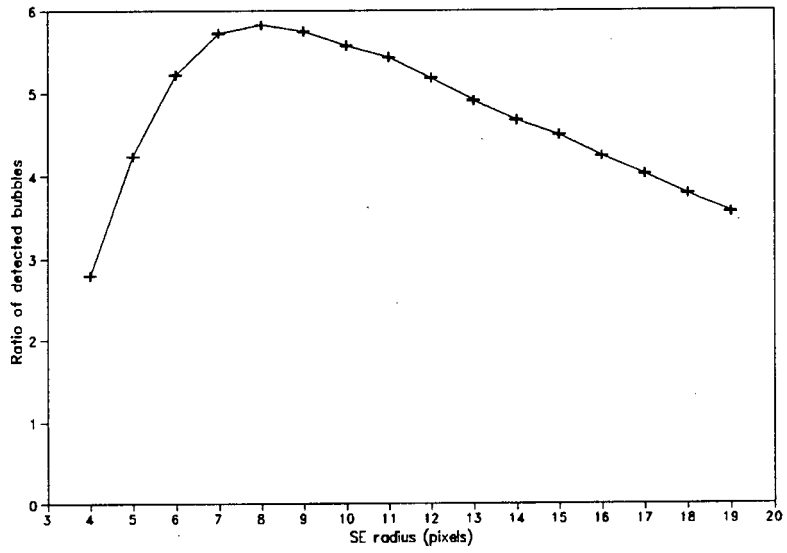
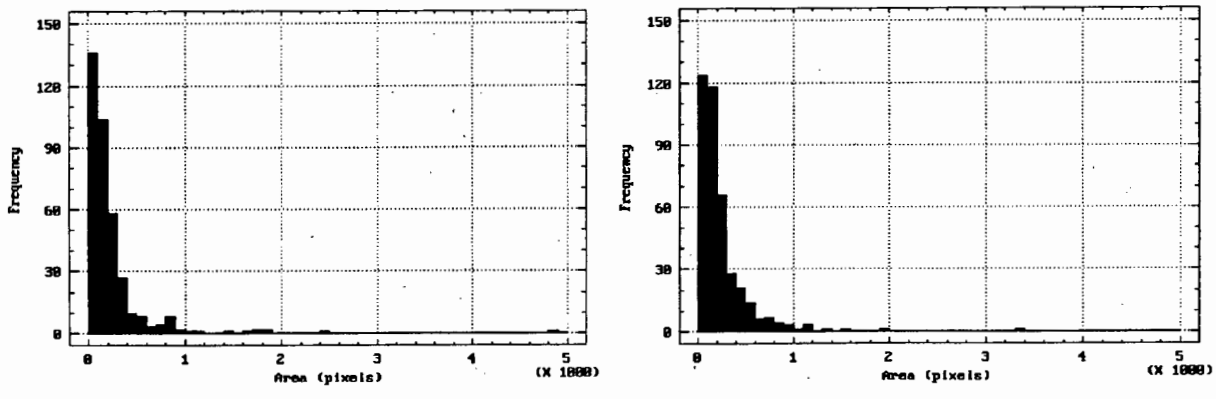
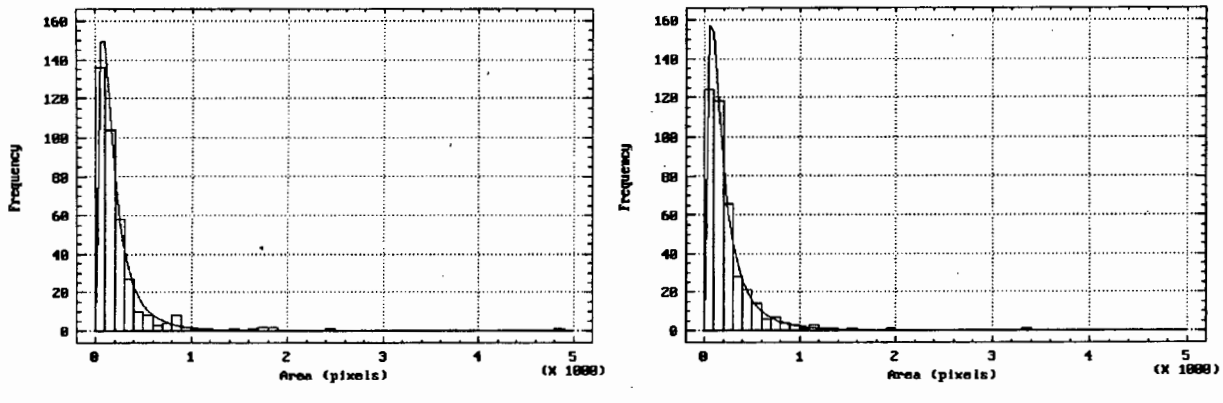


Figure 8.86 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.87 : (a) Expected size distribution with μ = 244.43, σ = 377.12, skewness = 6.56, Q_{50} = 147.00 and mode = 92.00.
(b) Observed size distribution with μ = 241.14, σ = 283.81, skewness = 4.91, Q_{50} = 164.00 and mode = 69.00.

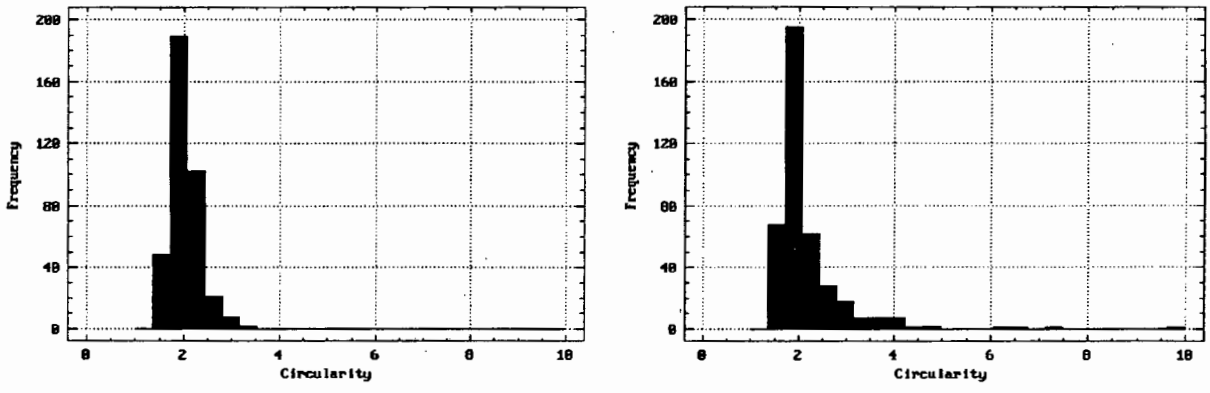


(a) (b)

Figure 8.88 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 225.04 and σ = 242.32.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 241.04 and σ = 275.69.

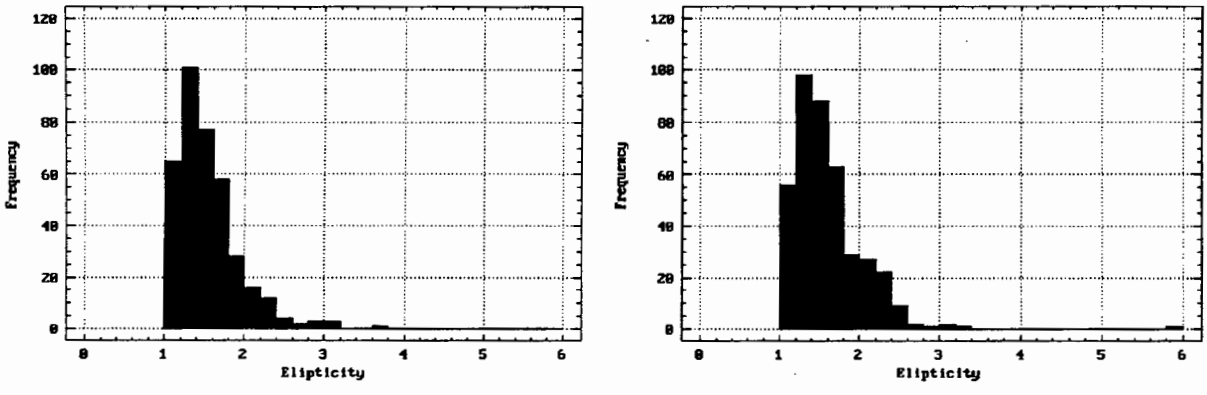
Figure	χ^2 tests	Conclusion
8.87	Calculated : χ^2_8 = 22.98 Tabulated : $\chi^2_{8;0.05}$ = 15.51	Reject H_0
8.88 (a)	Calculated : χ^2_6 = 15.26 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Reject H_0
8.88 (b)	Calculated : χ^2_7 = 2.75 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Accept H_0

Table 8.20 : χ^2 tests for Figures 8.87, 8.88 (a) and (b).



(a) (b)

Figure 8.89 : (a) Expected circularity distribution with μ = 2.03, σ = 0.31, skewness = 1.26, Q_{50} = 1.97 and mode = 1.94.
(b) Observed circularity distribution with μ = 2.16, σ = 0.78, skewness = 4.53, Q_{50} = 1.91 and mode = 1.83.



(a) (b)

Figure 8.90 : (a) Expected elipticity distribution with μ = 1.54, σ = 0.40, skewness = 1.71, Q_{50} = 1.46 and mode = 1.30.
(b) Observed elipticity distribution with μ = 1.60, σ = 0.45, skewness = 2.99, Q_{50} = 1.51 and mode = 1.29.

Figure	χ^2 tests	Conclusion
8.89	Calculated : χ^2_4 = 143.19 Tabulated : $\chi^2_{4;0.05}$ = 9.49	Reject H_0
8.90	Calculated : χ^2_7 = 17.03 Tabulated : $\chi^2_{7;0.05}$ = 14.07	Reject H_0

Table 8.21 : χ^2 tests for Figures 8.89 and 8.90.

8.2.11 BUB12.CFI

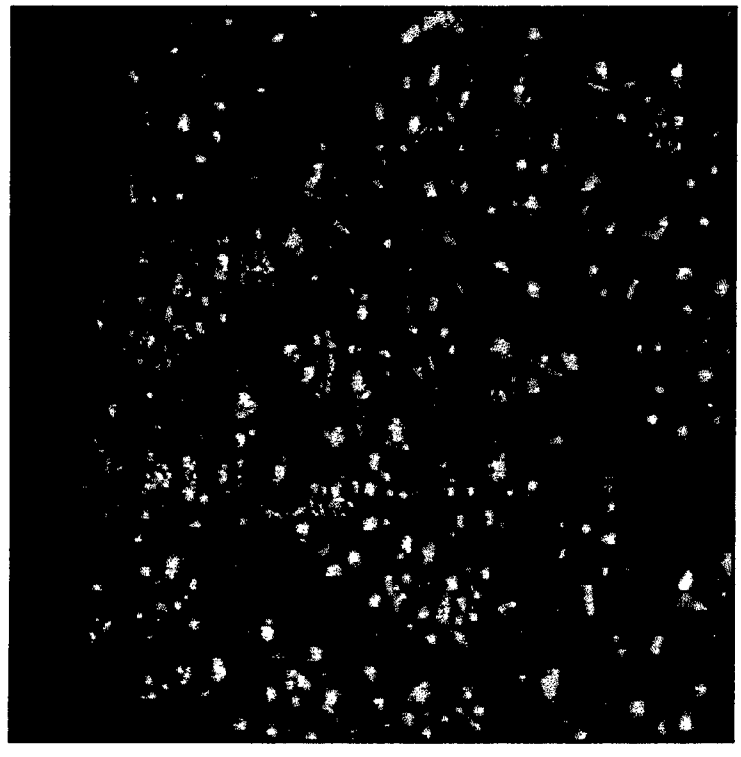


Figure 8.91 : BUB12.CFI - original image.

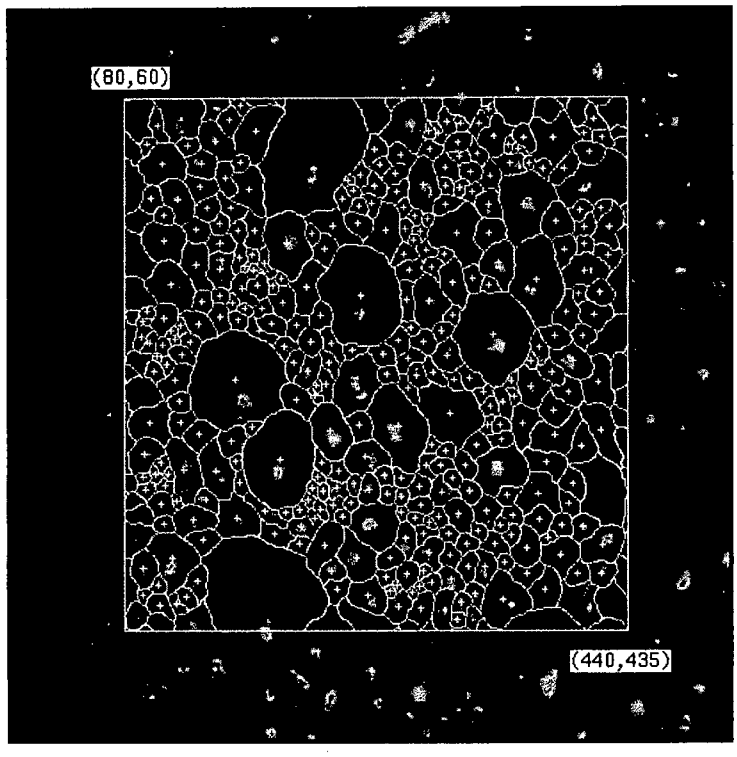


Figure 8.92 : Manually segmented image.

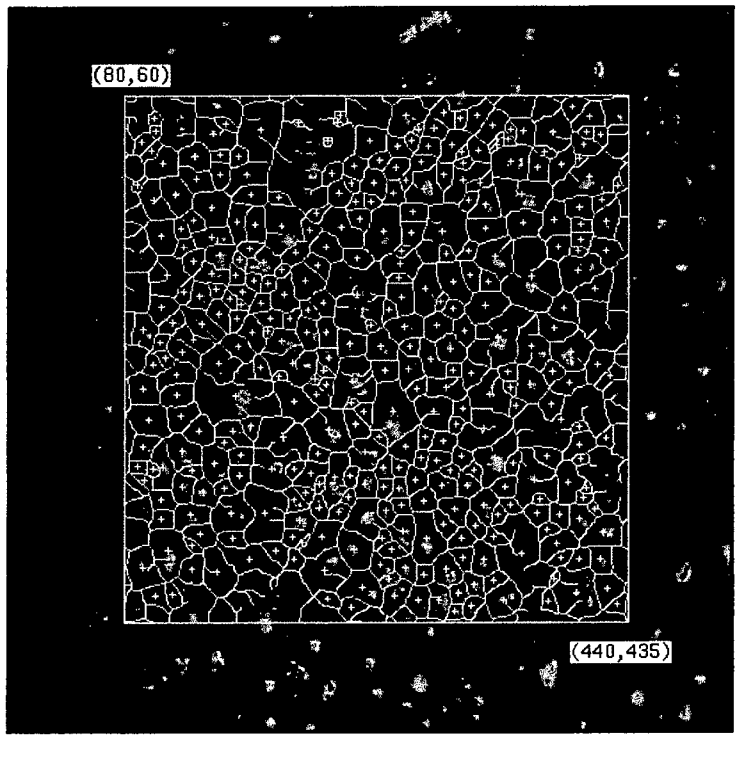


Figure 8.93 : Automatically segmented image, using the optimal DSSE of radius 10 pixels.

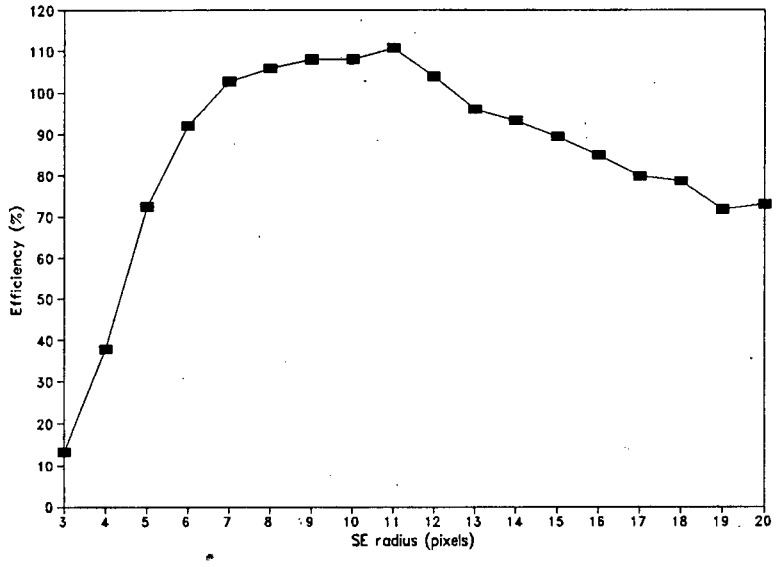


Figure 8.94 : Segmentation efficiency curve.

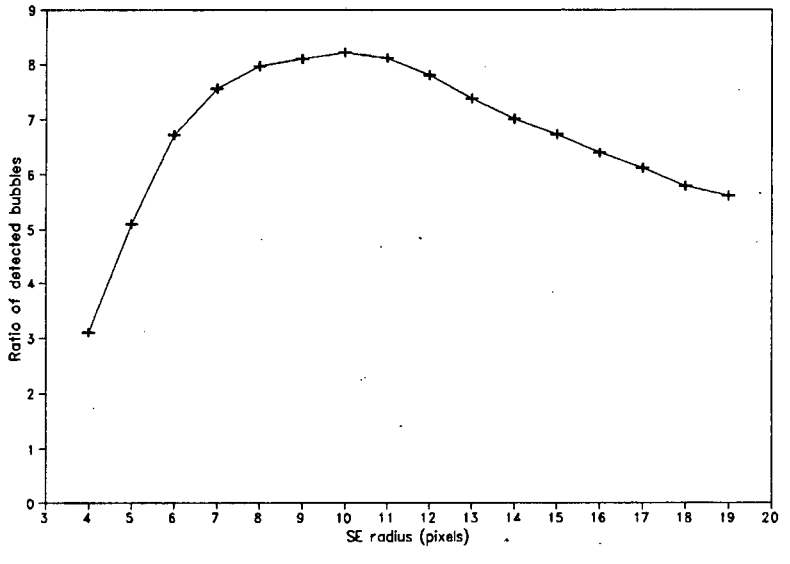
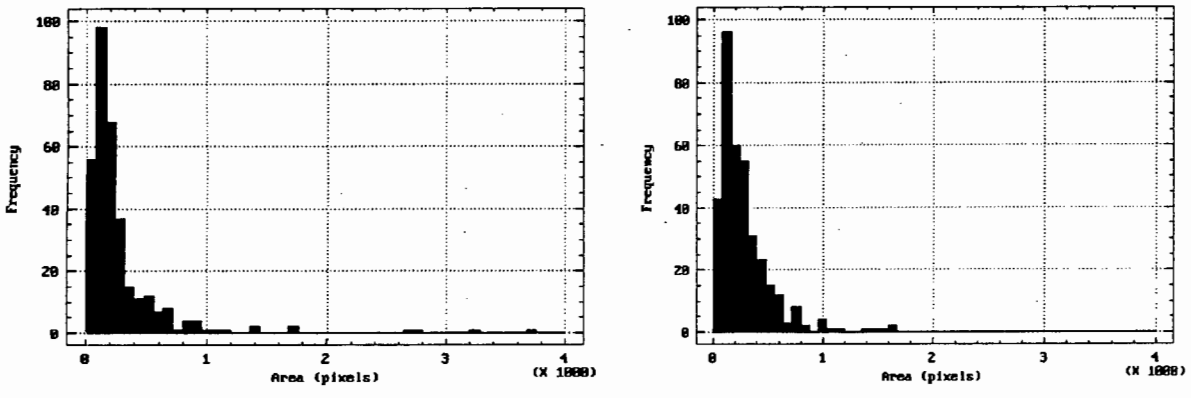
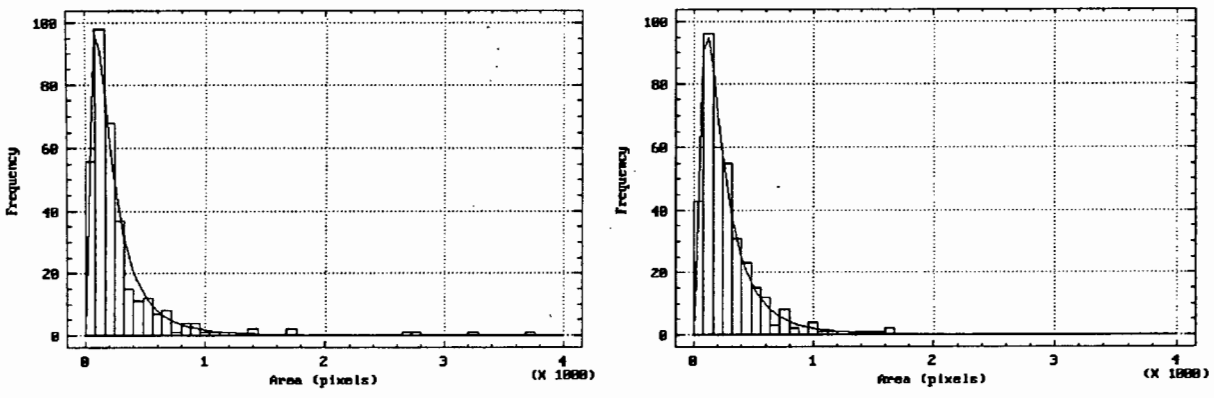


Figure 8.95 : Three point averaged optimal DSSE curve.



(a) (b)

Figure 8.96 : (a) Expected size distribution with μ = 284.44, σ = 399.21, skewness = 5.03, Q_{50} = 177.00 and mode = 144.00.
(b) Observed size distribution with μ = 280.36, σ = 250.20, skewness =. 2.54, Q_{50} = 210.00 and mode = 132.00.

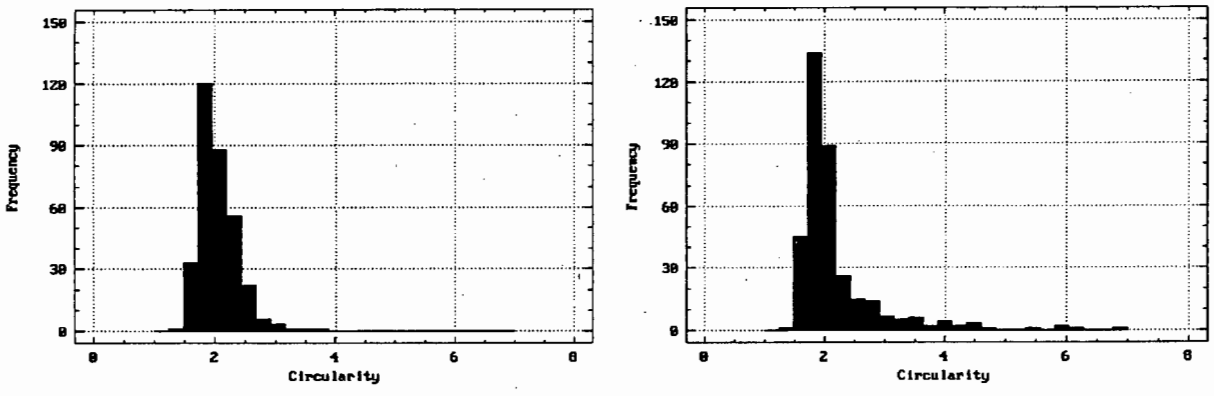


(a) (b)

Figure 8.97 : (a) Log normal fit to the expected size distribution, with the best fit parameters μ = 266.26 and σ = 279.24.
(b) Log normal fit to the observed size distribution, with the best fit parameters μ = 284.03 and σ = 274.65.

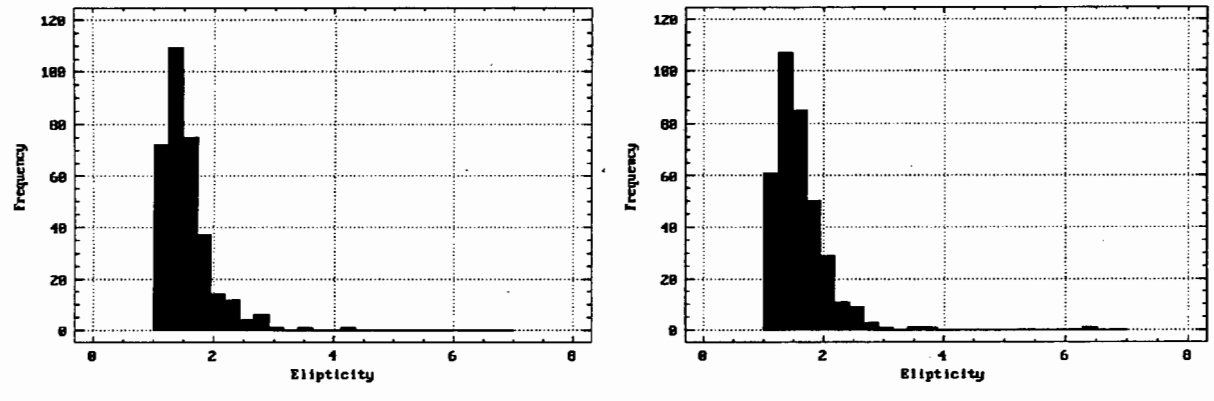
Figure	χ^2 tests	Conclusion
8.96	Calculated : χ^2_9 = 42.57 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Reject H_0
8.97 (a)	Calculated : χ^2_9 = 11.31 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Accept H_0
8.97 (b)	Calculated : χ^2_9 = 11.25 Tabulated : $\chi^2_{9;0.05}$ = 16.92	Accept H_0

Table 8.22 : χ^2 tests for Figures 8.96, 8.97 (a) and (b).



(a) (b)

Figure 8.98 : (a) Expected circularity distribution with μ = 2.06, σ = 0.32, skewness = 1.48, Q_{50} = 1.98 and mode = 1.87.
(b) Observed circularity distribution with μ = 2.19, σ = 0.73, skewness = 3.15, Q_{50} = 1.96 and mode = 1.87.



(a) (b)

Figure 8.99 : (a) Expected elipticity distribution with μ = 1.55, σ = 0.42, skewness = 2.09, Q_{50} = 1.45 and mode = 1.34.
(b) Observed elipticity distribution with μ = 1.61, σ = 0.47, skewness = 3.88, Q_{50} = 1.51 and mode = 1.25.

Figure	χ^2 tests	Conclusion
8.98	Calculated : χ^2_6 = 160.02 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Reject H_0
8.99	Calculated : χ^2_6 = 20.62 Tabulated : $\chi^2_{6;0.05}$ = 12.59	Reject H_0

Table 8.23 : χ^2 tests for Figures 8.98 and 8.99.

8.3 DISCUSSION OF THE RESULTS

The results presented in sections 8.2.1 to 8.2.11, highlight three important factors :

- The affect of the optimal DSSE size on the average size of the detected bubbles.
- The detrimental affect that multiple highlights have on the accurate segmentation of surface froth images.
- The importance of correct camera and lighting configuration.

What follows is a detailed discussion on the size and shape distributions that were obtained and how the above factors affect the results.

8.3.1 THE SIZE DISTRIBUTIONS

From a first analysis of the size distributions, the immediate conclusion one draws is the fact that the system does not work. That is, the chi-square tests for the expected and observed distributions clearly show that the distributions cannot be considered similar, as in each test the null hypothesis H_0 was rejected.

However, as stated in Chapter 7, section 7.5.1 (B), it was assumed that the manually segmented images would provide the best possible, unbiased results with which to validate the performance of the machine vision system. A problem here is that the human eye cannot distinguish between more than about thirty or forty grey levels when viewing a grey level image (Boyle and Thomas, 1988, p. 29). For images with 256 grey levels, of the type processed, there is thus a strong probability of erroneously identifying and tracing bubble boundaries manually, thereby biasing the results. This is particularly true for smaller bubbles and in regions of excessive shadow. In both these cases, the boundaries are usually ill defined and subsequent identification is the result of suitable guess work. This bias could be eliminated by having at least five people segment the same images manually. These images could then be processed individually, the results for the respective images cumulated and then averaged. The averaged results would be more accurate and less prone to individual bias. This method of eliminating individual bias may not however, be feasible, as the task of manually segmenting images is time consuming (at least four hours per image per person) and tedious. In light of this discussion, the manually determined distributions may not be truly

representative of the froth structures and this must be taken into account when interpreting the results.

Although the distributions cannot be considered to be similar in shape, it will be shown here with the aid of Table 8.24, that the automatically determined mean sizes are descriptive of the processed surface froth images.

		MANUAL RESULTS		AUTOMATIC RESULTS	
IMAGE	LABEL	N_E	MEAN SIZE	N_O	MEAN SIZE
BUB1.CFI	1	173	587.79	155	671.48
BUB2.CFI	2	188	590.21	180	632.43
BUB3.CFI	3	198	464.23	166	551.93
BUB4.CFI	4	191	513.51	180	548.16
BUB5.CFI	5	236	467.44	239	469.33
BUB6.CFI	6	279	372.13	288	391.61
BUB7.CFI	7	240	414.73	240	428.45
BUB8.CFI	8	284	378.54	283	399.73
BUB9.CFI	9	408	217.71	417	215.22
BUB10.CFI	10	237	408.85	231	398.99
BUB11.CFI	11	370	244.43	399	241.14
BUB12.CFI	12	332	284.44	359	280.36

Table 8.24 : Summarised manually and automatically determined mean bubble sizes, with the corresponding number of detected bubbles.

Figure 8.100 is a scattergram of the number of expected bubbles (N_E) against the number of observed bubbles (N_O) and Figure 8.101, is a scattergram of the expected mean bubble size against the observed mean bubble size. In both graphs the points have been labelled (as per the labels given in Table 8.24), for ease of identification. If the system were performing ideally, that is, if it were mimicking the human visual process directly, then in both graphs the points would lie on the 45° line that is illustrated. However, due to lighting problems and the affect of DSSE size on bubble identification, the points sit above and below the line. A detailed explanation of these deviations will be provided shortly.

Prior to discussing the results, it is important to realize that the twelve images that were processed, are representative of three different froth structures. For the purposes of the following discussion, it will be assumed that they come from three different flotation cells A, B and C where : BUB1.CFI - BUB4.CFI are from cell A, BUB5.CFI - BUB8.CFI are from cell B and BUB9.CFI - BUB12.CFI are from cell C.

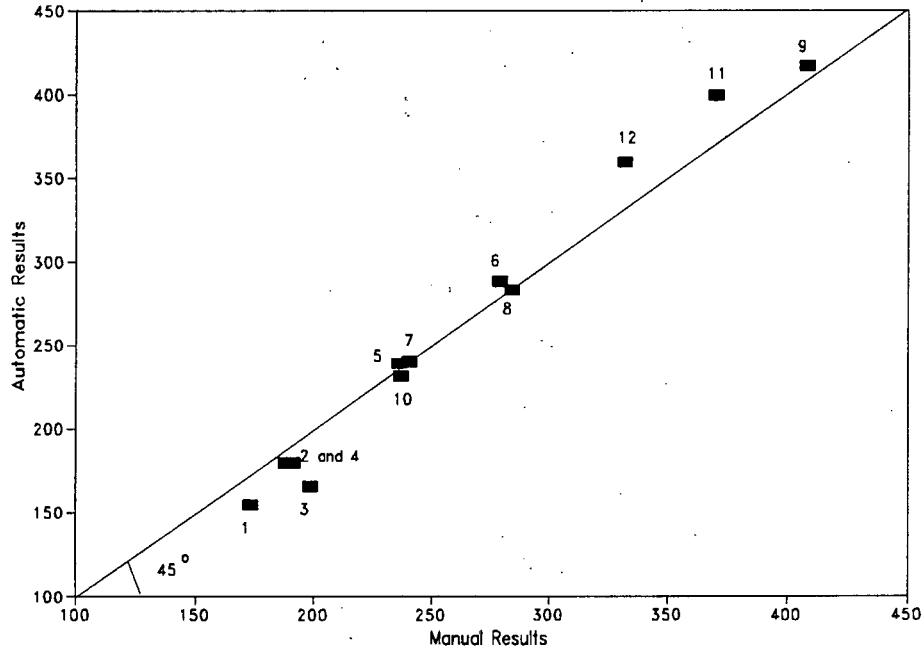


Figure 8.100 : Scattergram of the expected number of bubbles (N_E) against the observed number of bubbles (N_O).

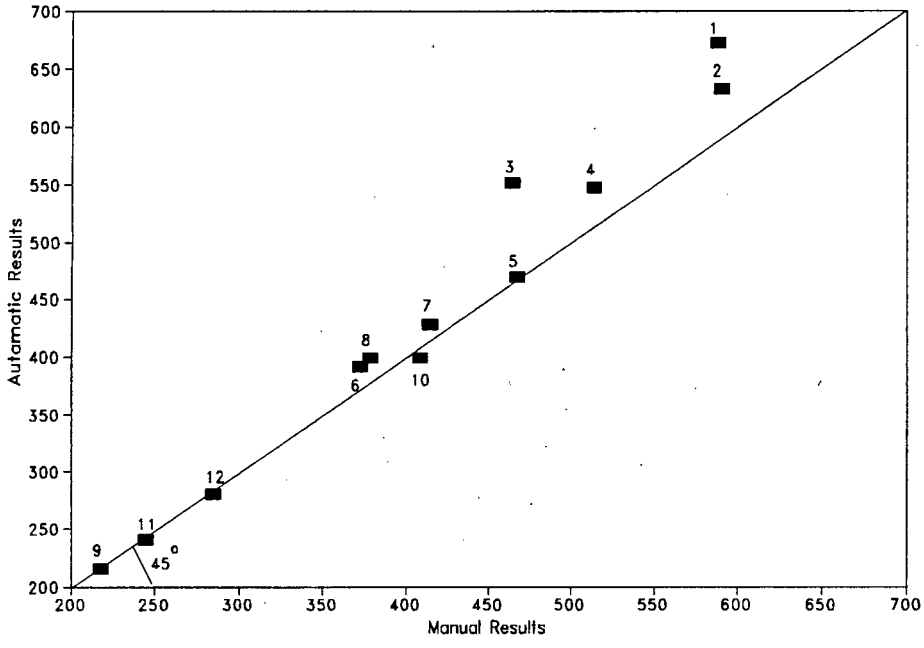


Figure 8.101 : Scattergram of the expected mean bubble size against the observed mean bubble size.

An explanation of the results obtained for BUB1.CFI - BUB4.CFI

An analysis of the manually segmented images reveals a large concentration of smaller bubbles, which indicates a definite bias. This bias can be attributed to the fact that these were the first images to be segmented manually. As can be seen from the manually segmented images for BUB5.CFI - BUB12.CFI, the segmentation improved with practice.

An analysis of the automatically segmented images, shows that the detected bubbles are on average larger than expected. In some cases, due to the size of the SEs used, the smaller bubbles have coalesced forming larger bubbles. The size of the SE used thus limits the size of the smallest bubbles that can be identified. As a result the observed bubbles are on average larger than the expected, hence the points 1, 2, 3 and 4 sit above the 45° line in Figure 8.101 and are fewer in number, hence the points 1, 2, 3 and 4 sit below the 45° line in Figure 8.100.

An explanation of the results obtained for BUB5.CFI - BUB8.CFI

A study of both the manually and automatically segmented surfaces for BUB5.CFI - BUB8.CFI, highlights the detrimental affect that multiple highlights have on the accurate segmentation of surface froth images. On larger bubble surfaces, dual highlights are clearly visible, which result in the fragmentation of the bubbles during the segmentation process. As explained in Chapter 6, section 6.6.4, this is due to the fact that the region between the highlights are erroneously identified as bubble boundaries. Again, the large SE used limits the size of the smallest bubbles that can be detected and the observed bubble size, is therefore on average larger than the expected bubble size. The counteractive affect of SE size and highlight fragmentation, seem to have had a neutralising affect and the number of detected bubbles as well as the mean bubble sizes are approximately equal. Hence the points 5, 6, 7 and 8, almost sit on the 45° lines in both Figures 8.100 and 8.101.

An explanation of the results obtained for BUB9.CFI - BUB12.CFI

The automatic segmentation results obtained for these images illustrate the importance of a correct camera and lighting configuration. These images were taken at an angle and the surfaces were poorly illuminated. Because of the obtuse angle of the camera and inadequate lighting, the bubble sides farthest away from the camera, lie in shadow.

These images are characterised by assemblies of very small bubbles and as a result a smaller SE size was used to automatically segment the images. It can be seen that the SE size has not in any way limited the size of the smallest bubble that can be detected. The problem here however, is that the few large bubbles that are present have been inaccurately segmented due to excessive shadow regions. Due to the large concentration of small bubbles, the fragmentation of the (few) larger bubbles, does not affect the mean sizes significantly, thus the points 9, 11 and 12 sit virtually on the 45° line in Figure 8.101 (10 is explained later). However, the fragmentation of the larger bubbles, means that the observed number of bubbles exceeds the expected number and as a result points 9, 11 and 12 sit above the 45° line in Figure 8.100.

BUB10.CFI is a special case as it is a close up view of the same structures in BUB9.CFI, BUB11.CFI and BUB12.CFI. This particular image illustrates the importance of a fixed lighting and camera assembly. Preferably one that sits vertically above the surface froth. Due to the close up view, the number of expected and observed bubbles are considerably lower than for the other three images, which explains why the point 10 in Figure 8.100, sits so far below 9, 11 and 12. Due to excessive shadow regions resulting from the obtuse angle of the camera, the larger bubbles have fragmented during the segmentation process. The larger SE has not been able to identify the smaller bubbles successfully and therefore, the detected bubbles are on average larger than those in the manually segmented image. Again the results illustrate a neutralising affect and the expected and observed means are approximately equal and larger than those for 9, 11 and 12. This explains the position of point 10 in Figure 8.101.

8.3.2 THE CIRCULARITY DISTRIBUTIONS

For reasons discussed in Chapter 7, section 7.4.2 (page 7-13) and from the results presented, the circularity factor is not the optimum measurement of bubble shape. A comparison of the circularity and elipticity distributions shows that the circularity measure, typically has a wider variation and is therefore less robust. In all instances, the expected and observed circularity distributions failed the χ^2 test.

8.3.3 THE ELIPTICITY DISTRIBUTIONS

As the elipticity factor is computed using the moments of a blob and therefore considers every pixel within a blob, it provides a reliable and robust shape measurement. This is substantiated by the distributions and statistical measurements presented in this chapter, where some of the distributions were accepted as similar by the chi-square test. It can be concluded therefore that this is the better of the two measurements, for quantifying bubble shape.

8.4 A CONTROL CONCEPT USING MEAN BUBBLE SIZE AND ELIPTICITY MEASUREMENTS

Although the size and shape distributions show that the developed machine vision system does not mimic the human visual process directly, the computed statistical parameters for the distributions are descriptive of the processed surface froth images. Is it possible therefore, using the mean size and elipticity measurements, to recognize different froth structures?

8.4.1 THE 3-SIGMA CONTROL CHART

The control chart is a simple, effective and widely used graphical technique for interpreting fluctuations of measurements on successive random samples in a repetitive manufacturing process (Guttman, Wilks and Hunter, 1982, pp. 270-278). Depending on the state of the process being monitored, a control chart is classified as either theoretical or empirical, where :

- Theoretical control charts are used for detecting important departures from an existing **known** state of statistical control[1]. With theoretical control charts, the mean and standard deviation of the distribution for the measurements are known.
- Empirical control charts are used to assist in **establishing** a state of statistical control over a given measurement on a sequence of manufactured objects. These control charts depend on computations made from a historical sequence of samples.

If a (manufacturing) process is in a state of statistical control with respect to some

[1] A manufacturing process is said to be in a *state of statistical control* with respect to a measured characteristic, if measurements made on a series of the manufactured objects behave like a sequence of independent and identically distributed random variables.

measurement x, it is possible that at any time the process could be thrown out of control by some cause or set of causes. By constructing a control chart for the measurement x, the objective is to detect, with a **high degree** of probability, variations in the process with respect to x. This information can then be used to assist in the control of the process. To construct a control chart for a process in a state of statistical control, n sample measurements $\{x_1, x_2, x_3, \ldots, x_{n-1}, x_n\}$ with respect to a measurement x, are taken at a particular time. If the sampling process is repeated at different times, the samples will behave as independent random variables having the same distributions, where the common distribution has the mean value μ_x and standard deviation σ_x. In such a case the probability that x will fall within a given interval can be computed, assuming that the process remains in a state of statistical control with respect to x. If x is (approximately) normally distributed, which is usually but not always the case, with mean μ_x and standard deviation σ_x, then the probability that x falls inside the limits $(\mu_x \pm 3\sigma_x)$ is 0.9973. The probability of x falling outside the limits is thus 0.0027. Using this information, a 3-sigma control chart, Figure 8.102, can be constructed for x.

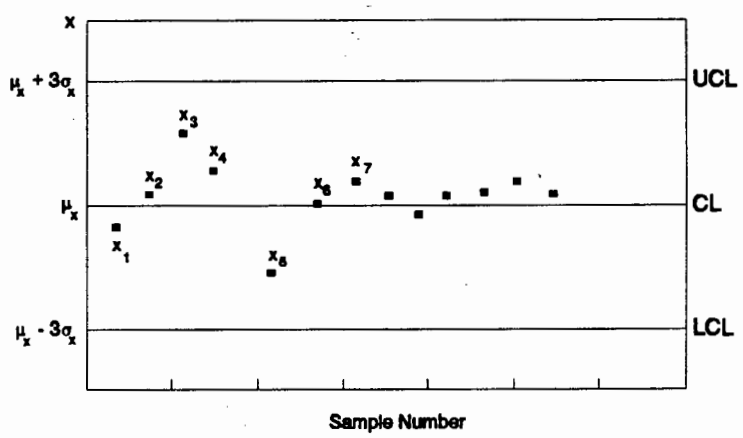


Figure 8.102 : 3-sigma control chart for the measurement x if μ_x and σ_x are known.

The control chart contains three horizontal lines :

The upper control line : $UCL = \mu_x + 3\sigma_x$

The centre control line : $CL = \mu_x$

The lower control line : $LCL = \mu_x - 3\sigma_x$

Points falling outside the upper and lower control limits, show that a disturbance has occurred in the process thereby causing a shift in the distribution.

For a process that follows a normal distribution and where the mean (μ) and standard deviation (σ) of the distribution are known, the 3-sigma control limits are given by (Guttman, Wilks and Hunter, 1982, p. 273) :

$$\textit{Sample average } \bar{x}: \quad \mu \pm 3\frac{\sigma}{\sqrt{n}} \qquad \ldots(8.1)$$

where n is the sample size.

8.4.2 A 3-SIGMA CONTROL SCATTERGRAM

The concept of single variable 3-sigma control charts can be extended to 3-sigma control scattergrams, for two variables. This extension of the control chart can be applied to characterised surface froth images, where the measured mean bubble size and elipticity are used to ascertain whether a flotation cell is in a state of statistical control. It should be possible using a 3-sigma control scattergram, to detect variations in froth structure, thereby providing a fundamental control concept that uses these measured parameters.

The following discussion assumes that :

- For the surface froth images processed, the physical and chemical conditions that affect flotation performance, were constant. That is the pH level, aeration rate, impeller rotation speed, reagent feed and pulp feed were all constant. It can be assumed therefore, that the froth structures for cell A, B and C, are representative of processes in a state of statistical control.
- The chosen AOI for each image is large enough so as not to bias the results. That is, the mean size and elipticity measurements made for BUB1.CFI - BUB4.CFI are samples from the same distribution and similarly for the measurements for BUB5.CFI - BUB8.CFI and BUB9.CFI - BUB12.CFI.

Consider that at least fifty consecutive images were processed from cell A and that the same AOI was used for all images. If the mean bubble size (μ_s) and mean elipticity (μ_e) were computed for each image, this would give rise to the set of mean bubble size measurements $\{\mu_{1s}, \mu_{2s}, \mu_{3s}, \ldots, \mu_{50s}\}$ and mean elipticity measurements $\{\mu_{1e}, \mu_{2e}, \mu_{3e}, \ldots, \mu_{50e}\}$. The Central Limit Theorem states that (Miller, 1984, p.72) :

> *"A sampling distribution tends towards normality as the sample size increases, irrespective of the shape of the population being sampled."*

It would be reasonable on the basis of the Central Limit Theorem, to assume that the mean size and elipticity values are normally distributed with a mean size value $\bar{\mu}_s$ and standard deviation of $\bar{\sigma}_s$ and mean elipticity value $\bar{\mu}_e$ and standard deviation $\bar{\sigma}_e$. These four parameters can then be used to construct a 3-sigma control scattergram, Figure 8.103, where the respective upper and lower control lines define a region of control, as illustrated.

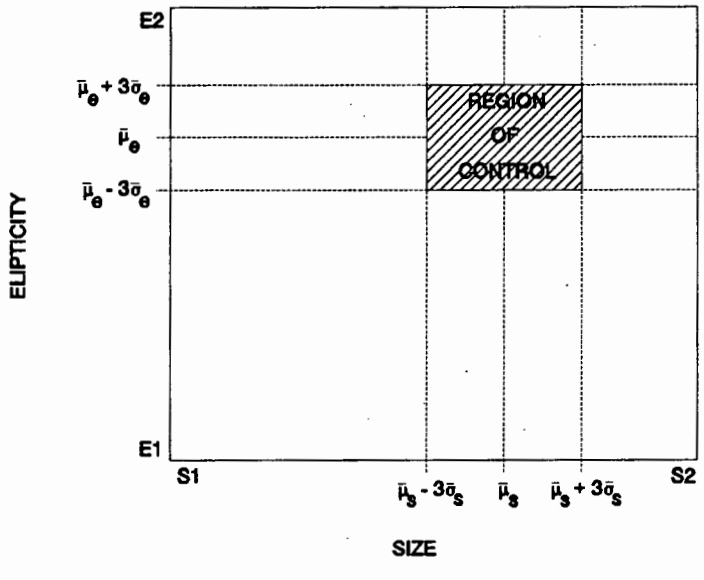


Figure 8.103 : A 3-sigma control scattergram.

That is, there is a 99.73% probability that while the process stays in a state of statistical control, the mean size and elipticity measurements will fall within the region of control. As soon as there is a disturbance to the process, that causes the bubbles to change in size and shape, the measured parameters will fall outside of the region of control. This would indicate

a change in froth structure and the information can be acted upon accordingly.

The principle of this control concept was applied to the manually and automatically determined mean size and elipticity measurements that were obtained. The results used are given in Tables 8.25 and 8.26 and the corresponding 3-sigma control scattergrams are illustrated in Figures 8.104 and 8.105, respectively. In both cases, the 3-sigma control scattergrams were constructed with respect to cell A, that is BUB1.CFI - BUB4.CFI. The upper and lower control lines were calculated according to equation (8.1), with the number of samples (n) equal to 4. In both scattergrams, the points have been labelled for ease of identification.

Both Figures 8.104 and 8.105, illustrate that there are three distinct clusters, corresponding to the three different froth structures that were processed. Furthermore, the scattergrams illustrate that the technique does recognize the different froth structures, as points 1, 2, 3 and 4, fall within the defined region of control. The remaining points all lie outside, indicating that the froth structures are different.

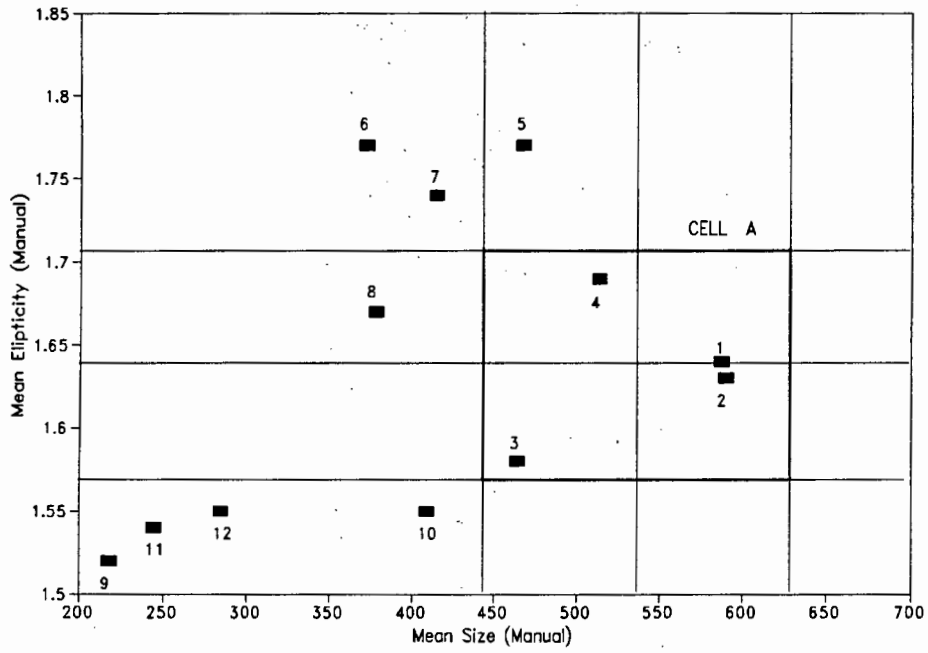


Figure 8.104 : 3-sigma control scattergram using μ_s and μ_e measurements for the manually segmented images.

			MANUAL SIZE		MANUAL ELIPTICITY	
CELL	IMAGE	LABEL	μ_s	$\bar{\mu}_s$ and $\bar{\sigma}_s$	μ_e	$\bar{\mu}_e$ and $\bar{\sigma}_e$
Cell A	BUB1.CFI	1	587.79		1.64	
	BUB2.CFI	2	590.21	538.94	1.63	1.64
	BUB3.CFI	3	464.23	61.22	1.58	0.05
	BUB4.CFI	4	513.51		1.69	
Cell B	BUB5.CFI	5	467.44		1.77	
	BUB6.CFI	6	372.13	408.21	1.77	1.74
	BUB7.CFI	7	414.73	43.71	1.74	0.05
	BUB8.CFI	8	378.54		1.67	
Cell C	BUB9.CFI	9	217.71		1.52	
	BUB10.CFI	10	408.85	288.86	1.55	1.54
	BUB11.CFI	11	244.43	84.56	1.54	0.01
	BUB12.CFI	12	284.44		1.55	
			UCL = 630.77 CL = 538.94 LCL = 447.11		UCL = 1.72 CL = 1.64 LCL = 1.57	

Table 8.25 : Manually determined μ_s and μ_e measurements, used to construct Figure 8.104.

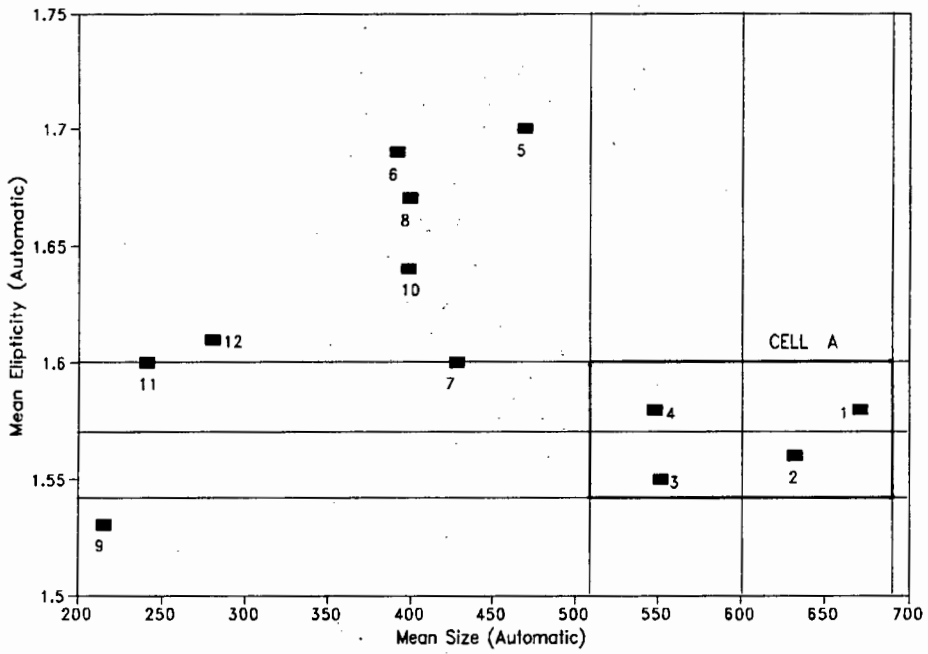


Figure 8.105 : 3-sigma control scattergram using μ_s and μ_e measurements for the automatically segmented images.

			AUTOMATIC SIZE		AUTOMATIC ELIPTICITY	
CELL	IMAGE	LABEL	μ_s	$\bar{\mu}_s$ and $\bar{\sigma}_s$	μ_e	$\bar{\mu}_e$ and $\bar{\sigma}_e$
Cell A	BUB1.CFI	1	671.48		1.58	
	BUB2.CFI	2	632.43	601.00	1.56	1.57
	BUB3.CFI	3	551.93	60.98	1.55	0.02
	BUB4.CFI	4	548.16		1.58	
Cell B	BUB5.CFI	5	469.33		1.70	
	BUB6.CFI	6	391.61	422.28	1.69	1.67
	BUB7.CFI	7	428.45	35.12	1.60	0.05
	BUB8.CFI	8	399.73		1.67	
Cell C	BUB9.CFI	9	215.22		1.53	
	BUB10.CFI	10	398.99	283.93	1.64	1.60
	BUB11.CFI	11	241.14	81.25	1.60	0.05
	BUB12.CFI	12	280.36		1.61	
			UCL = 692.47 CL = 601.00 LCL = 509.53		UCL = 1.60 CL = 1.57 LCL = 1.54	

Table 8.26 : Automatically determined μ_s and μ_e measurements, used to construct Figure 8.105.

CHAPTER 9

CONCLUSION AND RECOMMENDATIONS

The objective of this dissertation was to develop and design a machine vision system for measuring the size and shape of the bubbles that constitute the surface froth structures in industrial flotation cells. The complexity of the surfaces being processed revealed that the successful development of the required vision-based system, hinged on the ability to accurately and reliably segment surface froth images.

9.1 CONCLUSION

The segmentation of surface froth images proved to be a difficult task. Results obtained from the application of classical segmentation techniques to the segmentation of a test image, illustrated that :

- there was a need for a more specific approach to the segmentation problem.
- the bubble boundaries are the most reliable feature with which to segment surface froth images.

The realization that the valleys representing the bubble boundaries could be extracted using a suitably shaped probe moving along the image surface, motivated research into the application of morphological image processing to the segmentation problem. This work resulted in the development of an accurate and reliable segmentation technique for extracting the bubble boundaries in surface froth images. The segmentation technique is not however,

error free and results have shown that the accuracy of the technique, is to a large extent dependent upon the camera and lighting configuration used to view the scene. Of particular importance are :

- Excessive shadow regions, which are a result of improper surface illumination and camera placement. In these regions the boundaries are not clearly defined and this leads to incomplete and inaccurate bubble segmentation.
- Multiple highlights on bubble surfaces, which are the result of using more than one lighting source to illuminate the surface froth. For large bubbles in particular, segmented images show that the multiple highlights result in the fragmentation of the bubbles into a number of smaller bubbles, leading to inaccurate segmentation.
- Obtuse camera and lighting placement, which results in the formation of shadow regions on the side of the bubbles farthest away from the camera. As a result, the bubble boundaries are ill defined, causing incomplete boundary extraction or the fragmentation of the larger bubbles.

The segmented images consist of an interconnected network of lines (the extracted bubble boundaries), which give rise to distinct blobs. Results show that these blobs are representative of the individual bubbles and characterisation of segmented images was achieved using connectivity analysis to identify and process the individual blobs, to obtain bubble size and shape measurements.

By manually and automatically segmenting surface froth images, a set of expected and observed size, circularity and elipticity distributions were obtained and various descriptive statistics were computed for each distribution. A statistical comparison of the similarity of the expected and observed distributions, revealed however, that the machine vision system does not mimic the human visual process exactly. Yet, from analysis of the results it was possible to conclude that the computed mean bubble sizes are descriptive of the surfaces that were processed and that the elipticity measure is a more robust and reliable measurement of bubble shape, than bubble circularity. Using the automatically determined mean size and mean elipticity measurements, a control concept was developed, which illustrates the ability of the system, to recognize different froth structures using the two measured control parameters.

In conclusion, the machine vision system presented in this dissertation is capable of measuring mean bubble size and mean bubble elipticity. By plotting a 3-sigma control scattergram, for these two parameters, the system can recognize variations in froth structure. This information can therefore be used to optimize and assist in the automated control of flotation cell ore concentrators. A possible machine vision system that implements the work presented here, is illustrated in Figure 9.1.

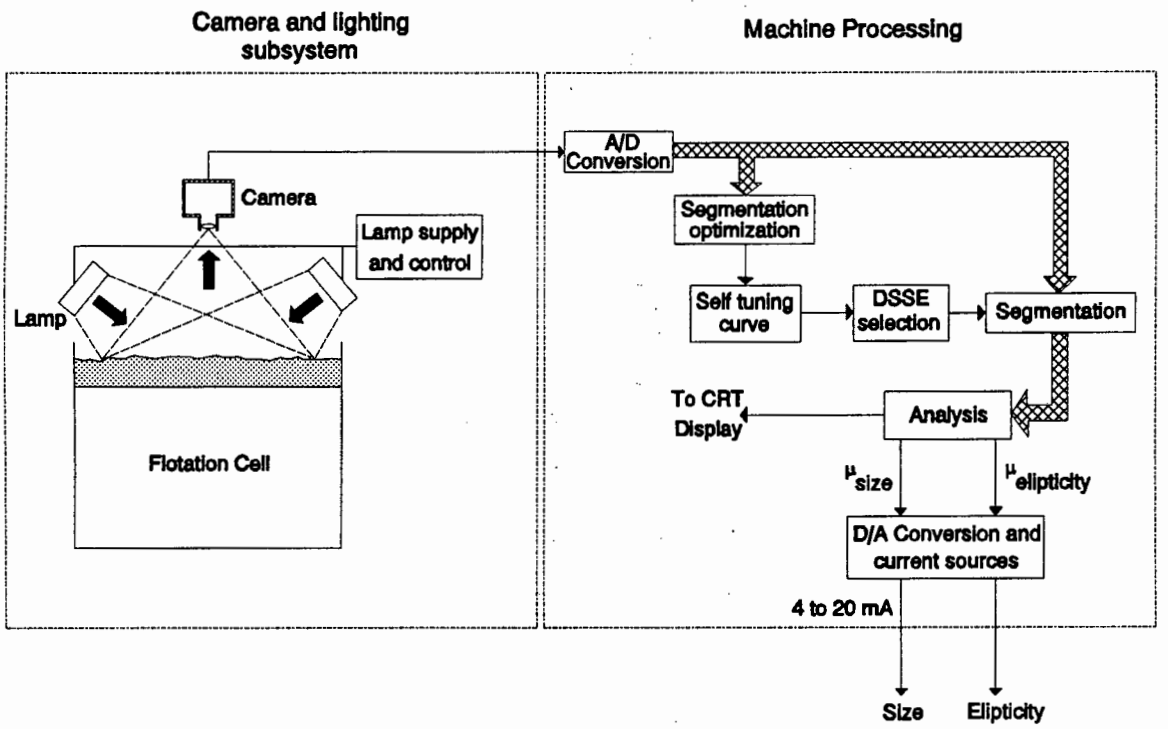


Figure 9.1 : A block diagram illustrating the possible application of the work presented in this dissertation, to a machine vision system that is capable of characterising the froth structures that are prevalent on the surface of industrial flotation cells.

9.2 RECOMMENDATIONS FOR FUTURE RESEARCH

Recommendations for future research work fall into two categories : system optimization and system validation.

9.2.1 SYSTEM OPTIMIZATION

System optimization can be achieved by :

- improving the camera and lighting configuration.
- improving the processing speed of the morphological segmentation technique.

The lighting and camera configuration used to acquire the images processed in this dissertation, proved to be a problem throughout the thesis. Due to research equipment constraints, it was simply not possible to investigate the affect of lighting and camera position on system performance and optimization. To do this properly, one would need an experimental flotation rig with an easily moveable lighting and camera fixture. The results obtained, clearly illustrate the detriment that improper lighting and camera placement have on the accuracy of the developed morphological segmentation technique. The fact that it has an adverse affect on the segmentation accuracy is important, as the resultant descriptions of poorly illuminated surfaces, give rise to erroneous results. Further research work is therefore needed to investigate the affect of camera and lighting configuration on the optimization of the system.

As graphical results illustrate (see Chapter 6, Figure 6.11 and Chapter 7, Figure 7.8), the morphological segmentation of surface froth images is slow. Optimization of this process to increase processing speed is therefore necessary and can be achieved by improving the software implementation of grey level operations or by implementing the grey level operations in hardware.

- The software implementation of the grey level operations can be optimized by writing inefficient code in assembly language. In practice though, because the software implementation of grey level morphological operations is slow due to the number of computations required, real-time implementation of grey level dilation and erosion is usually achieved by the application of dedicated pipeline (parallel) computer architecture (Wechsler, 1990). However, most morphological hardware architecture is limited to a fixed SE size, typically for a 3x3 neighbourhood and problems arise when the implementation of larger SEs is required. A way around this limit is the decomposition of the SE into smaller SEs, that are capable of being handled by the hardware and whose morphological composition is the given SE (Zhuang and Haralick, 1986) and (Gader, 1991).
- If software optimization fails to achieve an acceptable processing time of less than the required 5 minutes, it may be necessary to implement the system using dedicated hardware. Shih and Mitchell (1989), have shown that threshold

decomposition of grey level operations into binary operations that have the same dimensionality as the original operations, allows grey level morphological dilation and erosion to be implemented using only logic gates in new VLSI pipeline architecture; for any size or any range of input image and SE data. Such an architecture would be able to produce grey level morphological output at real-time video rates. However, the use of dedicated hardware would ultimately depend on a trade-off between processing time and system cost.

9.2.2 SYSTEM VALIDATION

System validation would involve the development of a flotation test rig that implements the fully designed machine vision system, working within the required performance specifications. The availability of such an experimental system would give researchers the opportunity to investigate the affect of various flotation parameters, such as frother concentration, aeration rate and pH level, on froth structure. This is something that at present, cannot be done. A working machine vision system would mean that the size and elipticity measurements could be correlated with the parameter information and thus an investigation of the affect of these parameters into the optimization and control of flotation cells could be undertaken.

References

Boyle R.D. and Thomas R.C., 1988, "Computer Vision - A First Course", *Blackwell Scientific Publications.*

Brink A.D., 1987, "The Selection and Evaluation of Grey Level Thresholds Applied to Digital Images", *M.Sc. dissertation*, Rhodes University, R.S.A.

Canny J., 1986, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, pp. 679-698.

Castleman K.R., 1979, "Digital Image Processing", *Prentice-Hall.*

Chadwick J.R., 1981, "Black Mountain - Annual Revenue about $150 000 000", *World Mining*, Vol. 34, No. 3, pp. 36-39.

Chang L. and Leu K., 1990, "A Fast Algorithm for the Restoration of Images Based on Chain Codes Description and Its Applications", *Computer Vision, Graphics, and Image Processing*, 50, pp. 296-307.

Cunningham R., 1981, "Segmenting Binary Images", *Robotics Age*, pp. 4-19.

Cutting G.W., Barber S.P. and Newton S., 1986, "Effects of Froth Structure and Mobility on the Performance and Simulation of Continuously Operated Flotation Cells", *International Journal of Mineral Processing"*, 16, pp. 43-61.

Davis L.S., 1977, I, "Understanding Shape : Angles and Sides", *IEEE Transactions on Computers*, Vol. C-26, No. 3, pp. 236-242.

Davis L.S., 1977, II, "Understanding Shape : II. Symmetry", *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 204-212.

Delp E.J. and Mitchell O.R., 1979, "Image Compression Using Block Truncation Coding", *IEEE Transactions on Communications*, Vol. COM-27, No. 9, pp. 1335-1342.

Dunn S.M., Keizer K.L. and Jongdaw Y., 1989, "Measuring the Area and Volume of the Human Body with Structured Light", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 6, pp. 1350-1364.

Engelbrecht J.A. and Woodburn E.T., 1975, "The Effects of Froth Height, Aeration Rate and Gas Precipitation on Flotation", *Journal of the South African Institute of Mining and Metallurgy*, pp. 125-132.

Gader P.D., 1991, "Separable Decompositions and Approximations of Greyscale Morphological Templates", *Image Understanding*, Vol. 53, No. 3, pp. 288-296.

Ghosh P.K., 1988, "A Mathematical Model for Shape Description using Minkowski Operators", *Computer Vision, Graphics and Image Processing*, 44, pp. 239-269.

Gold D., 1991, "The Investigation and Design of a Machine Vision System for the Detection and Control of the Separation Blade in a Spiral Ore Concentrator", *M.Sc. dissertation*, University of Cape Town, Cape Town, R.S.A.

Gonzalez R.C. and Wintz P., 1987, "Digital Image Processing", 2nd Edition, *Addison-Wesley Publishing Company.*

Guttman I., Wilks S.S. and Hunter J.S., 1982, "Introductory Engineering Statistics", 3rd Edition, *John Wiley and Sons.*

Haralick R.M. and Shapiro L.G., 1985, "Image Segmentation Techniques", *Computer Vision, Graphics and Image Processing*, 29, pp. 100-132.

Haralick R.M., Sternberg S.R. and Zhuang X., 1987, "Image Analysis using Mathematical Morphology", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 4, pp. 532-550.

Heald M.A., 1984, "On Choosing the Bin Width of a Gaussian Histogram", *American Journal of Physics*, 52, No. 3, p. 254.

Heijmans H.J.A.M., 1991, "Theoretical Aspects of Gray-Level Morphology", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 6, pp. 568-582.

Heijmans H.J.A.M. and Ronse C., 1990, "The Algebraic Basis of Mathematical Morphology 1. Dilations and Erosions", *Computer Vision, Graphics and Image Processing*, 50, pp. 245-295.

Horn B.K.P., 1986, "Robot Vision", *McGraw-Hill Book Company.*

Horst W.E. and Enochs R.C., 1980, "Instrumentation and Process Control", *Engineering and Mining Journal*, Vol. 181, No. 6, pp. 70-92.

Huang K.S., Jenkins B.K. and Sawchuk A.A., 1989, "Binary Image Algebra and Optical Cellular Logic Processor Design", *Computer Vision, Graphics and Image Processing*, 45, pp. 295-345.

Ives J.K. (Editor), 1984, "The Scientific Basis of Flotation", *NATO ASI Series.*

Kelly E.G. and Spottiswood D.J., 1982, "Introduction to Minerals Processing", 1st Edition", *Wiley*, New York.

Kulpa Z., 1977, "Area and Perimeter Measurement of Blobs in Discrete Binary Pictures", *Computer Graphics and Image Processing*, 6, pp. 434-451.

Lacroix V., 1988, "A Three-Module Strategy for Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 6, pp. 803-810.

Lee J.S.J., Haralick R.M. and Shapiro L.G., 1987, "Morphologic Edge Detection", *IEEE Transactions on Robotic Automation*, Vol. RA-3, pp. 142-156.

Liow Y., 1991, "A Contour Tracing Algorithm that Preserves Common Boundaries between Regions", *CVGIP : Image Understanding*, Vol. 53, No. 3, pp. 313-321.

Maragos P. and Schafer R.W., 1990, "Morphological Systems for Multidimensional Signal Processing", *Proceedings of the IEEE*, Vol. 78, No. 4, pp. 690-710.

MATROX Electronic Systems Limited, 1989, "IMAGER-AT, User's Manual", *MATROX Publications Department.*

Miller S., 1984, "Experimental Design and Statistics", 2nd Edition, *Methuen*, London and New York.

Nakagawa Y. and Rosenfeld A., 1978, "A Note on the Use of Local min and max Operations in Digital Picture Processing", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 8, pp. 632-635.

Otsu N., 1979, "A Threshold Selection Method from Gray-Level Histograms", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-9, No. 1, pp. 62-66.

Pavlidis T., 1978, "Survey - A Review of Algorithms for Shape Analysis", *Computer Graphics and Image Processing*, 7, pp. 243-258.

Pearson D.E. and Robinson J.A., 1985, "Visual Communication at Very Low Data Rates", *Proceedings of the IEEE*, Vol. 73, No. 4, pp. 795-812.

Perry R.H. and Green D., 1984, "Perry's Chemical Engineers' Handbook", 6th Edition, *McGraw-Hill Book Company.*

Persoon E. and Fu K., 1977, "Shape Discrimination Using Fourier Descriptors", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-7, No. 3, pp. 170-179.

Popplewell L.M. and Peleg M., 1989, "Calculating the Beta Function from Three Common Particle Size Distributions", *American Institute of Chemical Engineers (AIChE) Journal*, Vol. 35, No. 2, pp. 347-349.

Rosenfeld A. and Kak A.C., 1982, "Digital Picture Processing", 2nd Edition, Vol. 2, *Academic Press*.

Sahoo P.K., Soltani S. and Wong A.K.C., 1988, "A Survey of Thresholding Techniques", *Computer Vision, Graphics, and Image Processing*, 41, pp. 233-260.

Serra J., 1986, "Introduction to Mathematical Morphology", *Computer Vision, Graphics, and Image Processing*, 35, pp. 283-305.

Shapiro L.G., 1980, "A Structural Model of Shape", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 2, pp. 111-126.

Shih F.Y. and Mitchell O.R., 1989, "Threshold Decomposition of Gray-Scale Morphology into Binary Morphology", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 1, pp. 31-42.

Sternberg S.R., 1986, "Grayscale Morphology", *Computer Vision, Graphics, and Image Processing*, 35, pp. 333-355.

Subrahmanyam T.V. and Forssberg E., 1988, "Froth Stability, Particle Entrainment and Drainage in Flotation - A Review", *International Journal of Mineral Processing*, 23, pp. 33-53.

Swokowski E.W., 1984, "Calculus with Analytic Geometry", *Prindle, Weber and Schmidt*.

Tabatabai A.J. and Mitchell O.R., 1984, "Edge Location to Subpixel Values in Digital Imagery", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 2, pp. 188-201.

Tsai W., 1985, "Moment-Preserving Thresholding: A New Approach", *Computer Vision, Graphics, and Image Processing*, 29, pp. 377-393.

Twidle T.R., Engelbrecht P.C. and Koel J.W.S., 1986, "Improvements in Stabilizing Control at Black Mountain", *Journal of the South African Institute of Mining and Metallurgy*, Vol. 86, No. 1, pp. 15-24.

Wang Y.F., Mitiche A. and Aggarwal J.K., 1987, "Computation of Surface Orientation and Structure of Objects using Grid Coding", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 1, pp. 129-137.

Wechsler H., 1990, "Computational Vision", *Academic Press*.

Weszka J.S., 1978, "A Survey of Threshold Selection Techniques", *Computer Graphics and Image Processing*, 7, pp. 259-265.

Wigeson D., 1987, "Fragmentation Analysis Using Computer Vision Techniques", *M.Sc. dissertation*, University of the Witwatersrand, Johannesburg, R.S.A.

Wills B.A., 1981, "Mineral Processing Technology", 2nd Edition, *Pergamon Press*.

Woodburn E.T., Stockton J.B. and Robbins D.J., 1989, "Vision-Based Characterisation of 3 Phase Froths", *International Colloquium - Developments in Froth Flotation*, Vol. 1, pp. 1-30.

Young E.C., 1979, "The New Penguin Dictionary of Electronics", *Penguin Books Limited*.

Zhuang X. and Haralick R.M., 1986, "Morphological Structuring Element Decomposition", *Computer Vision, Graphics and Image Processing*, 35, pp. 370-382.

Appendix A

Segmentation Efficiency and Bubble Identification as a Function of SE Radius

Figures A.1 to A.18, are a sequence of images that illustrate the efficiency of morphologically segmenting the test image BUB1.CFI with discrete spherical structuring elements of radii 3 to 20 pixels, respectively. In each case the segmented image, given by the white lines, has been overlayed on the original image to show the effectiveness of the size of the structuring element used, and white pluses (+) mark the centroids of those identified bubbles whose area falls in the range 25 pixels to 7000 pixels.

Furthermore, the processed area of interest has been demarcated with a white box, with the top left-hand and bottom right-hand coordinates of the box given.

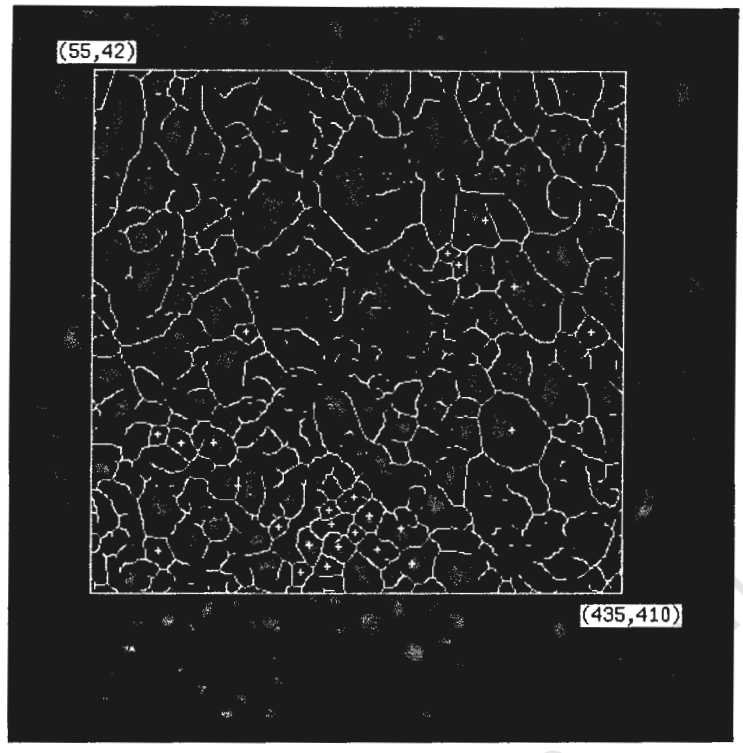


Figure A.1 : BUB1.CFI segmented with a SE of radius 3.

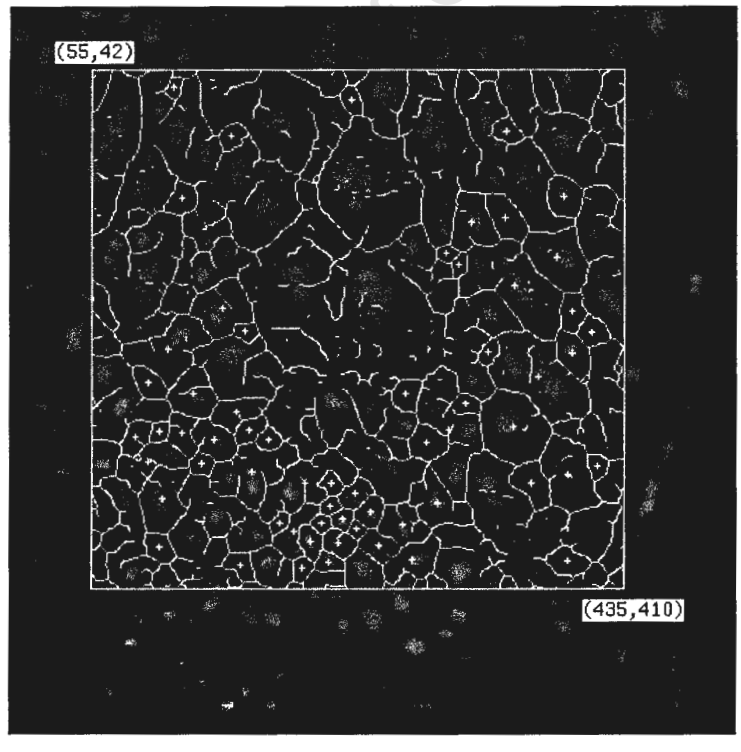


Figure A.2 : BUB1.CFI segmented with a SE of radius 4.

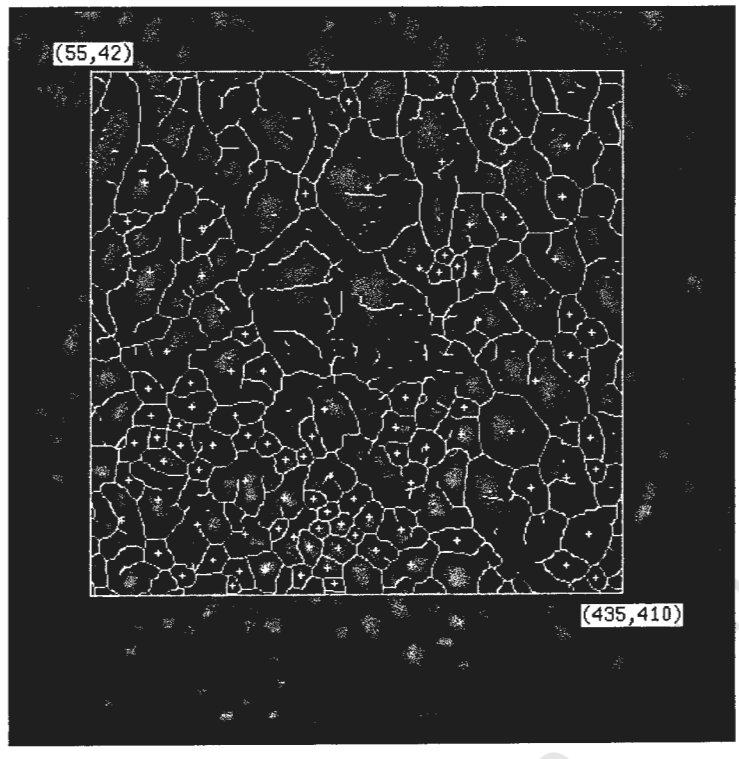


Figure A.3 : BUB1.CFI segmented with a SE of radius 5.

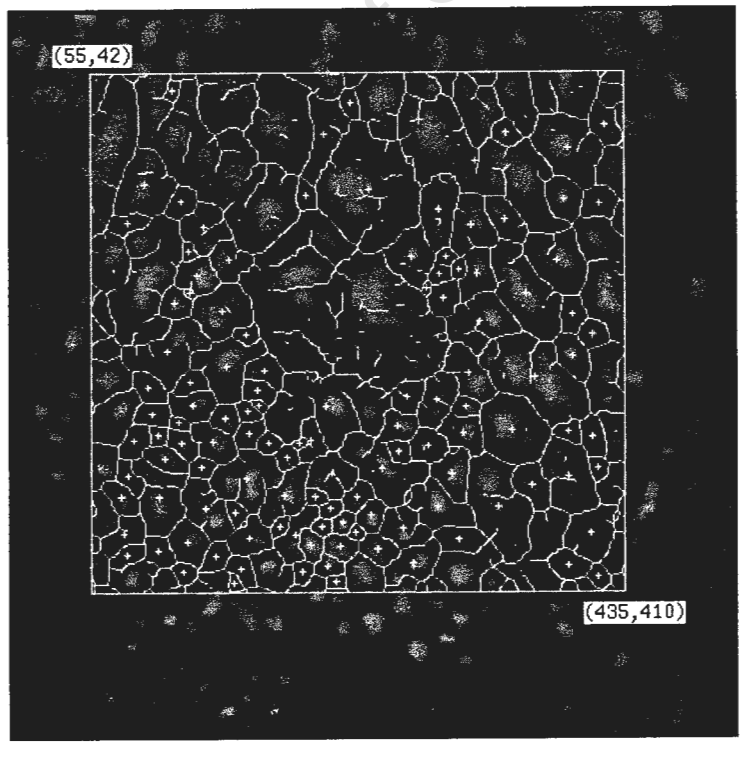


Figure A.4 : BUB1.CFI segmented with a SE of radius 6.

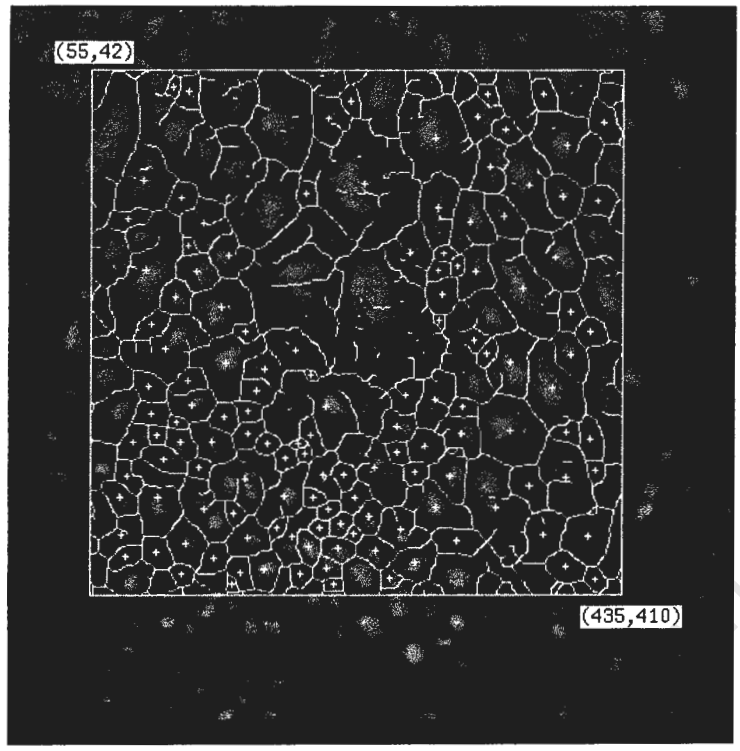


Figure A.5 : BUB1.CFI segmented with a SE of radius 7.

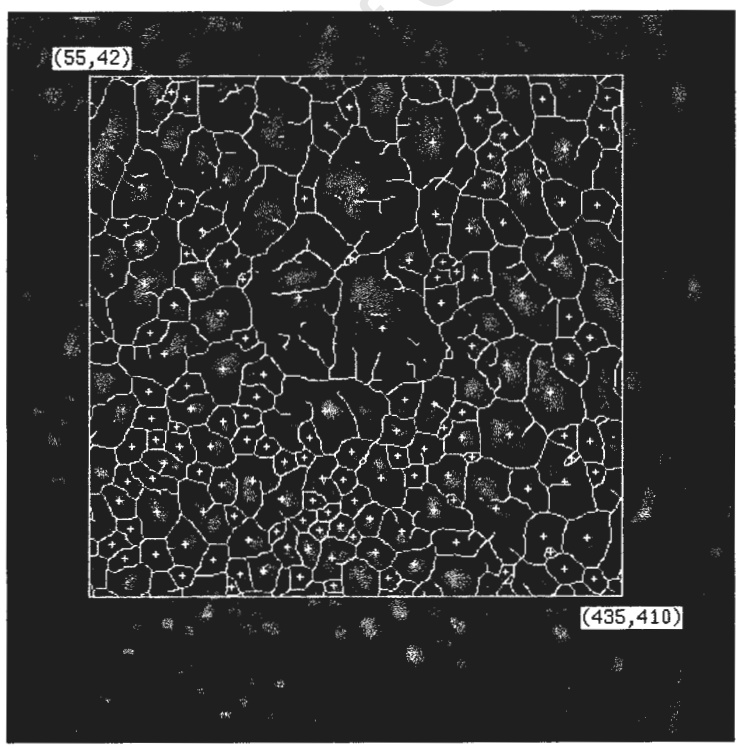


Figure A.6 : BUB1.CFI segmented with a SE of radius 8.

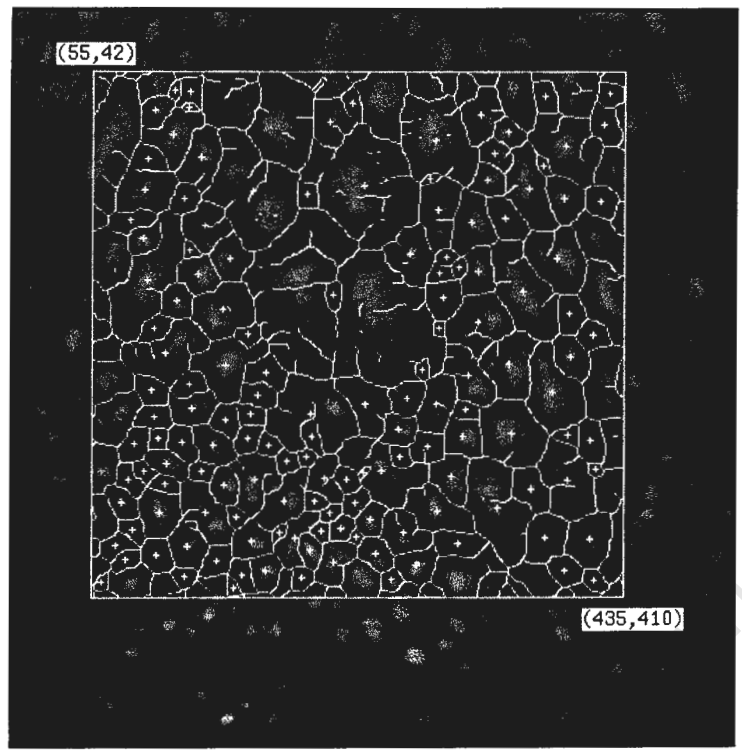


Figure A.7 : BUB1.CFI segmented with a SE of radius 9.

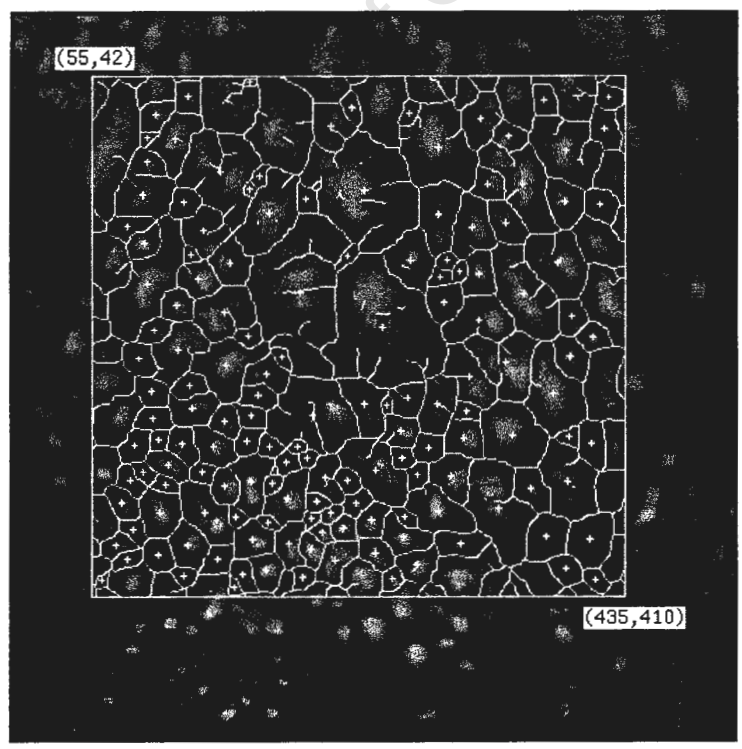


Figure A.8 : BUB1.CFI segmented with a SE of radius 10.

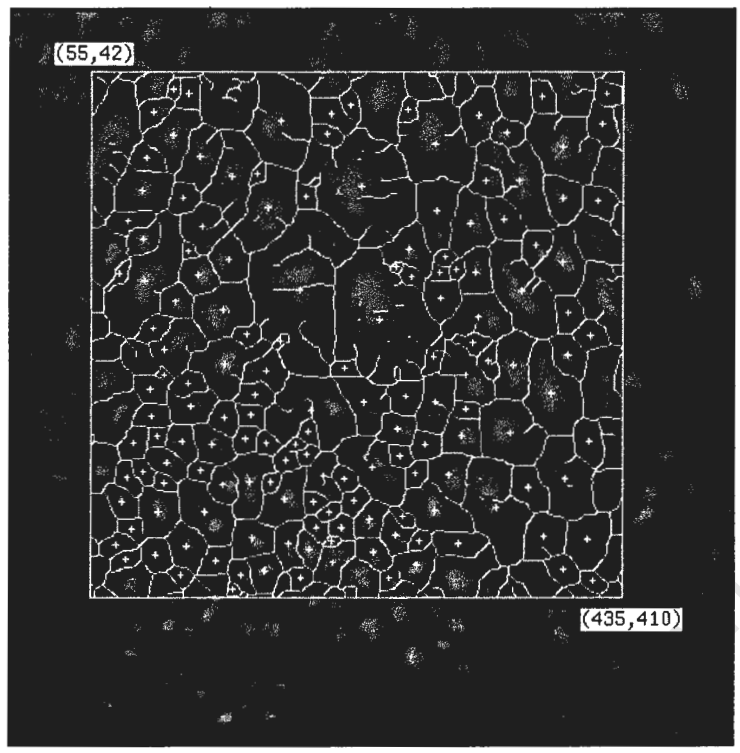


Figure A.9 : BUB1.CFI segmented with a SE of radius 11.

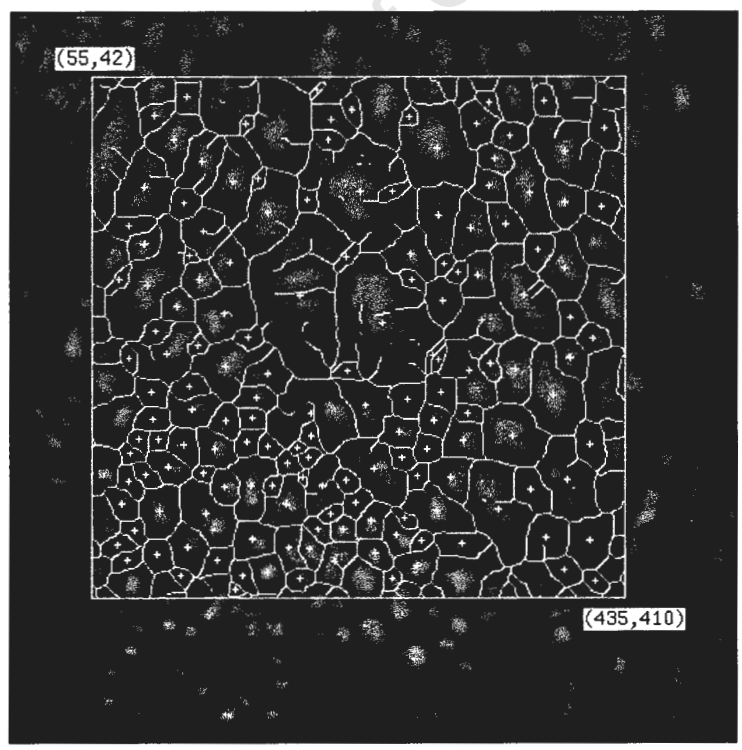


Figure A.10 : BUB1.CFI segmented with a SE of radius 12.

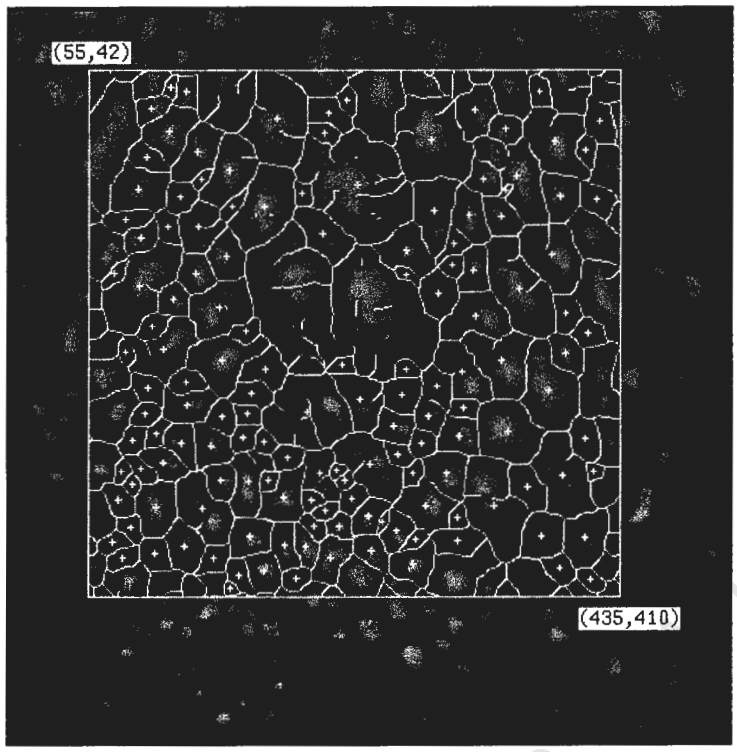


Figure A.11 : BUB1.CFI segmented with a SE of radius 13.

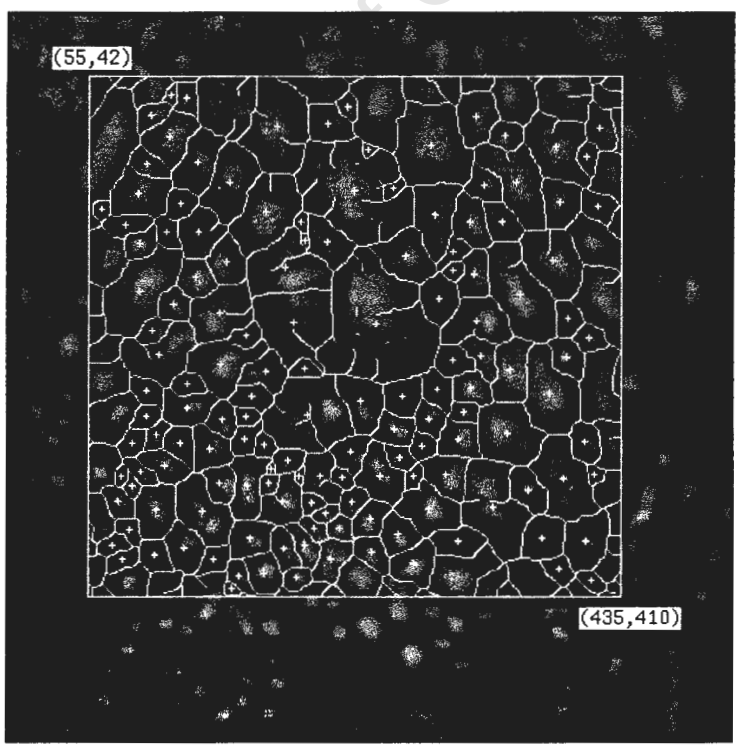


Figure A.12 : BUB1.CFI segmented with a SE of radius 14.

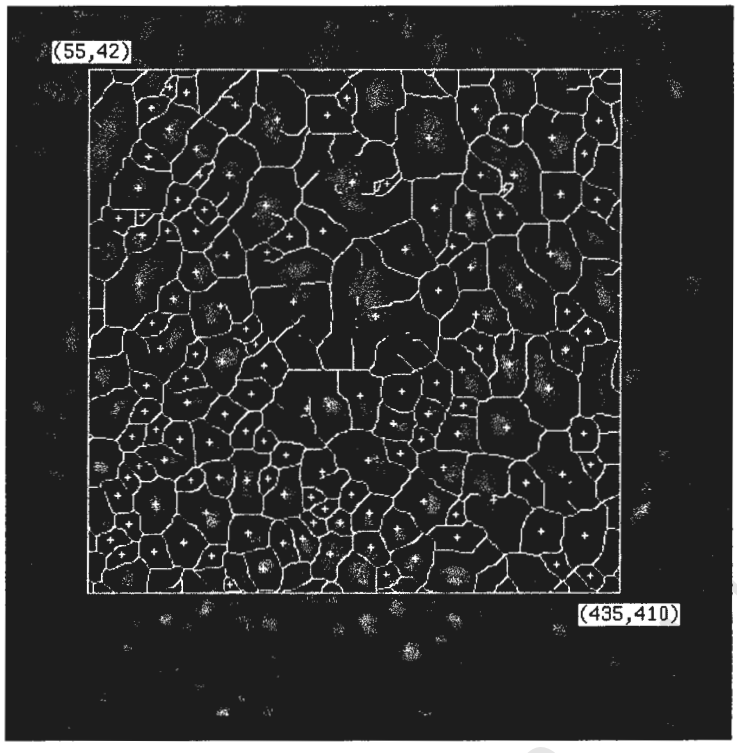


Figure A.13 : BUB1.CFI segmented with a SE of radius 15.

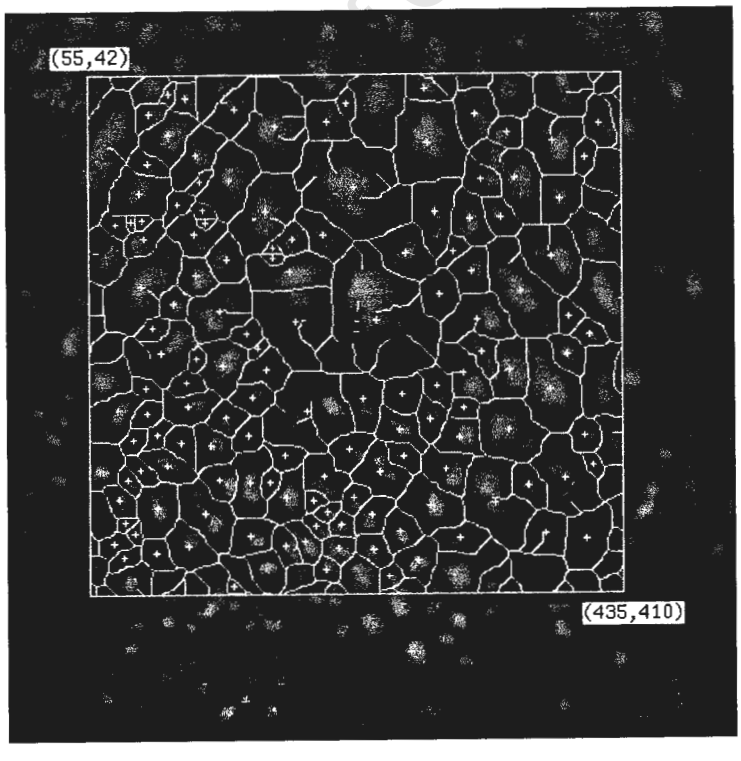


Figure A.14 : BUB1.CFI segmented with a SE of radius 16.

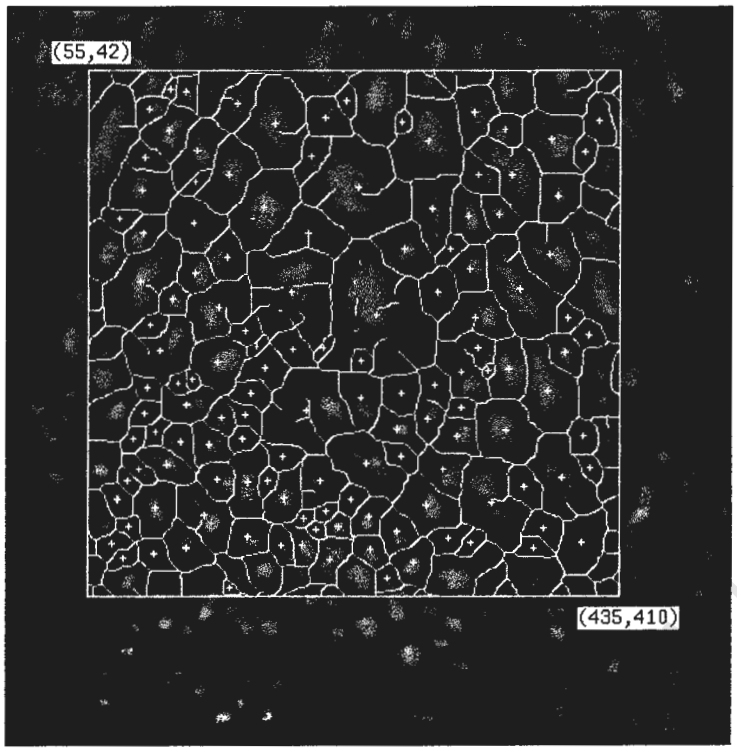


Figure A.15 : BUB1.CFI segmented with a SE of radius 17.

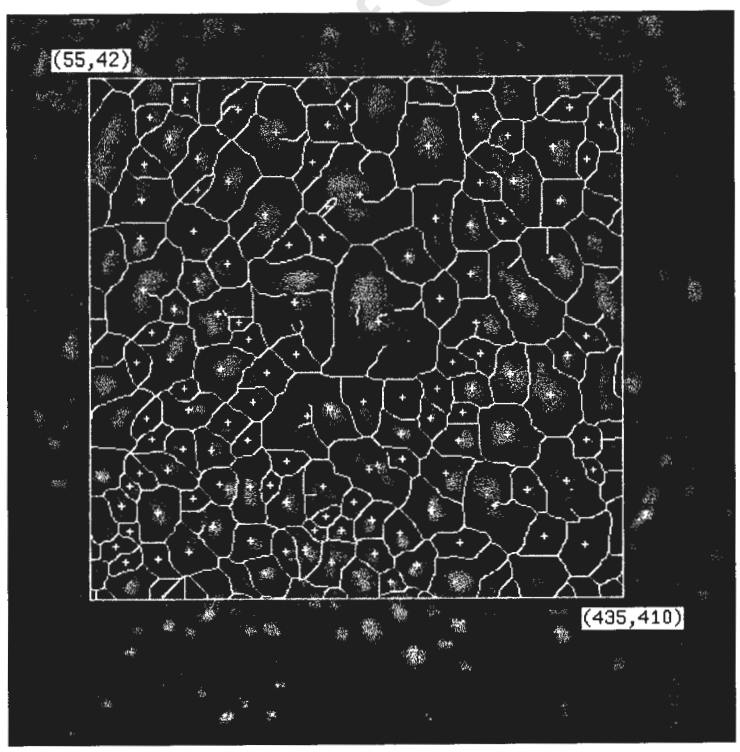


Figure A.16 : BUB1.CFI segmented with a SE of radius 18.

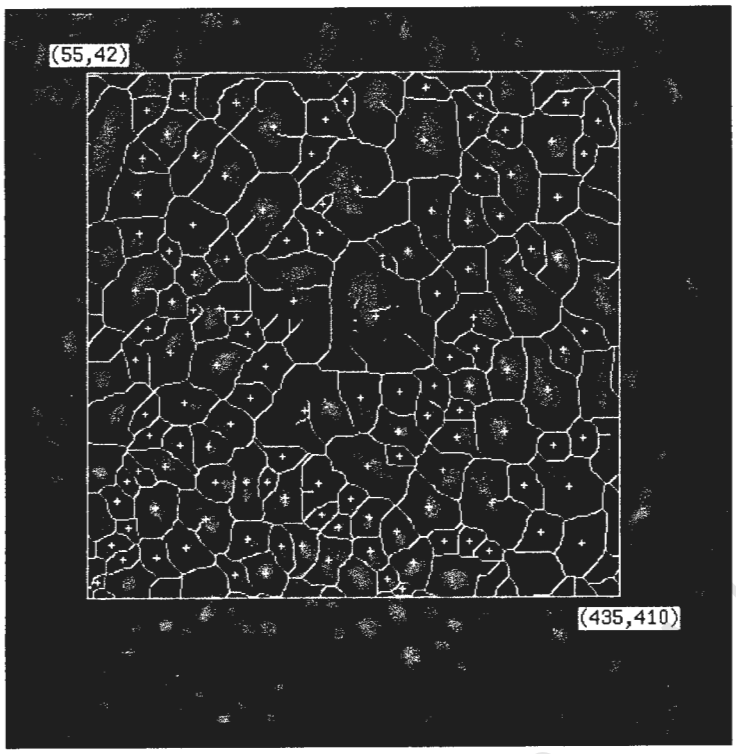


Figure A.17 : BUB1.CFI segmented with a SE of radius 19.

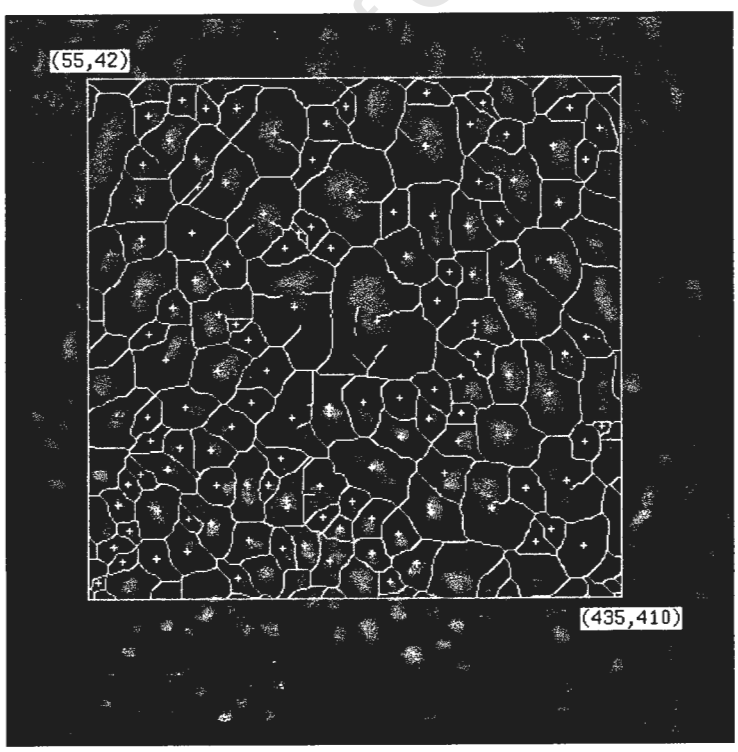


Figure A.18 : BUB1.CFI segmented with a SE of radius 20.

Appendix B

The Derivation of an Elipticity Measure Using Central Moments

The theory of moments can be used to derive a robust and reliable shape factor for measuring the circularity and elipticity of the blobs in segmented surface froth images, by extending the shape factor for an ideal elliptically shaped object.

B.1 A Shape Factor for an Ideal Elliptically Shaped Object

Consider a continuous, ideal elliptically shaped object defined by :

$$f(x,y) = \begin{cases} 1 \text{ if } \dfrac{x^2}{\alpha^2} + \dfrac{y^2}{\beta^2} \leq 1 \\ 0 \text{ otherwise} \end{cases}$$

centred at the origin (0,0), with semi-major axis α on the x-axis, and semi-minor axis β on the y-axis, as illustrated in Figure B.1.

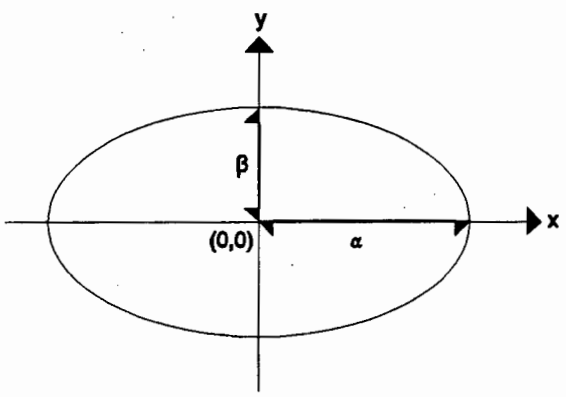


Figure B.1 : An ideal elliptical object.

A measure of the elipticity, or more correctly the elongation of the object is commonly given by (Horn, 1986, p. 53) :

$$E = \frac{\textit{major axis}}{\textit{minor axis}} = \frac{I_a}{I_b}$$

Where I_a is the axis of least inertia and corresponds to the major axis and I_b, is the axis of greatest inertia and corresponds to the minor axis (Horn, 1986, p. 49).

For the ideal elliptically shaped object illustrated in Figure B.1, the elipticity measure (E) can be shown to be :

$$E = \sqrt{\frac{I_a}{I_b}} = \frac{\beta}{\alpha} \qquad \ldots(B.1)$$

B.1.1 Proof

Equation (B.1) can be proved by showing that :

$$E^2 = \frac{I_x}{I_y} = \frac{\int y^2 dA}{\int x^2 dA}$$

where : I_x is the moment of inertia of the object about the x-axis and which will be shown to be the moment of least inertia, I_a.

I_y is the moment of inertia of the object about the y-axis and which will be shown to be the moment of greatest inertia, I_b.

The characteristic equation for the ellipse :

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} = 1$$

can be expressed as :

$$\beta^2 x^2 + \alpha^2 y^2 = \alpha^2 \beta^2$$

which can be rewritten as follows, to give an expression for x in terms of y :

$$x = \pm \frac{\alpha}{\beta} \sqrt{\beta^2 - y^2}$$

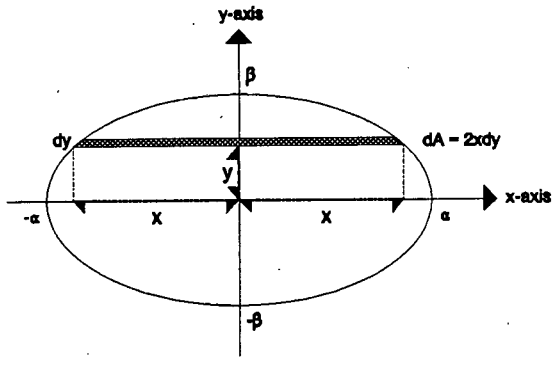


Figure B.2 : Evaluating I_x.

With reference to Figure B.2, the moment of inertia of the object about the x-axis is evaluated as follows :

$$I_x = \int_{y=-\beta}^{\beta} y^2 \, dA$$

$$= 4 \int_{y=0}^{\beta} y^2 x \, dy$$

Substituting for x :

$$I_x = \frac{4\alpha}{\beta} \int_0^{\beta} y^2 \sqrt{\beta^2 - y^2} \, dy$$

The integral is a standard integral of the form (Swokowski, 1984) :

$$\int u^2 \sqrt{a^2 - u^2} \, du = \frac{u}{8}(2u^2 - a^2)\sqrt{a^2 - u^2} + \frac{a^4}{8} \arcsin\left(\frac{u}{a}\right) + C$$

and substituting for u = y and a = β, it evaluates to :

$$I_x = \frac{4\alpha}{\beta}\left[\frac{y}{8}(2y^2 - \beta^2)\sqrt{\beta^2 - y^2} + \frac{\beta^4}{8} \arcsin\left(\frac{y}{\beta}\right)\right]_0^{\beta}$$

$$= \frac{4\alpha}{\beta}\left(\frac{\pi \beta^4}{16}\right)$$

$$= \frac{\pi}{4} \alpha \beta^3$$

A similar procedure for the integration of :

$$I_y = \int_{x=-\alpha}^{\alpha} x^2 \, dA$$

$$= 4 \int_{x=0}^{\alpha} x^2 y \, dx$$

evaluates to :

$$I_y = \frac{\pi}{4} \alpha^3 \beta$$

It can be seen that I_x is the least second moment and that I_y is the greatest second moment. It therefore follows that :

$$E^2 = \frac{I_a}{I_b} = \frac{I_x}{I_y} = \frac{\frac{\pi}{4} \alpha \beta^3}{\frac{\pi}{4} \alpha^3 \beta} = \frac{\beta^2}{\alpha^2}$$

which proves that :

$$E = \sqrt{\frac{I_a}{I_b}} = \frac{\beta}{\alpha}$$

B.2 Extending the Shape Factor to Cater for Non-ideal, Elliptically Shaped Objects

As has been illustrated by the results presented in Chapter 6, sections 6.4.2 and 6.5, the extracted blobs are not ideally shaped circles or ellipses, but tend to have the form illustrated in Figure B.3, having a definite orientation and axis of elongation.

Because the shape factor for an ideal ellipse provides a reliable and robust measurement of shape, it is desirable to extend the concept of this elipticity measure to cover shapes that are roughly circular or elliptical. This is achieved by fitting the best possible ellipse to the object and involves computing the equivalent major and minor axes of the ellipse, as illustrated in Figure B.3.

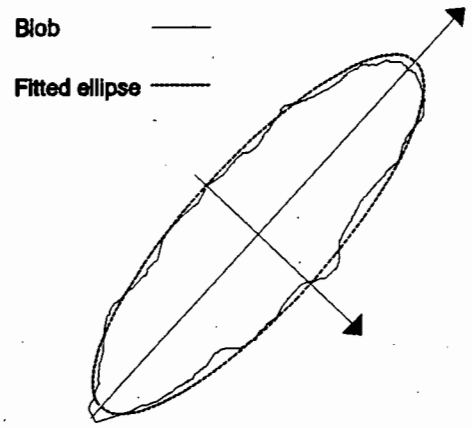


Figure B.3 : A realistic blob, with fitted ellipse, illustrating a definite orientation and axis of elongation.

In practice, the required axes are obtained by computing the extreme (maximum and minimum) moments of inertia for the object; where the axis of elongation is given by the minimum second moment and the corresponding perpendicular axis, by the maximum second moment (Horn, 1986, pp. 49-53).

B.2.1 The Principal Axis

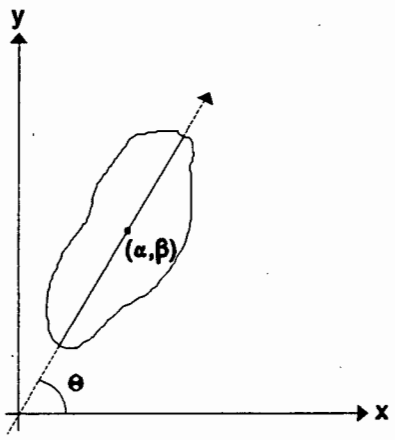


Figure B.4 : Determining the line of minimum inertia.

For an arbitrarily elongated object, Figure B.4, consider a straight line with slope Θ, that passes through the points (α, β) in the object. For any point (x,y), the equation of the line can be written as (Rosenfeld and Kak, 1982, p. 289) :

$$(y - \beta) = (x - \alpha)\tan\Theta$$

or representing the line by the function l(x,y,Θ) the equation can be rewritten as :

$$l(x,y,\Theta) = (x - \alpha)\sin\Theta - (y - \beta)\cos\Theta$$

For a continuous object f(x,y), the moment of inertia about such a line is given by :

$$I = \iint_A l(x,y,\Theta)^2 f(x,y)\,dx\,dy$$

$$= \iint_A [(x - \alpha)\sin\Theta - (y - \beta)\cos\Theta]^2 f(x,y)\,dx\,dy$$

The line of minimum inertia is that line that minimises the square of the distance to the points in the object and is determined by differentiating the integral I with respect to either α or β. Setting the result to zero will solve for the unknowns α and β. Thus differentiating I with respect to α, the integral becomes :

$$\frac{\partial I}{\partial \alpha} = \frac{\partial}{\partial \alpha} \iint_A [(x - \alpha)\sin\Theta - (y - \beta)\cos\Theta]^2 f(x,y)\,dx\,dy = 0$$

which gives :

$$(-2\sin\Theta) \iint_A [(x - \alpha)\sin\Theta - (y - \beta)\cos\Theta] f(x,y)\,dx\,dy = 0$$

Evaluating the expression gives :

$$m_{10}\sin\Theta - \alpha m_{00}\sin\Theta - m_{01}\cos\Theta + \beta m_{00}\cos\Theta = 0$$

$$\bar{x}\sin\Theta - \alpha\sin\Theta - \bar{y}\cos\Theta + \beta\cos\Theta = 0$$

$$(\bar{x} - \alpha)\sin\Theta - (\bar{y} - \beta)\cos\Theta = 0$$

With the equality holding iff :

$$\alpha = \bar{x} \qquad \beta = \bar{y}$$

The minimum inertia line therefore passes through the centroid of f(x,y) and is known as the *Principal Axis* of f(x,y), which can be regarded as the line that "best fits" f(x,y).

B.2.2 The Derivation of the Extreme Moments of Inertia

The extreme moments of inertia can be derived by using the Principal Axis, or by using the

second central moments matrix.

B.2.2.1 The Derivation of the Extreme Moments Using the Principal Axis

The moment of inertia about the line of slope Θ and which passes through $(\bar{x},\bar{y})$ is therefore determined from the integral :

$$I = \iint_A [(x - \bar{x})\sin\Theta - (y - \bar{y})\cos\Theta]^2 f(x,y)\,dx\,dy$$

Evaluating I yields :

$$\begin{aligned} I &= \iint_A [(x - \bar{x})\sin\Theta - (y - \bar{y})\cos\Theta]^2 f(x,y)\,dx\,dy \\ &= \iint_A (x - \bar{x})^2 f(x,y)\sin^2\Theta\,dx\,dy \\ &\quad - 2\iint_A (x - \bar{x})(y - \bar{y}) f(x,y)\sin\Theta\cos\Theta\,dx\,dy \\ &\quad + \iint_A (y - \bar{y})^2 f(x,y)\cos^2\Theta\,dx\,dy \\ &= \mu_{20}\sin^2\Theta - 2\mu_{11}\sin\Theta\cos\Theta + \mu_{02}\cos^2\Theta \end{aligned}$$

Using the trigonometric identities :

$$\sin^2\Theta = \frac{1}{2}(1 - \cos 2\Theta)$$

$$\cos^2\Theta = \frac{1}{2}(1 + \cos 2\Theta)$$

$$\sin\Theta\cos\Theta = \frac{1}{2}\sin 2\Theta$$

the evaluated integral can be rewritten as :

$$\begin{aligned} I &= \frac{\mu_{20}}{2}(1 - \cos 2\Theta) - \mu_{11}\sin 2\Theta + \frac{\mu_{02}}{2}(1 + \cos 2\Theta) \\ &= \frac{1}{2}(\mu_{20} + \mu_{02}) - \frac{1}{2}(\mu_{20} - \mu_{02})\cos 2\Theta - \mu_{11}\sin 2\Theta \end{aligned}$$

The angle Θ for which the quantity I is minimized, is found by differentiating I with respect to Θ and setting the result to zero :

$$\frac{\partial I}{\partial \Theta} = (\mu_{20} - \mu_{02})\sin 2\Theta - 2\mu_{11}\cos 2\Theta = 0$$

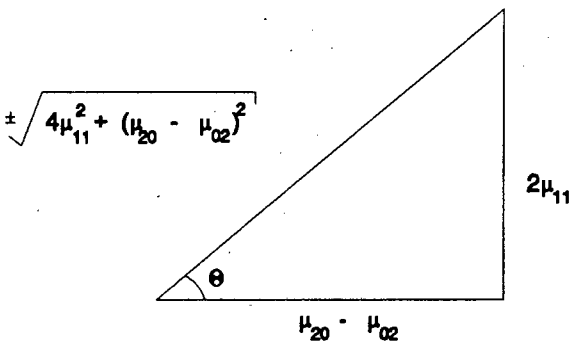
Hence :

$$\tan 2\Theta = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}}$$

Which implies that :

$$\Theta = \frac{1}{2}\arctan\left[\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right]$$

Using Pythagoras, expressions for $\sin 2\Theta$ and $\cos 2\Theta$ can be obtained from the expression for $\tan 2\Theta$ as follows :



$$\sin 2\Theta = \pm \frac{2\mu_{11}}{\sqrt{4\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}}$$

$$\cos 2\Theta = \pm \frac{\mu_{20} - \mu_{02}}{\sqrt{4\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}}$$

The positive values of the above expressions result in the desired minimum value for I and the negative values in the maximum value for I. This can be verified by taking the second partial derivative of I with respect to Θ, which gives :

$$\frac{\partial^2 I}{\partial \Theta^2} = 2(\mu_{20} - \mu_{02})\cos 2\Theta + 4\mu_{11}\sin 2\Theta$$

Substituting for the positive expressions for sin2Θ and cos2Θ, yields :

$$\frac{\partial^2 I}{\partial \Theta^2} > 0$$

indicating a minimum quantity. Similarly, if the negative expressions for sin2Θ and cos2Θ are substituted, the result is :

$$\frac{\partial^2 I}{\partial \Theta^2} < 0$$

which indicates a maximum quantity.

The expression for I can therefore be rewritten as follows, to give an expression for the extreme values of the moment of inertia.

$$I = \frac{1}{2}(\mu_{20} + \mu_{02}) - \frac{1}{2}\left[\frac{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}{\mu_{20} - \mu_{02}}\right]\cos 2\Theta$$

Which was obtained using the fact that :

$$\sin 2\Theta = \frac{2\mu_{11}\cos 2\Theta}{\mu_{20} - \mu_{02}}$$

By substituting the expression for cos2Θ, the extreme values of the moment of inertia thus simplify to :

$$I = \frac{1}{2}(\mu_{20} + \mu_{02}) \pm \frac{1}{2}\sqrt{4\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}$$

As previously stated, the minimum and maximum moments of inertia are defined as I_a and I_b respectively, where I_a corresponds to the major axis and I_b to the minor axis of an equivalent ellipse. From the previously determined quantity for I, the extreme moment values can be written as :

$$I_a = \frac{1}{2}(\mu_{20} + \mu_{02}) + \sqrt{\mu_{11}^2 + \left(\frac{\mu_{20} - \mu_{02}}{2}\right)^2}$$

$$I_b = \frac{1}{2}(\mu_{20} + \mu_{02}) - \sqrt{\mu_{11}^2 + \left(\frac{\mu_{20} - \mu_{02}}{2}\right)^2}$$

B.2.2.2 Derivation of the Extreme Moments of Inertia Using the Second Central Moments Matrix

Alternatively, the problem of computing the extreme moments can be solved by finding the rotation Θ that makes the 2x2 matrix of second central moments, diagonal (Horn, 1986, p. 63). The problem amounts to finding the eigenvalues and eigenvectors of the 2x2 matrix of second central moments given by :

$$[A] = \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix}$$

The eigenvalues of the matrix A are given by :

$$\det(\lambda I - A) = 0$$

Thus :

$$\begin{vmatrix} \lambda - \mu_{20} & \mu_{11} \\ \mu_{11} & \lambda - \mu_{02} \end{vmatrix} = 0$$

$$(\lambda - \mu_{20})(\lambda - \mu_{02}) - \mu_{11}^2 = 0$$

$$\lambda^2 - \lambda\mu_{20} - \lambda\mu_{02} + \mu_{20}\mu_{02} - \mu_{11}^2 = 0$$

$$\lambda^2 - \lambda(\mu_{20} + \mu_{02}) + \mu_{20}\mu_{02} - \mu_{11}^2 = 0$$

Using the standard solution for a quadratic equation :

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

the eigenvalues for the second central moments matrix, are thus given by :

$$\lambda = \frac{1}{2}(\mu_{20} + \mu_{02}) \pm \frac{1}{2}\sqrt{(\mu_{20} + \mu_{02})^2 - 4(\mu_{20}\mu_{02} - \mu_{11}^2)}$$

Simplifying the contents of the discriminant, gives :

$$(\mu_{20} + \mu_{02})(\mu_{20} + \mu_{02}) - 4\mu_{20}\mu_{02} + 4\mu_{11}^2$$

$$= \mu_{20}^2 + 2\mu_{20}\mu_{02} + \mu_{02}^2 - 4\mu_{20}\mu_{02} + 4\mu_{11}^2$$

$$= \mu_{20}^2 - 2\mu_{20}\mu_{02} + \mu_{02}^2 + 4\mu_{11}^2$$

$$= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$$

The eigenvalues of the matrix are thus :

$$\lambda_1 = \frac{1}{2}(\mu_{20} + \mu_{02}) + \sqrt{\mu_{11}^2 + \left(\frac{\mu_{20} - \mu_{02}}{2}\right)^2}$$

$$\lambda_2 = \frac{1}{2}(\mu_{20} + \mu_{02}) - \sqrt{\mu_{11}^2 + \left(\frac{\mu_{20} - \mu_{02}}{2}\right)^2}$$

The axis of minimum inertia (Principal Axis) therefore corresponds to the largest eigenvalue of [A] and the axis of maximum inertia, to the smallest eigenvalue of [A]. It can also be shown (Horn, 1986, p. 63), that the principal axis is in the direction of the eigenvector corresponding to the larger eigenvalue.

B.3 An Elipticity Measure

The elipticity measure E, given in equation (B.1), that was derived for an ideal elliptically shaped object can thus be extended to apply to more general objects that are roughly circular or elliptical in shape, by computing the measure :

$$E = \sqrt{\frac{I_a}{I_b}}$$

where I_a and I_b are the major and minor axes respectively, of the best fitting ellipse. For circular objects where $\mu_{20} = \mu_{02}$ and $\mu_{11} = 0$, the extreme moments I_a and I_b are equal and the elipticity measure reduces to a circularity measure.

Appendix C

Connected Component Analysis - A Detailed Explanation

The connected component analysis of segmented surface froth images was performed using previously written consoft4.c code, that was modified for the purposes of this dissertation and a detailed explanation of the implementation of the algorithm is presented here. This appendix has been compiled using relevant information from the CONSOFT[1] V4.0 Reference Manual.

C.1 The Connected Components Algorithm

Consoft4.c is a C module that performs connected component analysis on thresholded, binary images and is based on the algorithm described by R. Cunningham (1981). The module accepts as input a grey level or binary image, it then thresholds the image if required and identifies individual blobs by connectivity analysis. The blobs are then processed and a basic set of geometric parameters for each blob are produced as output. These parameters can then be used for object identification, object inspection or target tracking.

C.1.1 Overview

Consoft4.c implements the connected component algorithm using three major functions :

promfill() : This function is used to generate several look-up tables (LUTs) that are used by subsequent functions to speed up the calculations of the area, first x and y moments, second x and y moments and the second xy moment, using the left and right ends of a run.

gencrp() : This function thresholds the input image, generates a critical run-length list from the binarized image and computes the window state (*wstate*) in a 3x2 neighbourhood at each end of each run.

genblob() : This function performs connected component labelling and generates relationship linkages between the blobs. It also calculates a set of basic geometric

[1] CONSOFT is a trademark of Aspex Incorporated, 536 Broadway, New York, NY 10012, U.S.A. The CONSOFT software was purchased for use with this dissertation and was modified according to the needs of the dissertation.

parameters for each blob and stores the results in two array structures *b[]* and *bs[]*.

These functions are discussed in greater detail in section C.2.2 and the code for these functions is listed in Appendix F, in the image analysis module *analysis.c*.

C.1.2 Connectivity

Consoft4.c uses 4-connectedness, Figure C.1 (a), for the background and 8-connectedness, Figure C.1 (b), for the foreground (Horn, 1986, p.67), where background pixels in a binary image are considered as having a value of 0 and foreground pixels as having a value of 1.

Figure C.1 : Types of connectivity : (a) 4-connectedness : only edge-adjacent cells are considered neighbours and (b) 8-connectedness : corner-adjacent cells are also considered neighbours.

This means that a foreground pixel can be connected to any of its eight neighbours, but a background pixel can only be connected to its horizontal and vertical neighbours. For example, using 4- and 8-connectedness, the small image patch :

```
.       .
.       .
.. 0 0 0 0 0 0 ..
   0 1 1 0 0 0
   0 0 0 1 1 0
.. 0 0 0 0 0 0 ..
.       .
.       .
```

Would be labelled as :

```
.       .
.       .
.. X X X X X X ..
   X Y Y X X X
   X X X Y Y X
.. X X X X X X ..
.       .
.       .
```

While :

```
.       .
.       .
.. 1 1 1 1 1 1 ..
   1 0 0 1 1 1
   1 1 1 0 0 1
.. 1 1 1 1 1 1 ..
.       .
.       .
```

Would be labelled as :

```
.       .
.       .
.. X X X X X X ..
   X Y Y X X X
   X X X Z Z X
.. X X X X X X ..
.       .
.       .
```

Using 4- and 8-connectedness, consoft4.c systematically labels the blobs in a binary image and represents the labels in a hierarchial tree structure with the "outer" background as the root label. To eliminate ambiguous cases, where blobs touch or are cut off at the image border and are therefore incomplete, the image border is filled with zeros. This means that incomplete blobs are considered to belong to the root or background blob. To illustrate this labelling technique, consider as a first example the binary image illustrated in Figure C.2.

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 1 0 0 0
0 0 1 0 0 0 1 1 0 0
0 1 1 0 0 0 0 1 1 0
0 0 0 1 1 1 1 1 0 0
0 1 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 1 1 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0
```

Figure C.2 : Sample image one.

The final labelled image using 4- and 8-connectedness would be :

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 1 0 0 0
0 0 1 0 0 0 1 1 0 0
0 1 1 0 0 0 0 1 1 0
0 0 0 1 1 1 1 1 0 0
0 2 0 0 0 0 0 0 0 0
0 0 0 3 3 0 0 0 0 0
0 3 3 3 0 0 0 0 4 0
0 0 0 0 0 0 0 0 0 0
```

Figure C.3 : Labelled image.

Where the hierarchial tree structure of the labelled image is :

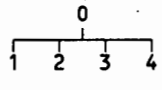


Figure C.4 : Hierarchial tree structure for the labels.

Label 0 is the parent of labels 1, 2, 3 and 4 and therefore labels 1, 2, 3 and 4 are the children of label 0. Labels 1, 2, 3 and 4 are also siblings of each other.

As a second example consider the binary image illustrated in Figure C.5.

```
0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0
0 1 0 0 1 1 0 0 1 0
0 1 0 0 1 1 0 0 1 0
0 1 1 1 1 1 1 1 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 1 0 0 1 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
```

Figure C.5 : Sample image two.

The final labelled image using 4- and 8-connectedness would be :

```
0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0
0 1 2 2 1 1 3 3 1 0
0 1 2 2 1 1 3 3 1 0
0 1 1 1 1 1 1 1 1 0
0 1 4 4 4 4 4 4 1 0
0 1 4 5 4 4 6 4 1 0
0 1 4 4 4 4 4 4 1 0
0 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
```

Figure C.6 : Labelled image.

Where the hierarchial tree structure of the labelled image is :

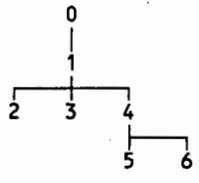


Figure C.7 : Hierarchial tree structure for the labels.

Label 0 is the parent of label 1 and therefore 1 is the child of 0. Label 1 is the parent of labels 2, 3 and 4 and labels 2, 3 and 4 are therefore children of 1. Labels 2, 3 and 4 are also siblings. Label 4 is the parent of labels 5 and 6 and labels 5 and 6 are therefore siblings, that have label 4 as their parent.

The binary images are labelled by scanning a 2x2 window, Figure C.8, over the original image from left to right, top to bottom.

```
A B
C H
```

Figure C.8 : 2x2 labelling window.

The window is used to determine the connectivity of pixel H, to its neighbours A,B and C, where H is the home pixel and A, B and C are the neighbouring pixels. The use of this 2x2

window in performing connectivity analysis is discussed further in section C.2.2 (C).

A difficulty with the labelling problem is that, depending on the shape of the input features, it is possible for there to be equivalent labels. For example, in the case when there are "U" shaped features in an image, when the window is scanned over the original image, the tips of the "U" shape are initially given different labels. When the window reaches the point where the two parts meet at the bottom of the "U" shape, the two labels are found to be equivalent. Consoft4.c handles this problem by using an equivalence list to keep track of all labels that are the same.

It is important to note that with connected component labelling, the time taken to compute the connectivity of an image, is data dependent. In other words, the more complex the binary image, the longer it takes to identify the connected components.

C.2 The Implementation of the Connected Components Algorithm

The CONSOFT package was written in Microsoft C, for use with an IBM or compatible computer running under MS-DOS V3.0 or later. The 640 Kbyte memory restriction of MS-DOS limits the size of executable programs which can be run and for this reason consoft4.c was written to work on 256x256x8 images that require 64 Kbytes (or 1 data segment) for storage. In order to process 512x512x8 images on the SUN workstation, the software required several minor modifications to the units *consoft4.h* and *consoft4.c*. These modified units were renamed *analysis.h* and *analysis.c* for the purpose of this dissertation and what follows is a detailed discussion of these units.

C.2.1 The Header File Analysis.h

This file contains constants, compiler directives and structure definitions that are used in the module *analysis.c*. The compiler directives can be used to increase the processing speed by selectively processing blob parameters of interest, such as area, circularity or elipticity.

The constants and their default values in *analysis.h* are :

#define MAXBLOB 1000 This defines the maximum number of blobs in the image. An image may not necessarily contain this number of blobs,

because discarded (merged) blobs may not have been reused.

#define MAXRUN 65535 This is the (modified) maximum allowed number of runs in the function *gencrp()*.

#define MAXCRP 65535 This is the (modified) maximum allowed number of critical runs in the function *gencrp()*.

#define MAXNUM 65535 This is the (modified) maximum allowed number of blob pointers in the function *genblob()*.

#define NROW 512 This is the (modified) maximum number of rows in an image.

#define MAXROW 511 This is the (modified) maximum row (starting from 0).

#define NCOL 512 This is the (modified) maximum number of columns in an image.

#define MAXCOL 511 This is the (modified) maximum column (starting from 0).

#define ENDLINE 0x40 This is the value for a flag to test for the end of a row. It should **not** be changed.

The compiler directives and their default settings in *analysis.h* are :

#define AREA 1
0 = do not compute the blob area.
1 = compute the blob area.

#define PERIM 1
0 = do not compute the blob perimeter.
1 = compute the blob perimeter.

#define PERFLAG 0
0 = use city block metric for perimeter calculation. This assumes a 1:1 aspect ratio.
1 = perform corrected perimeter calculation. This assumes a 5:4 aspect ratio.

#define N1MOM 1
0 = do not compute the first x and y moments.
1 = compute the first x and y moments.

#define N2MOM 1
0 = do not compute the second x and y moments.
1 = compute the first x and y moments.

#define BINARYMOD 1
0 = the input image is a grey level image.
1 = the input image is a binary image.

#define ENCLBOX 0
0 = do not compute the minimum x, maximum x, minimum y and maximum y coordinates.

	1 = compute the minimum x, maximum x, minimum y and maximum y coordinates.
#define BORD 0	0 = do not compute an x coordinate at the maximum y coordinate (border point).
	1 = compute an x coordinate at the maximum y coordinate (border point).
#define TREE 0	0 = traverse blobs using doubly linked list.
	1 = traverse blobs using hierarchial blob relationship.
#define DEBUG 0	0 = do not print out debugging information.
	1 = print out debugging information.

The defined data structures in *analysis.h* and an explanation of their fields are :

Data structure for holding basic blob properties

```
typedef struct blob {
                int flink;
                int rlink;
                int parent;
                int child;
                int sibling;
                unsigned char color;
                unsigned char xbord;
                unsigned int number;
                unsigned int perim;
                int nholes;
                int imin;
                int imax;
                int jmin;
                int jmax;
                    } BLOB;
```

In the source file *analysis.c*, the array of structures *b[]*, where :

struct blob b[MAXBLOB];

is used to hold the basic properties of the detected blobs.

Due to the labelling process used, not all the elements within the array *b[]* will be valid and two methods can be used to traverse the valid blob entries and to avoid any empty entries.

The first method uses the linked list information contained in *b[]* :

```
int flink;          /* Forward link */
int rlink;          /* Reverse link */
```

Each blob has a forward and reverse link to other blob entries, as illustrated in Figure C.9.

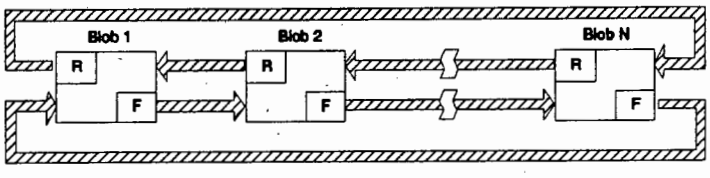


Figure C.9 : Blob traversal using doubly linked lists.

Starting at the background blob (blob 1), the complete list of blobs can be traversed by following the forward or reverse links. A doubly linked list is used to give a closed chain, so that the list can be traversed in either direction and that the list traversal can always be terminated, say by checking for the reoccurrence of the background blob. Note though, that the blob numbers and the order in which they are linked have no necessary relationship to any spatial arrangement of the blobs.

The second method of traversing the list of blob entries, is to use the tree relational parameters contained in *b[]* :

```
int parent;     /* Surrounding blob                                  */
int child;      /* Leftmost surrounded blob in tree                  */
int sibling;    /* Blob to the right in the tree, similarly surrounded */
```

If blobs B and C are completely surrounded by blob A, then blobs B and C are siblings of each other and both are children of the parent A. Each of the three fields hold an index which points to the related blob and therefore provide a hierarchial description of blob relationship. If no such blob exists, the field contains -1. These can then be used to indicate the termination of a branch on the tree.

The remaining fields of the *b[]* are :

unsigned int color — 0 = black blob, 1 = white blob. This indicates whether the grey level values in the blob fell below (0) or above (1) the threshold level.

unsigned char xbord — This is the x coordinate of a pixel whose y coordinate is maximum (see jmax below). It can be used as an initial point for perimeter tracing.

unsigned int number — This is an arbitrary but unique number assigned to a blob. It is not normally used.

unsigned int perim — This is the calculated perimeter of the blob. The perimeter can be calculated in two ways. Firstly, in the "city block" perimeter mode each pixel is assumed to have a square aspect ratio of 1:1. Here, the perimeter of an isolated pixel would be 4. Secondly, in the corrected mode the aspect ratio is assumed to be 5:4 and is only used if the input source is a camera.

int nholes — This indicates the number of holes (children) in a blob.

int imin — This is the minimum (leftmost) x coordinate of a blob.

int imax — This is the maximum (rightmost) x coordinate of a blob.

int jmin — This is the minimum (uppermost) y coordinate of a blob.

int jmax — This is the maximum (lowermost) y coordinate of a blob.

The last four parameters imin, imax, jmin and jmax, define the enclosing box of a blob. Note though, that many pixels in a blob may share the same minimum or maximum x and y values and that the corners of the enclosing box (imin,jmin) and (imax,jmax) are not necessarily points on the blob.

Data structure for holding additional geometric parameters for a blob

```
typedef struct slob {
                unsigned long area;
                unsigned long sumi;
                unsigned long sumj;
                unsigned long sumi2;
                unsigned long sumj2;
                float sumij;
                   } SLOB;
```

In the source file *analysis.c*, the array of structures *bs[]*, where :

struct slob bs[MAXBLOB];

is used to hold additional blob parameters. The fields of *bs[]* are :

unsigned long area — This is the number of pixels in a blob. It is not corrected for aspect ratio but can be, by using a multiplicative constant (see Chapter 7, section 7.1).

unsigned long sumi — This is the first order moment of a blob with respect to the x coordinate. It is calculated by summing the x coordinates of all pixels in the blob (see Chapter 7, section 7.2.2 (A)).

unsigned long sumj This is the first order moment of a blob with respect to the y coordinate. It is calculated by summing the y coordinates of all pixels in the blob (see Chapter 7, section 7.2.2 (A)).

unsigned long sumi2 This is the second order moment of a blob with respect to the x coordinate. It is calculated by summing the square of the x coordinate for each pixel in the blob (see Chapter 7, section 7.2.2 (A)).

unsigned long sumj2 This is the second order moment of a blob with respect to the y coordinate. It is calculated by summing the square of the y coordinate for each pixel in the blob (see Chapter 7, section 7.2.2 (A)).

float sumij This is the second order moment of a blob with respect to the x and y coordinates. It is calculated by summing the products of the x and y coordinates of all pixels in the blob.

Data structure for holding blob parameters for characterising surface froth structures

```
typedef struct glob {
                unsigned long area;
                int xcent;
                int ycent;
                float round;
                float eccent;
                    } GLOB;
```

In the source file *analysis.c*, the array of structures *bg[]*, where :

struct glob bg[MAXBLOB];

is used to hold the blob parameters that are used to characterise surface froth structures. The fields of *bg[]* are :

unsigned long area This is the area of a blob.

int xcent This is the x centroid of a blob.

int ycent This is the y centroid of a blob.

float round This is the circularity measure for a blob (see equation (7.2), Chapter 7, section 7.2.1).

float eccent This is the elipticity measure for a blob (see equation (7.14), Chapter 7, section 7.2.2 (B)).

Data structure for holding the statistical measurements for characterising surface froth images

```
typedef struct stats {
                float mean;
                float var;
                float stddev;
                float skew;
                } STATS;
```

This structure holds the statistical measurements for characterising segmented and analyzed surface froth images. The statistical data is computed using all blobs whose area falls within a specified range. The fields are :

float mean	Average data value (see equation (7.17), Chapter 7, section 7.5.1).
float var	Variance of the data (see equation (7.19), Chapter 7, section 7.5.1).
float stddev	Standard deviation of the data (see equation (7.18), Chapter 7, section 7.5.1).
float skew	Skewness of the data (see equation (7.20), Chapter 7, section 7.5.1).

C.2.2 The C Source File Analysis.c

The functions *image_analysis()*, *zero_nonaoi()*, *promfill()*, *gencrp()*, *genblob()*, *getablob()*, *update()*, *alloc()*, *surround()*, *updperv()*, *updperh()*, *m_errmsg()*, *characterise()*, *moment()* and *mark_centroid()* constitute the module *analysis.c*. Since the heart of the connectivity analysis algorithm is implemented by the functions *promfill()*, *gencrp()* and *genblob()*, these functions are discussed in detail..

(A) The Function promfill()

This function generates the following look-up tables (LUTs) :

```
unsigned int proma[512];        /* Right sum prom (area) */
unsigned int promb[512];        /* Left sum prom (area)  */
unsigned long promc[512];       /* Right sum x prom      */
unsigned long promd[512];       /* Left sum x prom       */
unsigned long prome[512];       /* Right sum x^2 prom    */
unsigned long promf[512];       /* Left sum x^2 prom     */
unsigned int promg[512];        /* Multiplication table  */
```

which are used to speed up the calculations of the area, first x and y moments, second x and y moments and the second xy moment, using the left and right ends of a run. The moment LUTs are generated by taking advantage of the fact that j, the row (or line) number, is

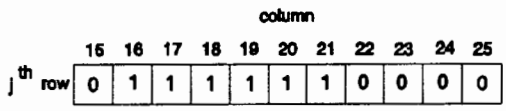
constant over the length of the run and since the run covers consecutive columns (i), the moments involving i can be computed using the closed forms for sums of powers of consecutive integers (see the listing for *promfill()* which is given in Appendix F). A variable *pw* (pixel width), which defines the aspect ratio of the image, is used to construct the LUTs to make the geometric parameters rotation invariant.

As an illustration of how the LUTs are used, consider Table C.1 which shows the computed LUT values for the first twenty five columns of an image, that is for i = 0 to 24.

i	proma	promb	promc	promd	prome	promf
0	0	0	0	0	0	0
1	1	0	1	0	1	0
2	2	1	3	1	5	1
3	3	2	6	3	14	5
4	4	3	10	6	30	14
5	5	4	15	10	55	30
6	6	5	21	15	91	55
7	7	6	28	21	140	91
8	8	7	36	28	204	140
9	9	8	45	36	285	204
10	10	9	55	45	385	285
11	11	10	66	55	506	385
12	12	11	78	66	650	506
13	13	12	91	78	819	650
14	14	13	105	91	1015	819
15	15	14	120	105	1240	1015
16	16	15	136	120	1496	1240
17	17	16	153	136	1785	1496
18	18	17	171	153	2109	1785
19	19	18	190	171	2470	2109
20	20	19	210	190	2870	2470
21	21	20	231	210	3311	2870
22	22	21	253	231	3795	3311
23	23	22	276	253	4324	3795
24	24	23	300	276	4900	4324

Table C.1 : Computed LUTs for the first twenty five columns of an image.

Consider the following section from the j^{th} row of an image, which contains part of a white (1) blob.



By standard calculations we have :

Blob area = 6

Sum x = 111

Sum x^2 = 2071

Using the LUTs we have :

Blob area = proma[21] - promb[16] = 21 - 15 = 6

Sum x = promc[21] - promd[16] = 231 - 120 = 111

Sum x^2 = prome[21] - promf[16] = 3311 - 1240 = 2071

It can be seen that the LUTs are an efficient and quick method for computing blob statistics using each end of a run.

(B) The Function gencrp()

This function thresholds the input image, if it is not already a binary image. It then compresses the image using run-length coding and generates a run-length list. At the end of each run the window state (*wstate*) in a 3x2 neighbourhood is computed and this information is then used in *genblob()* to generate the individual blobs. The crp list is a 1-D array of MAXCRP elements and is defined globally in the module *analysis.c* as :

```
unsigned int cr[MAXCRP] = {0,0};          /* CRP list */
```

and in the function *gencrp()* this list is accessed by the pointer **crpptr*, where :

```
unsigned int *crpptr;                     /* Pointer to CRP list */
```

points to the start of the critical run-length list, by the assignment :

```
crpptr = (unsigned int *)cr;
```

In generating the critical run-length list, consoft4.c uses a 16 bit (unsigned integer) value, as illustrated in Figure C.10, to store all the relevant information for a run.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Figure C.10 : 16 bit value used by consoft4.c for holding the CRP data.

Where : Bit D_{15} is used to indicate whether the entry in the table *cr[]* is a pointer to a valid blob record. If the bit is 0, that is the number is positive, then the entry is a valid pointer to an active blob record. If the bit is 1, that is the number is negative, then the original blob record has been merged and released and the corresponding positive value of the entry is the number of the blob that was merged with the original blob.

Bits D_6-D_{14} are the column position of the end of a run, for which the computed window state is valid. As there are 9 bits for holding column information, a total number of $2^9 = 512$ (or 1FF hexadecimal) columns can be processed.

Bits D_0-D_5 are used to store the window state and perimeter information, that is determined at the end of a run. The information is computed using a 3x2 window of the format :

```
D B
C A
F E
```

Pixels A, B, C, D are used to determine 1 of 16 possible window states which has occurred and the information is used by the connectivity algorithm. The 16 possible window states are discussed fully in the next section. Pixels E and F are used in the corrected perimeter calculation in order to determine whether the perimeter is going to be diagonal or not. The bits are arranged as illustrated in Figure C.11.

D5	D4	D3	D2	D1	D0
F	E	A	B	C	D

Figure C.11 : D_0-D_5 bit assignments.

As will be discussed in the next section, these 6 bits are used to access the corrected perimeter left and right LUTs, *lptrans[]* and *rptrans[]*.

(C) The Function genblob()

This function uses the critical run-length list generated by the function *gencrp()* to construct the individual blobs and to calculate the blob parameters, which are stored in the respective arrays *b[]* and *bs[]*. In generating the blobs, *genblob()* calls the additional functions *getablob()*, *update()*, *alloc()*, *surround()*, *updperv()* and *updperh()*.

Genblob() scans through the list of critical run-lengths and uses connectivity analysis to construct individual blobs. If any pixel of the run just completed is 8-connected to a pixel of the same colour on the previous line, an existing blob is extended to include this run. Otherwise a new blob record is allocated. Each blob is assigned a unique number and pixels

belonging to the blob are labelled with this number in the run-length list.

The merging of equivalent blobs involves a table of blob pointers (*bp[]* in the program), which is indexed by the blob number, *bn*. If the entry in the table for a given blob is positive (MSB=0), then it is a valid pointer to an **active** blob record. If the entry is negative (MSB=1), then the original blob record has been merged and released. The corresponding positive value of the entry is the number of the blob that was merged with the original blob. If more than one merge has occurred in the same blob, it may be necessary to chain through several locations to get to the required active blob.

The hierarchial relationships are updated whenever a blob is completed, as detected by the surrounding blob closing on itself. When this occurs, the surrounded blob is inserted into the list of children of the current blob by manipulating the "parent", "child" and "sibling" pointers in the two blob records.

The moments and other blob statistics could be computed on a pixel-by-pixel basis, but it is computationally faster to update the statistics at the end of each run, as this requires far fewer computations, as was discussed in section C.2.2 (A). Also, new blob records are not allocated until the first run of a new blob has ended.

Whenever a new blob record is allocated (the procedure *alloc()* in the program), its moments, perimeter and hole count are zeroed. The blob's minimum enclosing rectangle is initialized by setting *jmin* to the current line number *j*, *imin* to the starting column of the run and *imax* to the last column in the run.

As new runs are added to the blob on successive lines, the statistics are updated (the function *update()*), using the LUTs computed by the function *promfill()* (section C.2.2 (A)). The enclosing rectangle is updated by checking if the left or right end points of the new run exceed the old values of *imin* or *imax*, respectively. Since the blob grows steadily downward, *jmax* is simply set to *j* and there is no need to change *jmin*. Thus *jmin* is only updated when two blobs merge, in which case the other extreme must be checked and changed as well, to accurately represent the extent of the resulting merged blob.

The perimeter is incremented by two to account for the vertical boundary segments at the end of the run. The contributions to the perimeter of the horizontal boundary segments are handled specially and are discussed later.

The various tasks described above, take place whenever there is a transition from black to white or from white to black in either the current line or the previous line. To detect these transitions, the image is scanned in raster order with the following 2x2 window :

```
D B
C A
```

whose lower right corner (pixel A), is always the current pixel. The previous pixel on the same line and the two corresponding pixels on the previous line, make up the other elements inside the window. Since each pixel can be either black (0) or white (1), there are 2^4 or 16 possible states that can appear in the window. The 16 possible window states are :

```
0 :  0 0     15 :  1 1
     0 0           1 1

1 :  1 0     14 :  0 1
     0 0           1 1

2 :  0 0     13 :  1 1
     1 0           0 1

3 :  1 0     12 :  0 1
     1 0           0 1

4 :  0 1     11 :  1 0
     0 0           1 1

5 :  1 1     10 :  0 0
     0 0           1 1

6 :  0 1      9 :  1 0
     1 0           0 1

7 :  1 1      8 :  0 0
     1 0           0 1
```

The window states are numbered by treating the values of the pixels in the window as a four bit binary number, according to the formula :

$$2^3A + 2^2B + 2^1C + 2^0D = 8A + 4B + 2C + D$$

In addition to the 16 states, the critical run generator looks at the following 3x2 window :

```
D B
C A
F E
```

The additional two pixels below the current pixel (A) and the previous one, are needed for the corrected perimeter calculation, to determine whether the perimeter is going to be

diagonal or not.

The first six bits in the variable *wstate* are arranged : F E A B C D. In *genblob()* these six bits are extracted from *cr[crp]* into the variable *twobysix*. The first 4 bits of *twobysix* are used to control the 16 states described above and the full 6 bit number in *twobysix* is used to index the corrected left and right perimeter LUTs *lptrans[]* and *rptrans[]*.

Although there are 16 distinct window states, there are actually only seven cases to consider. Almost all transitions from black to white are dealt with in the same way as those for white to black. Of the eight pairs of window states, two (0/15) and 5/10) require no action at all, since there is no horizontal transition. A detailed discussion of the remaining six pairs of window states follows.

In this discussion, as well as in the program, *I* is the position of the column corresponding to the column coordinate of pixel A in the 2x2 window. The variables *curreg* and *regabv* are initially pointers to the blob records corresponding to pixels C and D, respectively. At the beginning of each line, *curreg* and *regabv* are bot set to the background (0), since there is one pixel margin of background around the entire image. The scan therefore starts in the second column from the left. In the program the variable *wstate*, derived using the state labelling formula, identifies which window state has occurred.

Case 7/8 : This is or may be the start of a new blob and the variable *newflg* is set to true. It is also the beginning of a new horizontal boundary segment (which will contribute to the perimeter) and a new run. The *I* coordinate is saved in the variable *lftend* for later use in the update and the statistics of the run just completed are updated.

Case 4/11 : Although this is not the start of a new run, it does mark the beginning of a new horizontal boundary segment and therefore *lftend* is again set to *I*. The variable *regabv* is set to the blob assignment of pixel B. This is obtained by looking it up in the run-length list.

Case 3/12 : The run-length list and blob statistics for the run just ended are updated and the variables *curreg* and *regabv* are both set to the blob assignment of pixel B.

Case 2/13 : If *newflg* is true, a new blob record is allocated for the run just completed. The variable *curreg* is set to the number of the new blob record and *newflg* is set to false. The statistics of the new blob are updated, including the perimeter, which is incremented from zero by (*I* - *lftend*). The variable *curreg* is set to *regabv*.

Case 9 : If *newflg* is true, a new blob is allocated and updated as in case 2/13. The variable *curreg* is set to the current value of *regabv* of the blob label of pixel B.

Case 6 : As the background is only 4-connected, a possible hole has been discovered. In addition a new run and possibly a new blob is starting, so cases 1/14 and 7/8 are combined.

Case 1/14 : There are three possibilities here. If the variable *newflg* is true, then what was originally assumed to be a new blob has merged with an existing blob. If *newflg* is false, then either two distinct blobs have merged or a blob has closed on itself, surrounding the blob containing pixel D. In the routine *surround()*, *regabv* (pixel D's blob) is saved in the variable *tmp* and *regabv* is set to pixel B's label. If *newflg* is true, *curreg* is set to the new *regabv* and *newflg* is set to false. This will include the new run in the old blob.

If *newflg* is false and *curreg* is not equal to the new *regabv*, then the two blobs are merged by releasing *regabv*, deleting it from the sequential blob list and combining both blob's statistics into *curreg*. The blob number table *bp[regabv]* is updated to make *regabv* equivalent to *curreg*. If, however, *newflg* is false and *curreg* is equal to *regabv*, then *tmp* is added to the list of children of *curreg*, since it is indeed a hole in *curreg* and the hole count is incremented by one.

Regardless of *newflg*, this case also terminates a horizontal boundary segment of *tmp* and *curreg*. If *holreg* is not really a hole (ie. *curreg* is not equal to *regabv*), the perimeter of *tmp* is subtracted from that of *curreg* (since the outer perimeter is only required) and the perimeter *tmp* is incremented by (*I* - *lftend*).

C.3 Why Connected Component Analysis is Ideal for Analyzing Segmented Surface Froth Images

As segmented surface froth images consist of an interconnected network of extracted bubble boundaries (of pixel value 1), which give rise to isolated regions or blobs (of pixel value 0) that are representative of the individual bubbles, the images are well suited to connected component analysis. The method is efficient and quick and only those blobs i, where *b[i].color* == *0* (excluding the background blob) need to be considered when statistically characterising the froth structure. The background blob is excluded, because by zeroing the non-AOI, as illustrated in Figure C.12, those blobs truncated by the border of the AOI which is defined by the coordinates $\{(x_1,y_1);(x_2,y_2)\}$, are set to the background blob and are therefore of no interest.

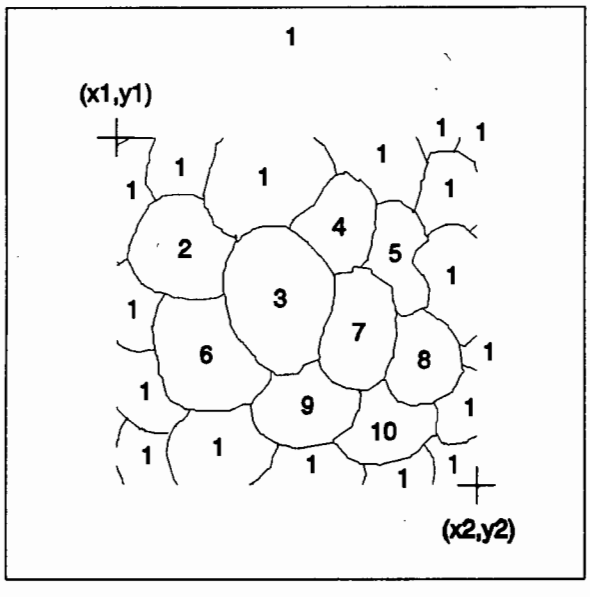


Figure C.12 : Blob identification for the AOI of a segmented surface froth image.

Appendix D

Photographs of the Surface Froth Images Processed

Figures D.1 to D.14 are the photographs of the raw surface froth structures, BUB1.CFI to BUB14.CFI, that were processed as part of this dissertation.

Images BUB1.CFI to BUB12.CFI are photographs of surface froth structures from three different cells, which, for ease of identification, are called cell A, cell B and cell C, where :

- BUB1.CFI to BUB4.CFI are from cell A.
- BUB5.CFI to BUB8.CFI are from cell B.
- BUB9.CFI to BUB12.CFI are from cell C.

The results of segmenting and characterising these images are presented in Chapters 5 and 6 for BUB1.CFI and Chapter 8, section 8.2 for BUB2.CFI to BUB12.CFI.

Images BUB13.CFI and BUB14.CFI, which, due to the incorrect lighting and camera configuration used to video the scene, represent suboptimal data and were merely used to test the reliability of the morphological segmentation technique discussed in Chapter 6.

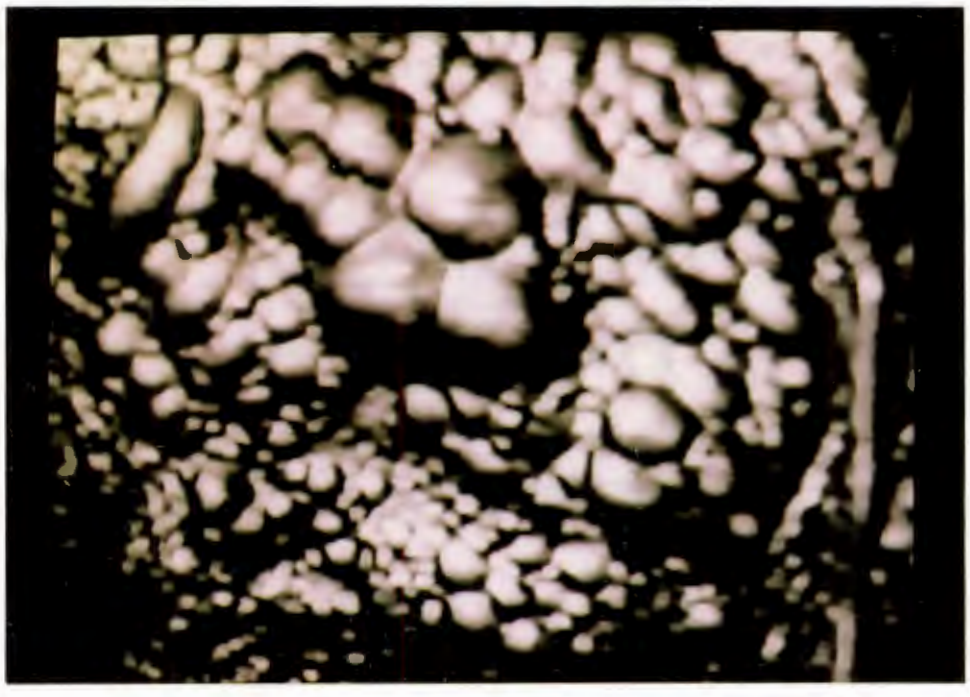


Figure D.1 : BUB1.CFI.

Figure D.2 : BUB2.CFI.

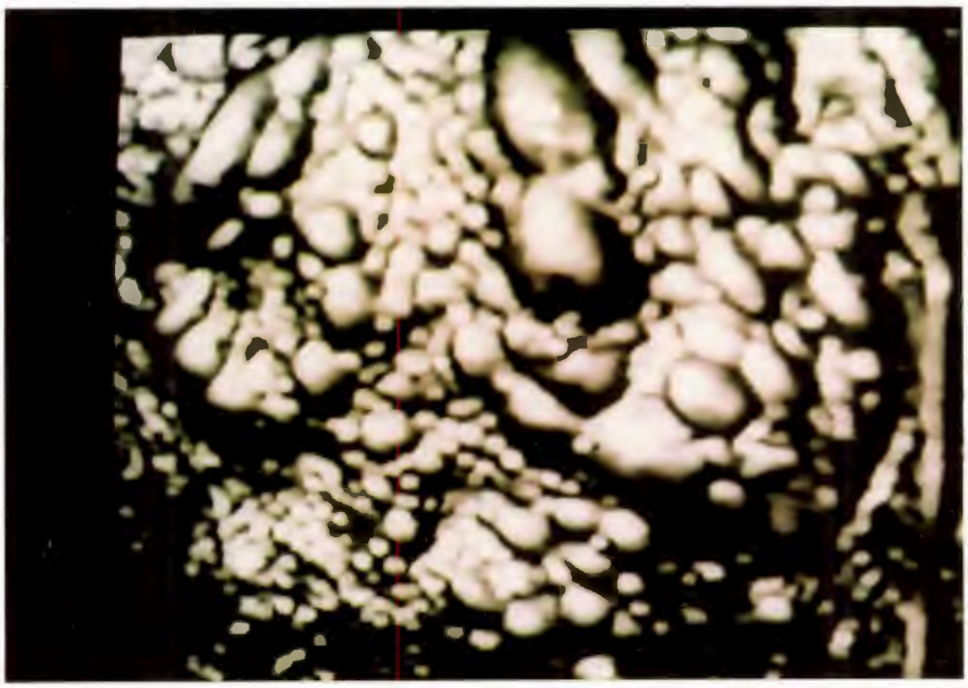


Figure D.3 : BUB3.CFI.

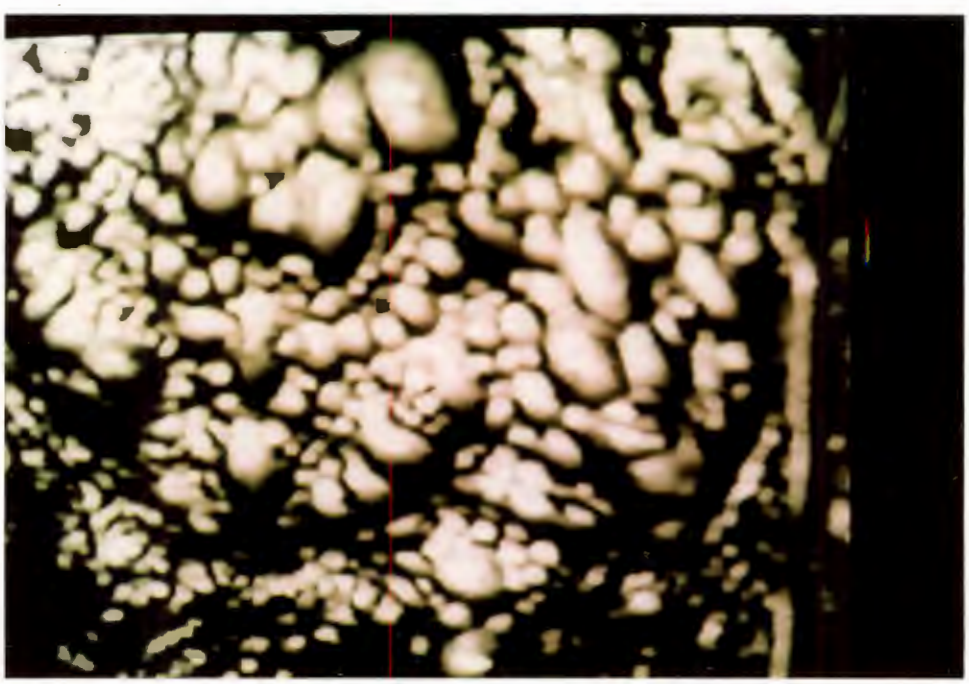


Figure D.4 : BUB4.CFI.

Figure D.5 : BUB5.CFI.

Figure D.6 : BUB6.CFI.

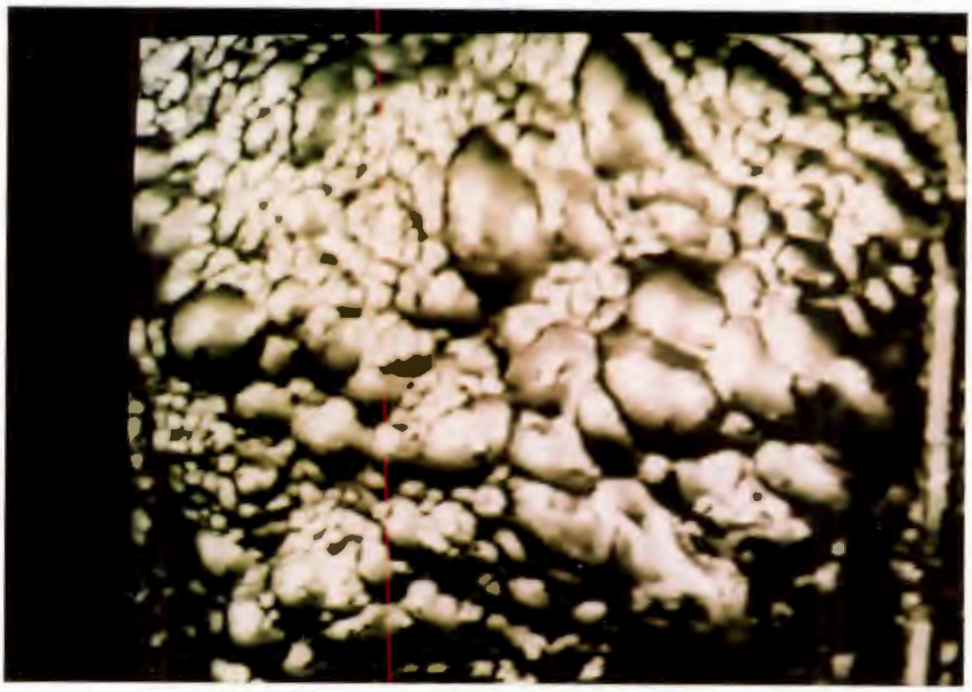


Figure D.7 : BUB7.CFI.

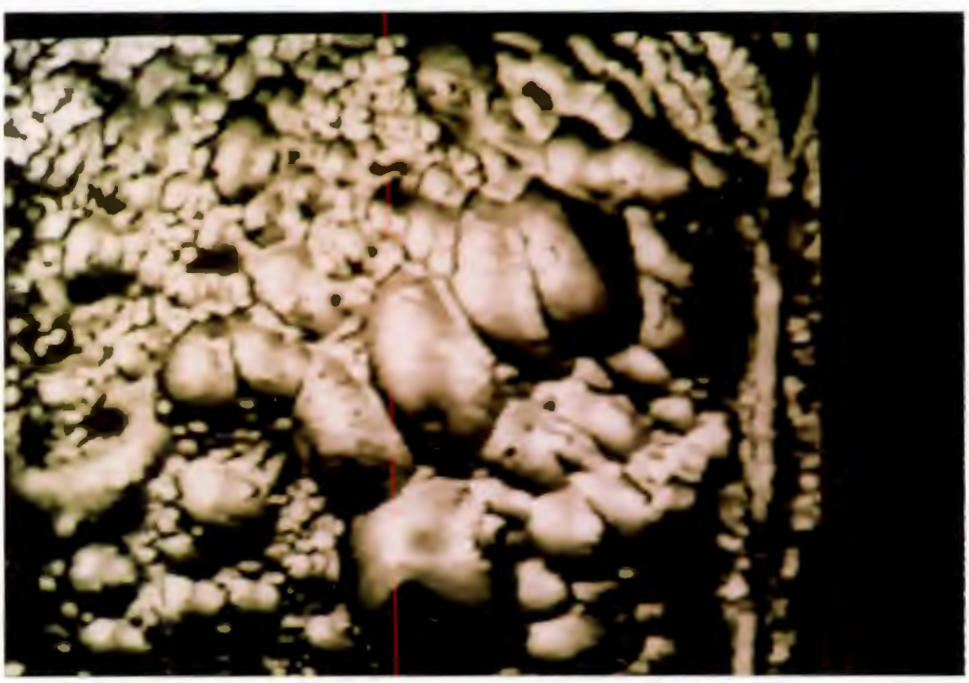


Figure D.8 : BUB8.CFI.

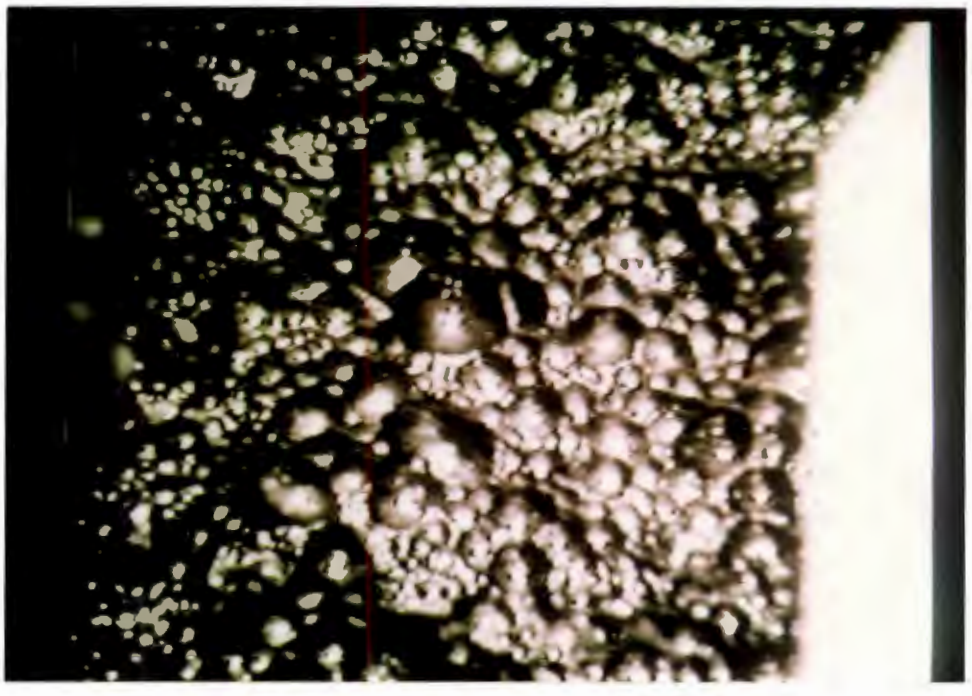


Figure D.9 : BUB9.CFI.

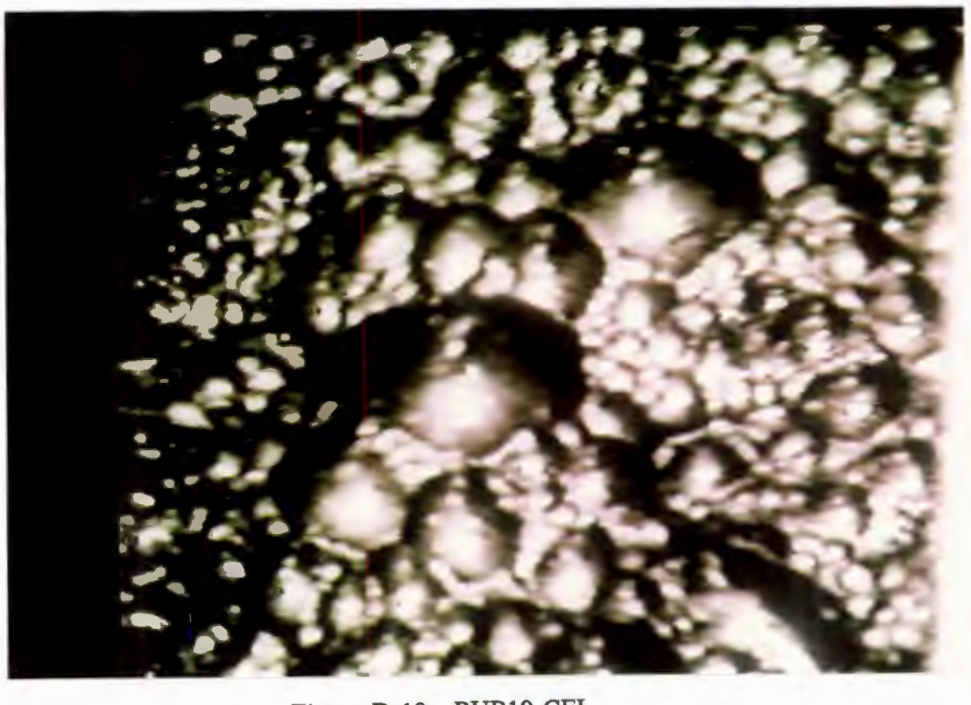


Figure D.10 : BUB10.CFI.

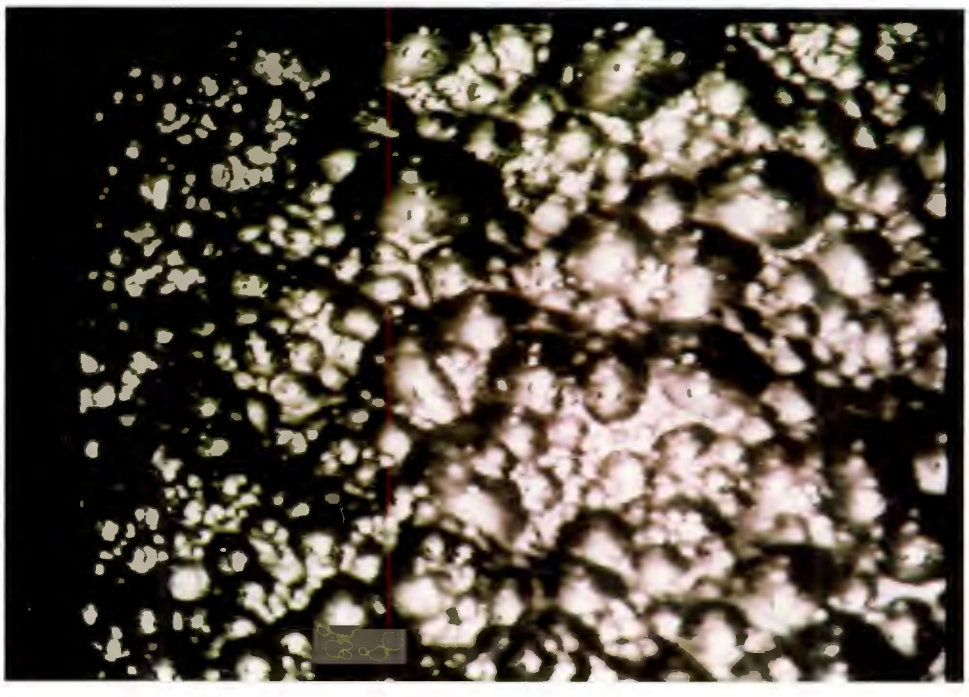


Figure D.11 : BUB11.CFI.

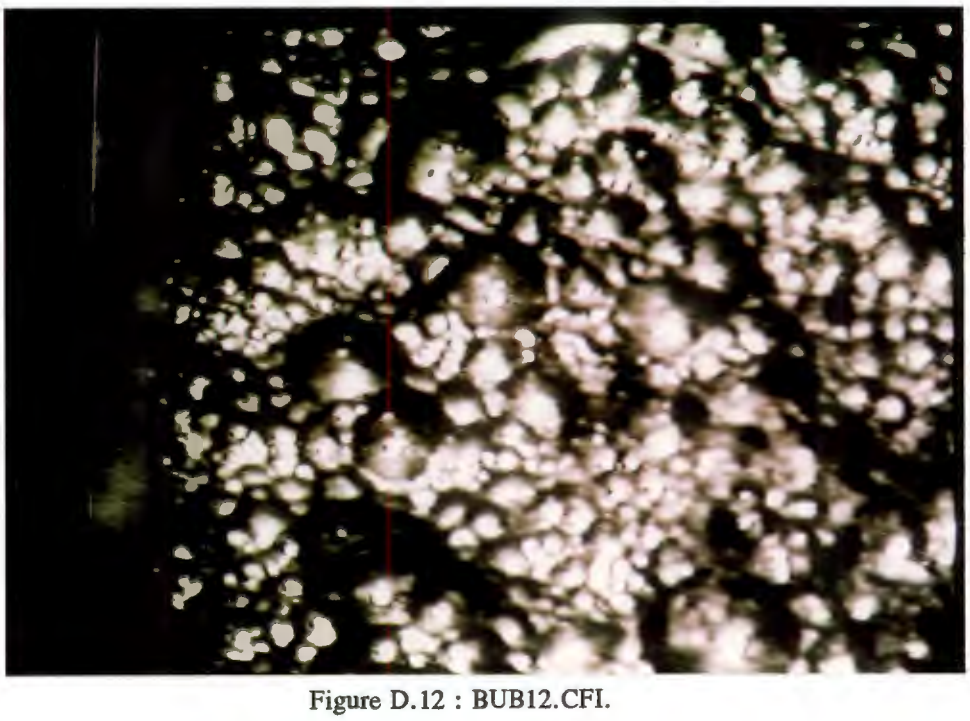


Figure D.12 : BUB12.CFI.

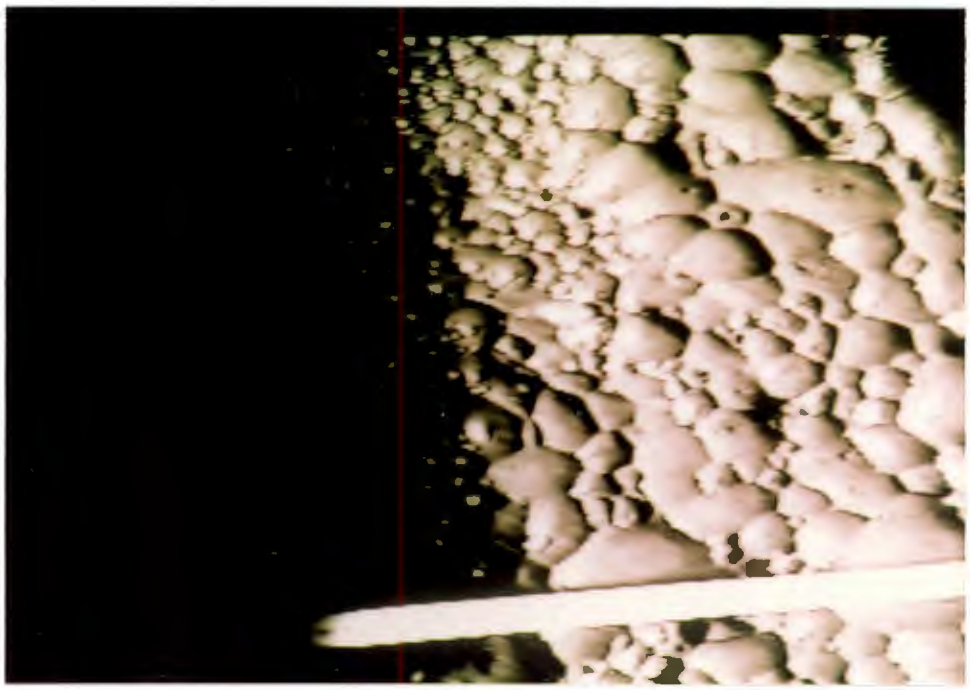


Figure D.13 : BUB13.CFI.

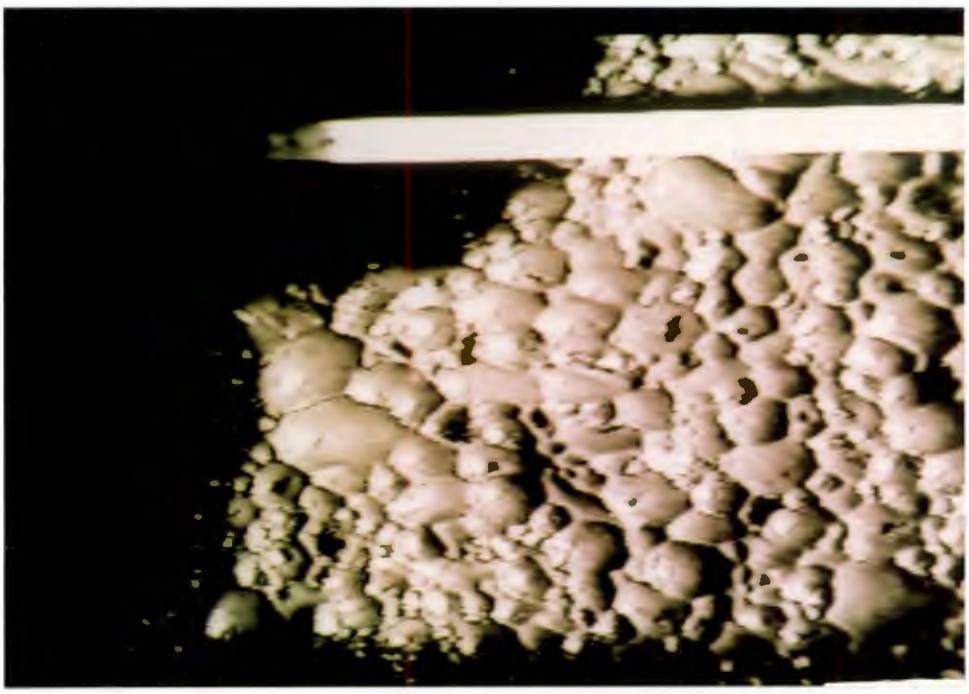


Figure D.14 : BUB14.CFI.

Appendix E

Tabulated Results for the Processed Images

This appendix contains the tabulated values for the results presented in Chapter 8, sections 8.2.1 to 8.2.11. The data for the individual images BUB2.CFI to BUB12.CFI, is given as follows :

- Segmentation efficiency and optimal SE values.
- Chi-square test for the size distributions obtained by manually and automatically segmenting the image.
- Chi-square test for the log normal fit to the size distribution obtained by manually segmenting the image.
- Chi-square test for the log normal fit to the size distribution obtained by automatically segmenting the image.
- Chi-square test for the circularity distributions obtained by manually and automatically segmenting the image.
- Chi-square test for the elipticity distributions obtained by manually and automatically segmenting the image.

As stated in Chapter 8, section 8.1, the distributions obtained by manually segmenting the images are referred to as the **expected** (E) distributions and those obtained by automatically segmenting the images with the respective optimal SE, are referred to as the **observed** (O) distributions.

BUB2.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(45,20);(420,421)\}$.

Size range : 25 pixels to 7000 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 188$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 180$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	22	11.70	1.00
4	60	31.91	2.73
5	105	55.85	4.77
6	138	73.40	6.27
7	158	84.04	7.18
8	163	86.70	7.41
9	182	96.81	8.27
10	180	95.74	8.18
11	180	95.74	8.18
12	181	96.28	8.23
13	175	93.09	7.95
14	178	94.68	8.09
15	170	90.43	7.73
16	159	84.57	7.23
17	151	80.32	6.86
18	151	80.32	6.86
19	145	77.13	6.59
20	141	75.00	6.41

Table E.1 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 7000 pixels.

Chosen number of bins : 40.

Bin width : 175 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	175	28	30	28.80	0.02
176	350	41	66	63.36	7.98
351	525	38	34	32.64	0.88
526	700	20	14	13.44	3.20
701	875	16	11	10.56	2.80
876	1050	6	8	7.68	0.37
1051	1225	10	4	3.84	9.88
1226	1750	11	11	10.56	0.02
1751	7000	10	10	9.60	0.02
		180	188	180.48	25.08

Table E.2 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 552.44$

$\sigma = 574.81$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	175	30	33.9	0.45
176	350	66	52.3	3.59
351	525	34	34.9	0.02
526	700	14	21.7	2.73
701	875	11	13.8	0.57
876	1050	8	9.0	0.11
1051	1225	4	6.0	0.67
1226	1575	6	7.1	0.17
1576	7000	15	9.3	3.49
		188	188.0	11.80

Table E.3 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 645.10$

$\sigma = 724.40$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	175	28	28.9	0.03
176	350	41	45.1	0.37
351	525	38	32.0	1.13
526	700	20	21.2	0.07
701	875	16	14.2	0.23
876	1050	6	9.8	1.47
1051	1225	10	6.9	1.39
1226	1575	7	8.6	0.30
1576	2100	8	6.4	0.40
2101	7000	6	7.1	0.17
		180	180.2	5.56

Table E.4 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 5.

Chosen number of bins : 20.

Bin width : 0.2.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.79	36	10	9.60	72.60
1.80	1.99	66	36	34.56	28.60
2.00	2.19	26	40	38.40	4.00
2.20	2.39	13	35	33.60	12.63
2.40	2.59	12	23	22.08	4.60
2.60	2.79	9	16	15.36	3.06
2.80	2.99	5	12	11.52	3.69
3.00	3.39	6	9	8.64	0.81
3.40	5.00	7	7	6.72	0.01
		180	188	180.48	130.00

Table E.5 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 4.

Chosen number of bins : 20.

Bin width : 0.15.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.14	12	23	22.08	4.60
1.15	1.29	42	22	21.12	20.64
1.30	1.44	30	37	35.52	0.86
1.45	1.59	34	29	27.84	1.36
1.60	1.74	15	20	19.20	0.92
1.75	1.89	20	12	11.52	6.24
1.90	2.04	7	15	14.40	3.80
2.05	2.19	7	8	7.68	0.06
2.20	2.34	4	4	3.84	0.01
2.35	4.00	9	18	17.28	3.97
		180	188	180.48	42.46

Table E.6 : χ^2 test for the expected and observed elipticity distributions.

BUB3.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(45,49);(413,388)\}$.

Size range : 25 pixels to 7000 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 198$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 166$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	20	10.10	1.00
4	67	33.84	3.35
5	97	48.99	4.85
6	123	62.12	6.15
7	135	68.18	6.75
8	149	75.25	7.45
9	160	80.81	8.00
10	164	82.83	8.20
11	166	83.84	8.30
12	167	84.34	8.35
13	159	80.30	7.95
14	158	79.80	7.90
15	147	74.24	7.35
16	144	72.73	7.20
17	138	69.70	6.90
18	135	68.18	6.75
19	127	64.14	6.35
20	124	62.63	6.20

Table E.7 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 7000 pixels.

Chosen number of bins : 40.

Bin width : 175 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	175	25	56	47.04	10.33
176	350	56	70	58.80	0.13
351	525	30	28	23.52	1.79
526	700	16	14	11.76	1.53
701	875	12	7	5.88	6.37
876	1575	22	16	13.44	5.45
1576	7000	5	7	5.88	0.13
		166	198	166.32	25.73

Table E.8 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 423.04$

$\sigma = 431.73$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	175	56	52.8	0.19
176	350	70	61.7	1.12
351	525	28	34.2	1.12
526	700	14	18.7	1.18
701	875	7	10.8	1.33
876	1050	8	6.5	0.35
1051	1400	4	6.7	1.09
1401	7000	11	6.5	3.12
		198	197.9	9.45

Table E.9 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 535.77$

$\sigma = 517.09$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	175	25	27.4	0.21
176	350	56	47.7	1.44
351	525	30	32.5	0.19
526	700	16	20.0	0.80
701	875	12	12.4	0.01
876	1050	9	7.9	0.15
1051	1225	5	5.2	0.01
1226	1575	8	5.9	0.75
1576	7000	5	6.9	0.52
		166	165.9	4.08

Table E.10 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 8.

Chosen number of bins : 40.

Bin width : 0.175.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.000	1.874	57	18	15.12	116.00
1.875	2.049	41	47	39.48	0.06
2.050	2.224	22	35	29.40	1.86
2.225	2.399	19	37	31.08	4.70
2.400	2.574	8	27	22.68	9.50
2.575	2.749	6	16	13.44	4.12
2.750	2.924	5	8	6.72	0.44
2.925	8.000	8	10	8.40	0.02
		166	198	166.32	136.70

Table E.11 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 4.

Chosen number of bins : 20.

Bin width : 0.15.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.14	12	15	12.60	0.03
1.15	1.29	39	39	32.76	1.19
1.30	1.44	28	41	34.44	1.20
1.45	1.59	30	28	23.52	1.79
1.60	1.74	20	24	20.16	0.00
1.75	1.89	11	20	16.80	2.00
1.90	2.04	8	6	5.04	1.73
2.05	2.19	6	6	5.04	0.18
2.20	2.34	4	7	5.88	0.60
2.35	4.00	8	12	10.08	0.43
		166	198	166.32	9.15

Table E.12 : χ^2 test for the expected and observed elipticity distributions.

BUB4.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(40,80);(449,409)\}$.

Size range : 25 pixels to 4000 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 191$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 180$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	25	13.09	1.00
4	84	43.98	3.36
5	137	71.73	5.48
6	158	82.72	6.32
7	164	85.86	6.56
8	172	90.05	6.88
9	180	94.24	7.20
10	180	94.24	7.20
11	173	90.58	6.92
12	168	87.96	6.72
13	163	85.34	6.52
14	161	84.29	6.44
15	162	84.82	6.48
16	158	82.72	6.32
17	148	77.49	5.92
18	144	75.39	5.76
19	142	74.34	5.68
20	136	71.20	5.44

Table E.13 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 4000 pixels.

Chosen number of bins : 40.

Bin width : 100 pixels.

Lower Limit	Upper Limit	O_j	E_j	χ^2
1	100	11	11.3	0.01
101	200	30	29.7	0.00
201	300	23	28.7	1.13
301	400	25	23.2	0.14
401	500	18	17.9	0.00
501	600	11	13.7	0.53
601	700	14	10.6	1.09
701	800	12	8.2	1.76
801	900	9	6.4	1.06
901	1000	5	5.1	0.00
1101	1200	6	7.4	0.26
1201	1500	6	6.7	0.07
1501	2000	6	5.5	0.05
2001	4000	4	5.6	0.46
		180	180.0	6.56

Table E.16 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 5.

Chosen number of bins : 20.

Bin width : 0.2.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.79	44	11	10.34	109.57
1.80	1.99	55	41	38.54	7.03
2.00	2.19	39	38	35.72	0.30
2.20	2.39	13	32	30.08	9.70
2.40	2.59	9	20	18.80	5.11
2.60	2.79	8	21	19.74	6.98
2.80	3.19	6	18	16.92	7.05
3.20	5.00	6	10	9.40	1.23
		180	191	179.54	146.97

Table E.17 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 4.

Chosen number of bins : 20.

Bin width : 0.15.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.14	13	17	15.98	0.56
1.15	1.29	31	24	22.56	3.16
1.30	1.44	43	32	30.08	5.55
1.45	1.59	25	26	24.44	0.01
1.60	1.74	25	25	23.50	0.10
1.75	1.89	8	16	15.04	3.30
1.90	2.04	12	10	9.40	0.72
2.05	2.19	7	11	10.34	1.08
2.20	2.34	7	7	6.58	0.03
2.35	4.00	9	23	21.62	7.37
		180	191	179.54	21.88

Table E.18 : χ^2 test for the expected and observed elipticity distributions.

BUB5.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(20,60);(420,430)\}$.

Size range : 25 pixels to 12000 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 236$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 239$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	29	12.29	1.00
4	84	35.59	2.90
5	128	54.24	4.41
6	181	76.69	6.24
7	192	81.36	6.62
8	224	94.92	7.72
9	240	101.69	8.27
10	239	101.27	8.24
11	250	105.93	8.62
12	237	100.42	8.17
13	234	99.15	8.07
14	229	97.03	7.90
15	216	91.53	7.45
16	212	89.83	7.31
17	205	86.86	7.07
18	205	86.86	7.07
19	191	80.93	6.59
20	185	78.39	6.38

Table E.19 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 12000 pixels.

Chosen number of bins : 100.

Bin width : 120 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	120	48	56	56.56	1.30
121	240	58	88	88.88	10.73
241	360	42	30	30.30	4.52
361	480	23	15	15.15	4.07
481	600	10	10	10.10	0.00
601	720	16	7	7.07	11.28
721	960	16	6	6.06	16.30
961	1080	8	6	6.06	0.62
1081	1920	9	7	7.07	0.53
1921	12000	9	11	11.11	0.40
		239	236	238.36	49.75

Table E.20 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 383.14$

$\sigma = 469.55$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	120	56	54.7	0.03
121	240	88	62.4	10.50
241	360	30	38.8	2.00
361	480	15	24.1	3.44
481	600	10	15.5	1.95
601	720	7	10.4	1.11
721	840	4	7.2	1.42
841	961	2	5.2	1.97
961	1200	9	6.6	0.87
1201	1560	4	5.1	0.24
1561	12000	11	6.1	3.94
		236	236.1	27.47

Table E.21 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 459.64$

$\sigma = 584.17$

Lower Limit	Upper Limit	O_j	E_j	χ^2
1	120	48	45.3	0.16
121	240	58	57.8	0.00
241	360	42	39.1	0.22
361	480	23	25.9	0.32
481	600	10	17.6	3.28
601	720	16	12.3	1.11
721	840	9	8.9	0.00
841	960	7	6.5	0.04
961	1200	8	8.7	0.06
1201	1440	2	5.2	1.97
1441	1920	7	5.6	0.35
1921	12000	9	6.1	1.38
		239	239.0	8.89

Table E.22 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 10.

Chosen number of bins : 25.

Bin width : 0.36.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.71	24	19	19.19	1.21
1.72	2.07	123	97	97.97	6.39
2.08	2.43	43	67	67.67	8.99
2.44	2.79	9	31	31.31	15.90
2.80	10.00	40	22	22.22	14.23
		239	236	238.36	46.72

Table E.23 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 6.

Chosen number of bins : 25.

Bin width : 0.2.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.19	19	20	20.20	0.07
1.20	1.39	59	52	52.52	0.80
1.40	1.59	38	49	49.49	2.67
1.60	1.79	41	32	32.32	2.33
1.80	1.99	25	20	20.20	1.14
2.00	2.19	25	24	24.24	0.02
2.20	2.39	17	14	14.14	0.58
2.40	2.59	7	9	9.09	0.48
2.60	6.00	8	16	16.16	4.12
		239	236	238.36	12.21

Table E.24 : χ^2 test for the expected and observed elipticity distributions.

BUB6.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(40,44);(428,450)\}$.

Size range : 25 pixels to 8000 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 279$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 288$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	62	22.22	1.00
4	114	40.86	1.84
5	185	66.31	2.98
6	225	80.65	3.63
7	251	89.96	4.05
8	264	94.62	4.26
9	282	101.08	4.55
10	288	103.23	4.65
11	278	99.64	4.48
12	263	94.27	4.24
13	267	95.70	4.31
14	262	93.91	4.23
15	253	90.68	4.08
16	248	88.89	4.00
17	233	83.51	3.76
18	228	81.72	3.68
19	222	79.57	3.58
20	218	78.14	3.52

Table E.25 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 8000 pixels.

Chosen number of bins : 50.

Bin width : 160 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	160	82	118	121.54	12.86
161	320	78	86	88.58	1.26
321	480	61	38	39.14	12.21
481	640	19	8	8.24	14.05
641	1120	35	10	10.30	59.23
1121	1280	5	5	5.15	0.00
1281	8000	8	14	14.42	2.86
		288	279	287.37	102.47

Table E.26 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 320.43$

$\sigma = 349.60$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	160	118	102.2	2.44
161	320	86	84.9	0.01
321	480	38	40.5	0.15
481	640	8	20.6	7.71
641	800	2	11.3	7.65
801	960	2	6.6	3.21
961	1280	11	6.7	2.76
1281	8000	14	6.2	9.81
		279	279.0	33.74

Table E.27 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 390.03$

$\sigma = 416.34$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	160	82	80.4	0.03
161	320	78	87.5	1.03
321	480	61	48.1	3.46
481	640	19	26.6	2.17
641	800	19	15.5	0.79
801	960	8	9.5	0.24
961	1120	8	6.1	0.59
1121	1440	7	6.7	0.01
1441	8000	6	7.7	0.38
		288	288.1	8.70

Table E.28 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 10.

Chosen number of bins : 40.

Bin width : 0.225.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.000	1.674	18	12	12.36	2.57
1.675	1.899	81	63	64.89	4.00
1.900	2.124	85	69	71.07	2.73
2.125	2.349	40	59	60.77	7.10
2.350	2.574	19	28	28.84	3.36
2.575	2.799	8	22	22.66	9.48
2.800	3.024	8	12	12.36	1.54
3.025	3.249	8	6	6.18	0.54
3.250	10.000	21	8	8.24	19.76
		288	279	287.37	51.08

Table E.29 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 6.

Chosen number of bins : 25.

Bin width : 0.2.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.19	28	34	35.02	1.41
1.20	1.39	61	49	50.47	2.20
1.40	1.59	49	51	52.53	0.24
1.60	1.79	50	40	41.20	1.88
1.80	1.99	39	33	33.99	0.74
2.00	2.19	29	20	20.60	3.43
2.20	2.39	17	24	24.72	2.41
2.40	2.59	8	9	9.27	0.17
2.60	6.00	7	19	19.57	8.07
		288	279	287.39	20.55

Table E.30 : χ^2 test for the expected and observed elipticity distributions.

BUB7.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(60,60);(448,428)\}$.

Size range : 25 pixels to 6000 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 240$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 240$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	40	16.67	1.00
4	105	43.75	2.63
5	160	66.67	4.00
6	191	79.58	4.78
7	208	86.67	5.20
8	231	96.25	5.78
9	230	95.83	5.75
10	231	96.25	5.78
11	241	100.42	6.03
12	240	100.00	6.00
13	236	98.33	5.90
14	226	94.17	5.65
15	210	87.50	5.25
16	214	89.17	5.35
17	207	86.25	5.18
18	197	82.08	4.93
19	204	85.00	5.10
20	187	77.92	4.68

Table E.31 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 6000 pixels.

Chosen number of bins : 50.

Bin width : 120 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1	120	32	64	64	16.00
121	240	65	80	80	2.81
241	360	51	29	29	16.69
361	480	26	18	18	3.56
481	600	19	14	14	1.79
601	720	14	8	8	4.50
721	960	14	6	6	10.67
961	1440	8	9	9	0.11
1441	6000	11	12	12	0.08
		240	240	240	56.21

Table E.32 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 364.53$

$\sigma = 436.58$

Lower Limit	Upper Limit	O_j	E_j	χ^2
1	120	64	57.6	0.71
121	240	80	65.1	3.41
241	360	29	39.7	2.88
361	480	18	24.2	1.59
481	600	14	15.3	0.11
601	720	8	10.1	0.44
721	840	2	6.9	3.48
841	1080	5	8.4	1.38
1081	1440	8	6.1	0.59
1441	6000	12	6.5	4.65
		240	239.9	19.24

Table E.33 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 425.05$

$\sigma = 437.26$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	120	32	34.5	0.18
121	240	65	62.0	0.15
241	360	51	45.3	0.72
361	480	26	29.8	0.48
481	600	19	19.7	0.02
601	720	14	13.2	0.05
721	840	8	9.1	0.13
841	960	6	6.4	0.03
961	1200	7	8.0	0.13
1201	1560	1	5.9	4.07
1561	6000	11	6.1	3.94
		240	240.0	9.90

Table E.34 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 6.

Chosen number of bins : 25.

Bin width : 0.2.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.79	76	33	33	56.03
1.80	1.99	50	62	62	2.32
2.00	2.19	45	40	40	0.63
2.20	2.39	26	46	46	8.70
2.40	2.59	13	22	22	3.68
2.60	2.79	9	13	13	1.23
2.80	3.39	6	14	14	4.57
3.40	6.00	15	10	10	2.50
		240	240	240	79.66

Table E.35 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 5.

Chosen number of bins : 25.

Bin width : 0.16.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.15	29	15	15	13.07
1.16	1.31	54	44	44	2.27
1.32	1.47	37	44	44	1.11
1.48	1.63	36	27	27	3.00
1.64	1.79	15	26	26	4.65
1.80	1.95	28	24	24	0.67
1.96	2.11	12	11	11	0.09
2.12	2.27	6	9	9	1.00
2.28	2.43	9	10	10	0.10
2.44	2.59	7	11	11	1.45
2.60	5.00	7	19	19	7.58
		240	240	240	35.28

Table E.36 : χ^2 test for the expected and observed elipticity distributions.

BUB8.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(40,45);(445,455)\}$.

Size range : 25 pixels to 8000 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 284$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 283$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	52	18.31	1.00
4	128	45.07	2.46
5	196	69.01	3.77
6	235	82.75	4.52
7	254	89.44	4.88
8	261	91.90	5.02
9	274	96.48	5.27
10	277	97.54	5.33
11	283	99.65	5.44
12	293	103.17	5.63
13	272	95.77	5.23
14	270	95.07	5.19
15	252	88.73	4.85
16	251	88.38	4.83
17	239	84.15	4.60
18	243	85.56	4.67
19	242	85.21	4.65
20	233	82.04	4.48

Table E.37 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 8000 pixels.

Chosen number of bins : 50.

Bin width : 160 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	160	87	144	144	22.56
161	320	85	75	75	1.33
321	480	43	27	27	9.48
481	640	26	9	9	32.11
641	800	12	8	8	2.00
801	1920	24	8	8	32.00
1921	8000	6	13	13	3.77
		283	284	284	103.25

Table E.38 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 300.34$

$\sigma = 348.56$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	160	144	117.3	6.08
161	320	75	82.1	0.61
321	480	27	37.4	2.89
481	640	9	18.8	5.11
641	800	8	10.3	0.51
801	960	4	6.0	0.67
961	1280	2	6.1	2.76
1281	8000	15	6.0	13.50
		284	284.0	32.13

Table E.39 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 389.82$

$\sigma = 445.23$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	160	87	85.6	0.02
161	320	85	82.9	0.05
321	480	43	44.7	0.06
481	640	26	24.9	0.05
641	800	12	14.7	0.50
801	960	7	9.1	0.48
961	1120	4	5.9	0.61
1121	1440	9	6.8	0.71
1441	8000	10	8.4	0.30
		283	283.0	2.78

Table E.40 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 8.

Chosen number of bins : 25.

Bin width : 0.28.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.83	86	91	91	0.27
1.84	2.11	105	97	97	0.66
2.12	2.39	37	55	55	5.89
2.40	2.67	21	28	28	1.75
2.68	3.23	14	7	7	7.00
3.24	8.00	20	6	6	32.67
		283	284	284	48.24

Table E.41 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 6.

Chosen number of bins : 25.

Bin width : 0.2.

Lower Limit	Upper Limit	O_j	e_j	E_j	χ^2
1.00	1.19	29	29	29	0.00
1.20	1.39	61	74	74	2.28
1.40	1.59	54	58	58	0.28
1.60	1.79	49	43	43	0.84
1.80	1.99	37	25	25	5.76
2.00	2.19	22	24	24	0.17
2.20	2.39	10	7	7	1.29
2.40	2.59	9	10	10	0.10
2.60	6.00	12	14	14	0.29
		283	284	284	11.01

Table E.42 : χ^2 test for the expected and observed elipticity distributions.

BUB9.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(80,60);(410,410)\}$.

Size range : 10 pixels to 4000 pixels.

Expected number of bubbles in specified size range, for processed AOI : $N_E = 408$.

Observed number of bubbles in specified size range, for processed AOI : $N_O = 417$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	110	26.96	1.00
4	242	59.31	2.20
5	356	87.25	3.24
6	383	93.87	3.48
7	424	103.92	3.85
8	417	102.21	3.79
9	409	100.25	3.72
10	396	97.06	3.60
11	377	92.40	3.43
12	360	88.24	3.27
13	315	77.21	2.86
14	313	76.72	2.85
15	290	71.08	2.64
16	259	63.48	2.35
17	254	62.25	2.31
18	242	59.31	2.20
19	218	53.43	1.98
20	221	54.17	2.01

Table E.43 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 4000 pixels.

Chosen number of bins : 60.

Bin width : 66.67 pixels.

BUB10.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(100,60);(440,425)\}$.

Size range : 10 pixels to 13200 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 237$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 231$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	16	6.75	1.00
4	50	21.10	3.13
5	122	51.48	7.63
6	170	71.73	10.63
7	175	73.84	10.94
8	210	88.61	13.13
9	219	92.41	13.69
10	232	97.89	14.50
11	231	97.47	14.44
12	225	94.94	14.07
13	214	90.30	13.38
14	210	88.61	13.13
15	204	86.08	12.75
16	202	85.23	12.63
17	178	75.11	11.13
18	174	73.42	10.88
19	167	70.46	10.44
20	151	63.71	9.44

Table E.49 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 13200 pixels.

Chosen number of bins : 110.

Bin width : 120 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	120	45	53	51.41	0.80
121	240	61	98	95.06	12.20
241	360	46	45	43.65	0.13
361	480	28	12	11.64	22.99
481	720	28	8	7.76	52.79
721	1080	8	6	5.82	0.82
1081	1440	6	6	7.76	0.40
1441	13200	9	6	6.79	0.72
		231	237	229.89	90.85

Table E.50 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 316.42$

$\sigma = 340.18$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	120	53	59.7	0.75
121	240	98	70.3	10.91
241	360	45	40.8	0.43
361	480	12	23.4	5.55
481	600	6	14.0	4.57
601	720	2	8.8	5.25
721	1080	6	12.2	3.15
1081	13200	15	7.8	6.65
		237	237.0	37.26

Table E.51 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 388.83$

$\sigma = 458.81$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	120	45	49.5	0.41
121	240	61	61.4	0.00
241	360	46	39.2	1.18
361	480	28	24.5	0.50
481	600	18	15.8	0.31
601	720	10	10.6	0.03
721	840	4	7.3	1.49
841	960	3	5.2	0.93
961	1200	3	6.6	1.96
1201	1560	5	5.0	0.00
1561	13200	8	5.8	0.83
		231	230.9	7.64

Table E.52 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 10.

Chosen number of bins : 25.

Bin width : 0.36.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.71	19	22	21.34	0.26
1.72	2.07	114	111	107.67	0.37
2.08	2.43	48	71	68.87	6.32
2.44	2.79	21	23	22.31	0.08
2.80	10.00	29	10	9.70	38.40
		231	237	229.89	45.43

Table E.53 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 6.

Chosen number of bins : 25.

Bin width : 0.2.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.19	29	44	42.68	4.38
1.20	1.39	49	54	52.38	0.22
1.40	1.59	52	51	49.47	0.13
1.60	1.79	39	34	32.98	1.10
1.80	1.99	28	25	24.25	0.58
2.00	2.19	11	14	13.58	0.49
2.20	2.39	12	7	6.79	4.00
2.40	6.00	11	8	7.76	1.35
		231	237	229.89	12.25

Table E.54 : χ^2 test for the expected and observed elipticity distributions.

BUB11.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(60,60);(400,430)\}$.

Size range : 10 pixels to 5000 pixels.

Expected number of bubbles in specified size range, for the processed AOI : $N_E = 370$.

Observed number of bubbles in specified size range, for the processed AOI : $N_O = 399$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	66	17.84	1.00
4	187	50.54	2.83
5	300	81.08	4.55
6	352	95.14	5.33
7	382	103.24	5.79
8	399	107.84	6.05
9	370	100.00	5.61
10	368	99.46	5.58
11	366	98.92	5.55
12	341	92.16	5.17
13	318	85.95	4.82
14	311	84.05	4.71
15	295	79.73	4.47
16	284	76.76	4.30
17	259	70.00	3.92
18	252	68.11	3.82
19	237	64.05	3.59
20	215	58.11	3.26

Table E.55 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 5000 pixels.

Chosen number of bins : 50.

Bin width : 100 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	100	124	136	146.88	3.56
101	200	118	104	112.32	0.29
201	300	66	58	62.64	0.18
301	400	28	27	29.16	0.05
401	500	21	10	10.80	9.63
501	600	14	8	8.64	3.33
601	800	13	7	7.56	3.91
801	1100	8	11	11.88	1.27
1101	5000	7	9	9.72	0.76
		399	370	399.60	22.98

Table E.56 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 225.04$

$\sigma = 242.32$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	100	136	116.0	3.45
101	200	104	113.2	0.75
201	300	58	58.7	0.01
301	400	27	31.4	0.62
401	500	10	17.8	3.42
501	600	8	10.7	0.68
601	700	3	6.7	2.04
701	900	12	7.3	3.03
901	5000	12	8.1	1.88
		370	369.9	15.26

Table E.57 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 241.04$

$\sigma = 275.69$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	100	124	122.4	0.02
101	200	118	116.9	0.01
201	300	66	62.6	0.18
301	400	28	34.7	1.29
401	500	21	20.5	0.01
501	600	14	12.7	0.13
601	700	6	8.2	0.59
701	800	7	5.5	0.41
801	1000	7	6.5	0.04
1001	5000	8	8.8	0.07
		399	398.8	2.75

Table E.58 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 10.

Chosen number of bins : 25.

Bin width : 0.36.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.71	68	48	51.84	5.04
1.72	2.07	195	189	204.12	1.12
2.08	2.43	62	102	110.16	21.05
2.44	2.79	28	21	22.68	1.25
2.80	10.00	46	10	10.80	114.73
		399	370	399.60	143.19

Table E.59 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 6.

Chosen number of bins : 25.

Bin width : 0.2.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.19	56	65	70.20	2.87
1.20	1.39	98	101	109.08	1.13
1.40	1.59	88	77	83.16	0.28
1.60	1.79	63	58	62.64	0.00
1.80	1.99	29	28	30.24	0.05
2.00	2.39	27	16	17.28	6.12
2.40	2.59	22	12	12.96	6.31
2.60	6.00	16	13	14.04	0.27
		399	370	399.60	17.03

Table E.60 : χ^2 test for the expected and observed elipticity distributions.

BUB12.CFI

Processed area of interest : $\{(x_1,y_1);(x_2,y_2)\} = \{(80,60);(440,435)\}$.

Size range : 10 pixels to 4000 pixels.

Expected number of bubbles in specified size range, for processed AOI : $N_E = 332$.

Observed number of bubbles in specified size range, for processed AOI : $N_O = 359$.

Segmentation Efficiency and Optimal SE Values.

SE Radius (pixels)	Blobs Detected	Efficiency (%)	Optimal SE
3	44	13.25	1.00
4	126	37.95	2.86
5	240	72.29	5.45
6	306	92.17	6.95
7	341	102.71	7.75
8	352	106.02	8.00
9	359	108.13	8.16
10	359	108.13	8.16
11	368	110.84	8.36
12	345	103.92	7.84
13	319	96.08	7.25
14	310	93.37	7.05
15	297	89.46	6.75
16	282	84.94	6.41
17	265	79.82	6.02
18	261	78.61	5.93
19	238	71.69	5.41
20	242	72.89	5.50

Table E.61 : Bubble detection as a function of SE radius, with calculated efficiency (η) and optimal SE (γ) values.

Size Distributions.

Histogram range : 1 to 4000 pixels.

Chosen number of bins : 50.

Bin width : 80 pixels.

Goodness-of-fit of the expected and observed distributions

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1	80	43	56	60.48	5.05
81	160	96	98	105.84	0.91
161	240	60	68	73.44	2.46
241	320	55	37	39.96	5.66
321	400	31	15	16.20	13.52
401	480	23	11	11.88	10.41
481	560	15	12	12.96	0.32
561	640	12	7	7.56	2.61
641	960	13	17	18.36	1.56
961	4000	11	11	11.88	0.07
		359	332	358.56	42.57

Table E.62 : χ^2 test for the expected and observed size distributions.

Modelling the expected size distribution with a log normal distribution

Best fit parameters : $\mu = 266.26$

$\sigma = 279.24$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	80	56	55.5	0.00
81	160	98	89.3	0.85
161	240	68	61.6	0.66
241	320	37	39.3	0.13
321	400	15	25.4	4.26
401	480	11	16.9	2.06
481	560	12	11.5	0.02
561	640	7	8.0	0.13
641	720	8	5.7	0.93
721	880	5	7.3	0.72
881	1120	6	5.5	0.05
1121	4000	9	6.0	1.50
		332	332.0	11.31

Table E.63 : χ^2 test for the log normal fit to the expected size distribution.

Modelling the observed size distribution with a log normal distribution

Best fit parameters : $\mu = 284.03$

$\sigma = 274.65$

Lower Limit	Upper Limit	O_i	E_i	χ^2
1	80	43	44.7	0.06
81	160	96	92.5	0.13
161	240	60	70.7	1.62
241	320	55	47.0	1.36
321	400	31	30.9	0.00
401	480	23	20.7	0.26
481	560	15	14.1	0.06
561	640	12	9.8	0.49
641	720	3	7.0	2.29
721	880	8	5.0	1.80
881	1120	2	6.5	3.12
1121	4000	11	10.2	0.06
		359	359.1	11.25

Table E.64 : χ^2 test for the log normal fit to the observed size distribution.

Circularity Distributions.

Histogram range : 1 to 7.

Chosen number of bins : 25.

Bin width : 0.24.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.71	46	34	36.72	2.35
1.72	1.95	134	120	129.60	0.15
1.96	2.19	89	88	95.04	0.38
2.20	2.43	26	56	60.48	19.66
2.44	2.67	15	22	23.76	3.23
2.68	2.91	14	6	6.48	8.73
2.92	7.00	35	6	6.48	125.52
		359	332	358.56	160.02

Table E.65 : χ^2 test for the expected and observed circularity distributions.

Elipticity Distributions.

Histogram range : 1 to 7.

Chosen number of bins : 25.

Bin width : 0.24.

Lower Limit	Upper Limit	O_i	e_i	E_i	χ^2
1.00	1.23	61	72	77.76	3.61
1.24	1.47	107	109	117.72	0.98
1.48	1.71	85	75	81.00	0.20
1.72	1.95	50	37	39.96	2.52
1.96	2.19	29	14	15.12	12.74
2.20	2.43	11	12	12.96	0.30
2.44	7.00	16	13	14.04	0.27
		359	332	358.56	20.62

Table E.66 : χ^2 test for the expected and observed elipticity distributions.

Appendix F

Software Listings

The following C source code is listed :

Main Programs

- *frothaut.c* : this program automatically segments and analyses images of surface froth structures, using a discrete spherical structuring element of given radius.
- *frothman.c* : this program thresholds and analyses manually segmented surface froth images.

C Modules

- definitions module : this consists of the header file *imagedef.h*.
- I/O module : this consists of the header file *imageio.h* and the source file *imageio.c*, which contains functions for reading and writing CFI files.
- image utilities module : this consists of the header file *imagutil.h* and the source file *imagutil.c*, which contains some fundamental routines that were necessary.
- filter module : this consists of the header file *filter.h* and the source file *filter.c*, which contains functions for low-pass and median filtering.
- morphological processing module : this consists of the header file *morph.h* and the source file *morph.c*, which contains functions that implement binary and grey level dilations, erosions, closings and openings.
- thinning module : this consists of the header file *thin.h* and the source file *thin.c*, which consists of functions for thinning thresholded surface froth images.
- image analysis module : this consists of the header file *analysis.h* and the source file *analysis.c*, which contains functions for performing connected component analysis of the segmented and thinned surface froth images. As discussed in Appendix C, part of this module consists of previously written code that was modified for the purposes of this dissertation.

Main Programs

```
/*****
FILE : frothaut.c

DESCRIPTION : Program that performs automatic analysis of surface froth structures.

EXECUTION   : frothaut <filein> <fileout> <resultfile> <radius> <minarea> <maxarea> <x1> <y1> <x2> <y2>

              Where :- <filein> is the raw CFI file to be processed
                       <fileout> is the processed image
                       <resultfile> is a text file containing analysis results
                       <radius> is the radius of the discrete spherical structuring element (DSSE)
                       <minarea> is the smallest blob size of interest
                       <maxarea> is the largest blob size of interest
                       <x1> <y1> top left-hand corner of the area of interest
                       <x2> <y2> bottom right-hand corner of the area of interest

COMPILATION : cc -o frothaut frothaut.c imageio.c imagutil.c filter.c morph.c thin.c analysis.c -lm
*****/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "imagedef.h"
#include "imageio.h"
#include "imagutil.h"
#include "filter.h"
#include "morph.h"
#include "thin.h"
#include "analysis.h"

void title();

void main(argc, argv)
int argc;
char *argv[];
{
  BYTE origim[IMAGESIZE][IMAGESIZE];                    /* Original image          */
  BYTE filtim[IMAGESIZE][IMAGESIZE];                    /* Filtered image          */
  BYTE segim[IMAGESIZE][IMAGESIZE];                     /* Segmented image         */
  BYTE temp1[IMAGESIZE][IMAGESIZE];                     /* Temporary image #1      */
  BYTE temp2[IMAGESIZE][IMAGESIZE];                     /* Temporary image #2      */
  char *filein, *fileout, *resultfile;
  int fileflag, radius, x1, y1, x2, y2;
  unsigned long minarea, maxarea;
  char **se;                                            /* Structuring element (SE) */

  if (argc!=11)
  {
    printf("Invalid number of parameters. \n");
    printf("USAGE : frothaut <filein> <fileout> <resultfile> <radius> <minarea> <maxarea> <x1> <y1> <x2>
          <y2>\n");
    printf("WHERE : <filein> raw input image file\n");
    printf("        <fileout> segmented and thresholded output image file with blob centroids marked\n");
    printf("        <resultfile> is a text file containing the analysis results\n");
    printf("        <radius> is the radius of the structuring element to be used\n");
    printf("        <minarea> is the smallest blob size that is of interest\n");
    printf("        <maxarea> is the largest blob size that is of interest\n");
    printf("        <x1> <y1> are the top left-hand coordinates of the area of interest\n");
    printf("        <x2> <y2> are the bottom right-hand coordinates of the area of interest\n");
    exit(0);
  }
  else
  {
    ++argv;
    filein = *argv++;
    fileout = *argv++;
    resultfile = *argv++;
    radius = atoi(*argv++);
    minarea = (unsigned long) atoi(*argv++);
    maxarea = (unsigned long) atoi(*argv++);
    x1 = atoi(*argv++);
    y1 = atoi(*argv++);
    x2 = atoi(*argv++);
```

```
    y2 = atoi(*argv);

    if (!INRANGE(1,radius,20))                          /* Valid radius ?          */
    {
      printf("ERROR : invalid <radius> value.\n");
      exit(0);
    }

    if (!INRANGE(1,minarea,maxarea))                    /* Valid minarea ?         */
    {
      printf("ERROR : invalid <minarea> value.\n");
      exit(0);
    }

    if (!INRANGE(minarea,maxarea,INFINITY))             /* Valid maxarea ?         */
    {
      printf("ERROR : invalid <maxarea> value.\n");
      exit(0);
    }

    if (!(INRANGE(1,x1,x2) && INRANGE(x1,x2,510)) || !(INRANGE(1,y1,y2) && INRANGE(y1,y2,510)))
    {
      printf("ERROR : invalid area of interest.\n");
      exit(0);
    }
  }

  title();

  if ((fileflag = readimage(origim,filein)) < 0)
  {
    if (fileflag == -1)
    {
      printf("\nError opening %s\n",filein);
      exit(0);
    }
    else if (fileflag == -2)
    {
      printf("\nError reading %s \n",filein);
      exit(0);
    }
  }

  printf("\nStart : ");
  system("date");                                       /* Start of processing     */

#if AVERAGE
  average(origim,filtim);                               /* Low-pass filter the raw image */
#endif

#if MEDIAN
  median(origim,filtim);                                /* or median filter the raw image*/
#endif

  se = declare_spherical_se(radius);                    /* Declare the DSSE        */
  gs_close(filtim,segim,se,radius);                     /* "Rolling Ball" Algorithm */
  remove_spherical_se(se,radius);

  subtract(segim,filtim,segim);                         /* Determine G_c(x,y)      */

  threshold(segim,segim,1);                             /* Extract bubble boundaries */

  fillborder(segim,2*radius,0);                         /* Clean up the image border */

  b_open(segim,temp1,RECT2x1);                          /* Image cleaning          */
  b_open(temp1,segim,RECT1x2);
  b_close(segim,temp2,PLUS3x3);

  skeletonize(temp2);                                   /* Thin the image          */

  image_analysis(temp2,minarea,maxarea,x1,y1,x2,y2,resultfile);   /* Characterise the image */

  printf("\nFinish : ");
  system("date");                                       /* End of processing       */
```

```
  binaryto255(temp2,temp2);                          /* Convert image for display purposes     */

  if ((fileflag = writeimage(temp2,fileout)) < 0)
  {
    if (fileflag == -1)
    {
      printf("\nError opening %s\n",fileout);
      exit(0);
    }
    else if (fileflag == -2)
    {
      printf("\nError writing %s \n",fileout);
      exit(0);
    }
  }
}

/*****
Function : title()

Description : Displays the program title.

Argument list : void

Return value : void
*****/
void title()
{
  system("clear");

  printf("_____________________________________________________");
  printf("\n                    FROTHAUT                      \n\n");
  printf("   Automatic processing of surface froth images.    \n");
  printf("_____________________________________________________");
  printf("\n");
}
```

```
/*****
FILE : frothman.c

DESCRIPTION : Program that analyses manually segmented surface froth images.

EXECUTION   : frothman <filein> <fileout> <resultfile> <minarea> <maxarea> <x1> <y1> <x2> <y2>

              Where :- <filein> is the manually segmented image
                       <fileout> is the processed image
                       <resultfile> is a text file containing analysis results
                       <minarea> is the smallest blob size of interest
                       <maxarea> is the largest blob size of interest
                       <x1> <y1> top left-hand corner of the area of interest
                       <x2> <y2> bottom right-hand corner of the area of interest

COMPILATION : cc -o frothman frothman.c imageio.c imagutil.c analysis.c -lm
*****/
#include <stdio.h>
#include <stdlib.h>
#include "imagedef.h"
#include "imageio.h"
#include "imagutil.h"
#include "analysis.h"

void title();

void main(argc, argv)
int argc;
char *argv[];
{
  BYTE image[IMAGESIZE][IMAGESIZE];                          /* Manually segmented image      */
  char *filein, *fileout, *resultfile;
  unsigned long minarea, maxarea;
  int fileflag, x1, y1, x2, y2;

  if (argc!=10)
  {
    printf("Invalid number of parameters. \n");
    printf("USAGE : frothman <filein> <fileout> <resultfile> <minarea> <maxarea> <x1> <y1> <x2> <y2>\n");
    printf("WHERE : <filein> raw input image file\n");
    printf("        <fileout> thresholded output image file with blob centroids marked\n");
    printf("        <resultfile> is a text file containing the analysis results\n");
    printf("        <minarea> is the smallest blob size that is of interest\n");
    printf("        <maxarea> is the largest blob size that is of interest\n");
    printf("        <x1> <y1> are the top left-hand coordinates of the area of interest\n");
    printf("        <x2> <y2> are the bottom right-hand coordinates of the area of interest\n");
    exit(0);
  }
  else
  {
    ++argv;
    filein = *argv++;
    fileout = *argv++;
    resultfile = *argv++;
    minarea = (unsigned long) atoi(*argv++);
    maxarea = (unsigned long) atoi(*argv++);
    x1 = atoi(*argv++);
    y1 = atoi(*argv++);
    x2 = atoi(*argv++);
    y2 = atoi(*argv);

    if (!INRANGE(1,minarea,maxarea))                         /* Valid minarea ?               */
    {
      printf("ERROR : invalid <minarea> value.\n");
      exit(0);
    }

    if (!INRANGE(minarea,maxarea,INFINITY))                  /* Valid maxarea ?               */
    {
      printf("ERROR : invalid <maxarea> value.\n");
      exit(0);
    }
```

```
    if (!(INRANGE(1,x1,x2) && INRANGE(x1,x2,510)) || !(INRANGE(1,y1,y2) && INRANGE(y1,y2,510)))
    {
      printf("ERROR : invalid area of interest.\n");
      exit(0);
    }
  }

  title();

  if ((fileflag = readimage(image,filein)) < 0)
  {
    if (fileflag == -1)
    {
      printf("\nError opening %s\n",filein);
      exit(0);
    }
    else if (fileflag == -2)
    {
      printf("\nError reading %s \n",filein);
      exit(0);
    }
  }

  threshold(image,image,254);                                    /* Extract the bubble boundaries */

  image_analysis(image,minarea,maxarea,x1,y1,x2,y2,resultfile);  /* Characterise the image        */

  binaryto255(image,image);                                  /* Convert image for display purposes    */

  if ((fileflag = writeimage(image,fileout)) < 0)
  {
    if (fileflag == -1)
    {
      printf("\nError opening %s\n",fileout);
      exit(0);
    }
    else if (fileflag == -2)
    {
      printf("\nError writing %s \n",fileout);
      exit(0);
    }
  }
}

/*****
Function : title()

Description : Displays the program title.

Argument list : void

Return value : void
*****/
void title()
{
  system("clear");

  printf("_____________________________________________________");
  printf("\n                    FROTHMAN                       \n\n");
  printf(" Analysis of manually segmented surface froth images \n");
  printf("_____________________________________________________");
  printf("\n");
}
```

```
/*****
FILE : imagedef.h

DESCRIPTION : Header file containing image definitions.
*****/
#if !defined (IMAGEDEF)
#define (IMAGEDEF)

#define IMAGESIZE 512                                          /* Maximum image size            */
#define INFINITY  262144                                       /* Maximum image area : 515x512  */

#define PI                3.14159
#define SQUARE(a)         ((a)*(a))
#define INRANGE(t1,a,t2)  (((t1)<=(a))&&((t2)>=(a)))
#define ROUND(a)          (((fabs(a)-(int)fabs(a))<= 0.5)?floor(a):ceil(a))

typedef unsigned char BYTE;                                    /*  8 bit data type              */

#endif
```

```
/*****
FILE : imageio.h

DESCRIPTION : Header file for the module imageio.c.
*****/
#include <stdio.h>
#include <stdlib.h>
#include "imagedef.h"

/* Function prototypes */
int readimage();
int writeimage();



/*****
FILE : imageio.c

DESCRIPTION : Functions for performing image I/O on the SUN workstation.
*****/
#include "imageio.h"

/*****
Function : readimage()

Description : This function reads a specified CFI (grey level) file from disk.

Argument list : BYTE oi[][]      (original) image
                char *filename   file to be read (includes full path to file)

Return value :    0 : image read successfully
                 -1 : error opening image file
                 -2 : error reading image file
****/
int readimage(oi, filename)
BYTE oi[IMAGESIZE][IMAGESIZE];
char *filename;
{
  int i, num, number;
  FILE *fptr;

  printf("Loading : %s\n",filename);

  num = IMAGESIZE * sizeof(BYTE) / sizeof(double);
  if ((fptr = fopen( filename, "rb" )) != NULL)
  {
    for (i=0; i<IMAGESIZE; i++)
      if (fread(oi[i], sizeof(double), num, fptr ) != num)
      {
        fclose(fptr);
        return (-2);
      }
    fclose(fptr);
    return (0);
  }
  else
    return (-1);
}

/*****
Function : writeimage()

Description : This function saves a specified CFI file to disk.

Argument list : BYTE oi[][]      image
                char *filename   file to be written

Return value :    0 : image written successfully
                 -1 : error opening image file
                 -2 : error writing image file
*****/
int writeimage(oi, filename)
BYTE oi[IMAGESIZE][IMAGESIZE];
char *filename;
```

```
{
  int i, num;
  FILE *fptr;

  printf("Writing : %s\n",filename);

  num = IMAGESIZE * sizeof(BYTE) / sizeof(double);
  if ((fptr = fopen( filename, "wb" )) != NULL)
  {
    for (i=0; i<IMAGESIZE; i++)
      if (fwrite( oi[i], sizeof(double), num, fptr ) != num)
      {
        fclose(fptr);
        return (-2);
      }
    fclose(fptr);
    return (0);
  }
  else
    return (-1);
}
```

```
/*****
FILE : imagutil.h

DESCRIPTION : Header file for imagutil.c.
*****/
#include "imagedef.h"

/* Function prototypes */
void threshold();
void subtract();
void fillborder();
void binaryto255();



/*****
FILE : imagutil.c

DESCRIPTION : Functions for performing general image operations.
*****/
#include "imagutil.h"

/*****
Function : threshold()

Description : This function performs image thresholding according to equation (5.1) given in Chapter 5.

Argument list : BYTE oi[][]      original grey level image
                BYTE ti[][]      thresholded binary image
                BYTE T           threshold level (range : 0 to 255)

Return value : void
*****/
void threshold(oi,ti,T)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE ti[IMAGESIZE][IMAGESIZE];
BYTE T;
{
   int col, row;

   for (row=0;row<IMAGESIZE;row++)
      for (col=0;col<IMAGESIZE;col++)
         if (oi[row][col] >= T)
           ti[row][col] = 1;
         else
           ti[row][col] = 0;
}

/*****
Function : subtract()

Description : This function subtracts two images. It assumes that the pixel values of oi1[][] are greater
              than or equal to those of oi2[][] so that no negative values are ever obtained. This
              assumption is valid when determining the residue edge detected images G_c(x,y) and G_o(x,y),
              after performing grey level morphological closing or opening, as discussed in Chapter 6,
              section 6.2.

Argument list : BYTE oi1[][]     base image
                BYTE oi2[][]     image to be subtracted
                BYTE si[][]      resultant image

Return value : void
*****/
void subtract(oi1,oi2,si)
BYTE oi1[IMAGESIZE][IMAGESIZE];
BYTE oi2[IMAGESIZE][IMAGESIZE];
BYTE si[IMAGESIZE][IMAGESIZE];
{
   int col, row;

   for (row=0;row<IMAGESIZE;row++)
      for (col=0;col<IMAGESIZE;col++)
         si[row][col] = oi1[row][col] - oi2[row][col];
}
```

```
/*****
Function : fillborder()

Description : Function to fill the border of an image with a specified colour and of specified width. This
              function is merely used to tidy up the borders of the automatically segmented images.

Argument list : BYTE pic[][]      image to be bordered
                BYTE bw           width of border
                BYTE bc           border colour

Return value : void
*****/
void fillborder(pic,bw,bc)
BYTE pic[IMAGESIZE][IMAGESIZE];
BYTE bw;
BYTE bc;
{
   int i, j, col, row;

   for (row=0;row<IMAGESIZE;row++)                                  /* Vertical border fill          */
     for (j=0;j<bw;j++)
     {
        pic[row][j] = bc;
        pic[row][IMAGESIZE-1-j] = bc;
     }

   for (col=bw;col<IMAGESIZE-bw;col++)                              /* Horizontal border fill        */
     for (i=0;i<bw;i++)
     {
        pic[i][col] = bc;
        pic[IMAGESIZE-1-i][col] = bc;
     }
}

/*****
Function : binaryto255()

Description : This function stretches the dynamic range of binary files for display purposes, by
              multiplying the individual pixel values of the binary image by 255. Thus when the image is
              displayed the features of interest, such as the bubble boundaries or highlights, will be
              white (255) and the rest of the image will be black (0).

Argument list : BYTE ti[][]       thresholded binary image, containing values of 0 or 1
                BYTE si[][]       stretched image, containing values of 0 or 255

Return value : void
*****/
void binaryto255()
BYTE ti[IMAGESIZE][IMAGESIZE];
BYTE si[IMAGESIZE][IMAGESIZE];
{
  int row, col;

  for (row=0;row<IMAGESIZE;row++)
   for (col=0;col<IMAGESIZE;col++)
     si[row][col] = ti[row][col] * 255;
}
```

```
/*****
FILE : filter.h

DESCRIPTION : Header file for the module filter.c.
*****/
#include "imagedef.h"

/* Compiler directives for selecting the filter type to be used */
/* 0 : filter not selected                                      */
/* 1 : filter selected                                          */
#define AVERAGE 1
#define MEDIAN  0

/* Function prototypes */
void average();
void median();
static int sort_function();



/*****
FILE : filter.c

DESCRIPTION : Image preprocessing module for the automatic processing of surface froth images.
*****/
#include "filter.h"

/****
Function : average()

Description : Function that performs low-pass filtering using a 3x3 kernel.

Argument list : oi[][]   original (grey level) image
                fi[][]   filtered (grey level) image

Return value : void
*****/
void average(oi, fi)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE fi[IMAGESIZE][IMAGESIZE];
{
   int col, row, sum;

   for (row=1; row<IMAGESIZE-1; row++)
      for (col=1; col<IMAGESIZE-1; col++)
      {
         sum = oi[row-1][col-1] + oi[row-1][col] + oi[row-1][col+1] +
               oi[row][col-1] + oi[row][col] + oi[row][col+1] +
               oi[row+1][col-1] + oi[row+1][col] + oi[row+1][col+1];
         fi[row][col] = (BYTE) (sum/9);
      }
}

/*****
Function : median()

Description : Function that performs median filtering using a 3x3 kernel.

Argument list : oi[][]   original image
                fi[][]   filtered image

Return value : void
*****/
void median(oi, fi)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE fi[IMAGESIZE][IMAGESIZE];
{
  int index, row, col, i, j;
  BYTE filtermask[9];

  for (row=1;row<IMAGESIZE-1;row++)
    for (col=1;col<IMAGESIZE-1;col++)
    {
      index = 0;
```

```
      for (i=-1;i<=1;i++)
        for(j=-1;j<=1;j++)
        {
          filtermask[index] = oi[row+i][col+j];
          index++;
        }

      qsort(filtermask,9,sizeof(BYTE),sort_function);

      fi[row][col] = filtermask[4];
    }
}

/*****
Function : sort_function()

Description : This function is used by the qsort() routine, in the function median().

Argument list :  BYTE *a, *b      numerical values to be sorted

Return value : negative if *a < *b
               0 if *a = *b
               positive if *a > *b
*****/
static int sort_function(a, b)
BYTE *a;
BYTE *b;
{
  return (*a - *b);
}
```

```
/*****
FILE : morph.h

DESCRIPTION : Header file for the module morph.c.
*****/
#include <malloc.h>
#include <math.h>
#include "imagedef.h"

/* Compiler directives for selecting the required composite   */
/* morphological operation (closing or opening) to be used.   */
/* These directives insure that the appropriate DSSE is       */
/* generated for the required operation                       */
/* 0 : operation not selected                                 */
/* 1 : operation selected                                     */
#define CLOSING 1
#define OPENING 0

/* Binary structuring elements */
#define  RECT2x1   0
#define  RECT1x2   1
#define  PLUS3x3   2

/* Function prototypes */
char **declare_spherical_se();
void remove_spherical_se();

void gs_erode();                                    /* Grey level morphological operations    */
void gs_dilate();
void gs_open();
void gs_close();

void b_erode();                                     /* Binary morphological operations        */
void b_dilate();
void b_open();
void b_close();



/*****
FILE : morph.c

DESCRIPTION : Functions for performing morphological operations on grey level and binary images.
*****/
#include "morph.h"

/*****
Function : declare_spherical_se()

Description : Allocates memory for and generates an NxN discrete spherical structuring element (DSSE). The
              size of the DSSE array is given by (2*radius+1) x (2*radius+1).

Argument list : int r            radius of structuring element

Return value : char **           pointer to structuring element array
*****/
char **declare_spherical_se(r)
int r;
{
   char **sphere;                                                  /* Structuring element          */
   int i, j;
   double z;

   sphere = (char **) calloc(2*r+1,sizeof(char *));                /* Allocate storage for the DSSE */
   for (i=0;i<2*r+1;i++)
     sphere[i] = (char *) calloc(2*r+1,sizeof(char));

   for (i=-r;i<=r;i++)                                             /* Generate the DSSE            */
     for (j=-r;j<=r;j++)
     {
       z = r*r - i*i - j*j;
       if (z < 0.0 )
          sphere[i+r][j+r] = -1.0;                                 /* Data points to be ignored    */
       else
```

```
      {
#if OPENING
        sphere[i+r][j+r] = (char) ROUND((float) sqrt(z));        /* Discrete hemisphere          */
#endif

#if CLOSING
        sphere[i+r][j+r] = (char) ROUND(r - (float) sqrt(z));    /* Inverted discrete hemisphere */
#endif
      }
    }

  return (sphere);
}

/*****
Function : remove_spherical_se()

Description : Frees the memory reserved for the NxN discrete spherical structuring element.

Argument list : BYTE **sphere    pointer to the structuring element array
                int r            radius of structuring element

Return value : void
*****/
void remove_spherical_se(sphere, r)
char **sphere;
int r;
{
  int i;

  for (i=2*r+1;i>0;i--)  free((char*) sphere[i]);
  free((char) sphere);
}

/*****
Function : gs_erode()

Description : Function that performs grey level erosion using the structuring element passed to the
              function.

Argument list :  BYTE oi[][]     original image
                 BYTE ei[][]     eroded image
                 char **se       pointer to structuring element array
                 int r           radius of structuring element

Return value : void
*****/
void gs_erode(oi, ei, se, r)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE ei[IMAGESIZE][IMAGESIZE];
char **se;
int r;
{
  int row, col, i, j, min, temp;

  for (row=r;row<IMAGESIZE-r;row++)
    for (col=r;col<IMAGESIZE-r;col++)
    {
      min = 255;
      for (i=-r;i<=r;i++)
        for (j=-r;j<=r;j++)
        {
          if (se[i+r][j+r] > 0)
          {
            temp = oi[row+i][col+j] - se[i+r][j+r];
            if (temp < 0) temp = 0;                              /* Watch for underflow          */
            if (temp < min) min = temp;
          }
        }
      ei[row][col] = (BYTE) min;
    }
}
```

```
/*****
Function : gs_dilate()

Description : Function that performs grey level dilation, using the structuring element passed to the
              function.

Argument list : BYTE oi[][]      original image
                BYTE di[][]      dilated image
                char **se        pointer to structuring element array
                int r            radius of structuring element

Return value : void
*****/
void gs_dilate(oi,di,se,r)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE di[IMAGESIZE][IMAGESIZE];
char **se;
int r;
{
  int row, col, i, j;
  int max, temp;

  for (row=r;row<IMAGESIZE-r;row++)
    for (col=r;col<IMAGESIZE-r;col++)
    {
      max = 0;
      for (i=-r;i<=r;i++)
        for (j=-r;j<=r;j++)
        {
          if (se[i+r][j+r] > 0)
          {
            temp = oi[row+i][col+j] + se[i+r][j+r];
            if (temp > 255) temp = 255;                    /* Watch for overflow          */
            if (temp > max) max = temp;
          }
        }
      di[row][col] = (BYTE) max;
    }
}

/*****
Function : gs_open()

Description : Function that performs grey level morphological opening using the structuring element passed
              to the function.

Argument list : BYTE oi[][]      original image
                BYTE opi[][]     opened image
                char ** se       pointer to structuring element array
                int r            radius of structuring element

Return value : void
*****/
void gs_open(oi,opi,se,r)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE opi[IMAGESIZE][IMAGESIZE];
char **se;
int r;
{
   int col, row;
   BYTE temp[IMAGESIZE][IMAGESIZE];

   for (row=0;row<IMAGESIZE;row++)
     for (col=0;col<IMAGESIZE;col++)
     {
       temp[row][col] = oi[row][col];
       opi[row][col] = oi[row][col];
     }

   gs_erode(oi,temp,se,r);
   gs_dilate(temp,opi,se,r);
}
```

```
/****
Function : gs_close()

Description : Function that performs grey level morphological closing, using the structuring element passed
              to the function.

Argument list : BYTE oi[][]       original image
                BYTE cli[][]      closed image
                char **se         structuring element
                int r             radius of structuring element

Return value : void
*****/
void gs_close(oi,cli,se,r)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE cli[IMAGESIZE][IMAGESIZE];
char **se;
int r;
{
   int col, row;
   BYTE temp[IMAGESIZE][IMAGESIZE];

   for (row=0;row<IMAGESIZE;row++)
     for (col=0;col<IMAGESIZE;col++)
     {
       temp[row][col] = oi[row][col];
       cli[row][col] = oi[row][col];
     }

    gs_dilate(oi,temp,se,r);
    gs_erode(temp,cli,se,r);
}

/*****
Function : b_erode()

Description : Function that performs binary morphological erosion using the specified structuring element.

Argument list : BYTE oi[][]       original image
                BYTE ei[][]       eroded image
                int SE            type of structuring element to be used : RECT2x1, RECT1x2 or PLUS3x3.

Return value : void
*****/
void b_erode(oi,ei,SE)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE ei[IMAGESIZE][IMAGESIZE];
int SE;
{
   int col, row;

   if (SE == RECT2x1)
   {
     for (row=0;row<IMAGESIZE-1;row++)
        for (col=0;col<IMAGESIZE;col++)
           ei[row][col] = (oi[row][col] && oi[row+1][col]);
   }
   else if (SE == RECT1x2)
   {
     for (row=0;row<IMAGESIZE;row++)
        for (col=0;col<IMAGESIZE-1;col++)
           ei[row][col] = (oi[row][col] && oi[row][col+1]);
   }
   else if (SE == PLUS3x3)
   {
     for (row=1;row<IMAGESIZE-1;row++)
        for (col=1;col<IMAGESIZE-1;col++)
           ei[row][col] = (oi[row][col] && oi[row-1][col] && oi[row+1][col] &&
                           oi[row][col-1] && oi[row][col+1]);
   }
}
```

```
/*****
Function : b_dilate()

Description : Function that performs binary morphological dilation using the specified structuring element.

Argument list : BYTE oi[][]      original image
                BYTE di[][]      dilated image
                int SE           type of structuring element to be used : RECT2x1, RECT1x2 or PLUS3x3.

Return value : void
*****/
void b_dilate(oi,di,SE)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE di[IMAGESIZE][IMAGESIZE];
int SE;
{
   int col, row;

   if (SE == RECT2x1)
   {
      for (row=0;row<IMAGESIZE-1;row++)
        for (col=0;col<IMAGESIZE;col++)
           di[row][col] = (oi[row][col] || oi[row+1][col]);
   }
   else if (SE = RECT1x2)
   {
      for (row=0;row<IMAGESIZE;row++)
        for (col=0;col<IMAGESIZE-1;col++)
           di[row][col] = (oi[row][col] || oi[row][col+1]);
   }
   else if (SE = PLUS3x3)
   {
      for (row=1;row<IMAGESIZE-1;row++)
        for (col=1;col<IMAGESIZE-1;col++)
           di[row][col] = (oi[row][col] || oi[row-1][col] || oi[row+1][col] ||
                           oi[row][col-1] || oi[row][col+1]);
   }
}

/*****
Function : b_open()

Description : Function that performs binary morphological opening using the specified structuring element.

Argument list : BYTE oi[][]      original image
                BYTE opi[][]     opened image
                int SE           type of structuring element to be used : RECT2x1, RECT1x2 or PLUS3x3.

Return value : void
*****/
void b_open(oi,opi,SE)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE opi[IMAGESIZE][IMAGESIZE];
int SE;
{
   int col, row;
   BYTE temp[IMAGESIZE][IMAGESIZE];

   for (row=0;row<IMAGESIZE;row++)
      for (col=0;col<IMAGESIZE;col++)
      {
        opi[row][col] = 0;
        temp[row][col] = 0;
      }

   b_erode(oi,temp,SE);
   b_dilate(temp,opi,SE);
}
```

```
/*****
Function : b_close()

Description : This function performs binary morphological closing using the specified structuring element.

Argument list : BYTE oi[][]       original image
                BYTE cli[][]      closed image
                int SE            type of structuring element to be used : RECT2x1, RECT1x2 or PLUS3x3.

Return value : void
*****/
void b_close(oi,cli,SE)
BYTE oi[IMAGESIZE][IMAGESIZE];
BYTE cli[IMAGESIZE][IMAGESIZE];
int SE;
{
   int col, row;
   BYTE temp[IMAGESIZE][IMAGESIZE];

   for (row=0;row<IMAGESIZE;row++)
      for (col=0;col<IMAGESIZE;col++)
      {
         cli[row][col] = 0;
         temp[row][col] = 0;
      }

   b_dilate(oi,temp,SE);
   b_erode(temp,cli,SE);
}
```

```
/*****
FILE : thin.h

Description : Header file for the module thin.c.
*****/
#include <stdlib.h>
#include "imagedef.h"

#define FAILED          0
#define PASSED          1
#define FALSE           0
#define TRUE            1
#define DELETED         0
#define NOTHINGDELETED  1

/* Function prototypes */
void skeletonize();
void step1();
void step2();
int delete();
void setupmask();
int condition1();
int condition2();
int condition3();
int condition4();
int condition5();
int condition6();



/*****
FILE : thin.c

DESCRIPTION : Functions for skeletonizing an image. The algorithm implemented here is that presented by
              Gonzalez and Wintz (1987, pp. 398-402) and is discussed in Chapter 6, section 6.4.2 (B).
*****/
#include "thin.h"

/*****
Function : skeletonize()

Description : This function performs iterative thinning of an image, to produce a skeleton. It operates on
              and produces a binary image.

Argument list : BYTE pic[][]      binary image to be skeletonized

Return value : void
*****/
void skeletonize(pic)
BYTE pic[IMAGESIZE][IMAGESIZE];
{
   BYTE temp[IMAGESIZE][IMAGESIZE];                    /* Temporary storage array       */
   int col, row;

   for (row=0;row<IMAGESIZE;row++)
      for (col=0;col<IMAGESIZE;col++)
         temp[row][col] = 0;

   for(;;)
   {
      step1(pic,temp);
      if (delete(pic,temp) == NOTHINGDELETED)
        break;
      step2(pic,temp);
      if (delete(pic,temp) == NOTHINGDELETED)
        break;
   }
}
```

```
/*****
Function : step1()

Description : Function that performs the first step of the skeletonization algorithm.

Argument list : BYTE pic[][]      binary image to be skeletonized
                BYTE temp[][]     temporary storage image

Return value : void
*****/
void step1(pic,temp)
BYTE pic[IMAGESIZE][IMAGESIZE];
BYTE temp[IMAGESIZE][IMAGESIZE];
{
  int col, row;
  BYTE mask[9];

  for (row=1;row<IMAGESIZE-1;row++)
    for (col=1;col<IMAGESIZE-1;col++)
    {
      if (pic[row][col] == 1)
      {
        setupmask(pic,mask,row,col);
        temp[row][col] = (condition1(mask) * condition2(mask) * condition3(mask) * condition4(mask));
      }
    }
}

/*****
Function : step2()

Description : Function that performs the second step of the skeletonization algorithm.

Argument list : BYTE pic[][]      binary image to be skeletonized
                BYTE temp[][]     temporary working picture

Return value : void
*****/
void step2(pic,temp)
BYTE pic[IMAGESIZE][IMAGESIZE];
BYTE temp[IMAGESIZE][IMAGESIZE];
{
  int col, row;
  BYTE mask[9];

  for (row=1;row<IMAGESIZE-1;row++)
    for (col=1;col<IMAGESIZE-1;col++)
      if (pic[row][col] == 1)
      {
        setupmask(pic,mask,row,col);
        temp[row][col] = (condition1(mask) * condition2(mask) * condition5(mask) * condition6(mask));
      }
}

/*****
Function : delete()

Description : This function removes those pixels flagged for deletion.

Argument list : BYTE pic[][]      binary image to be skeletonized
                BYTE temp[][]     temporary working image

Return value : DELETED if pixels deleted
               NOTHINGDELETED if no pixels are deleted
*****/
int delete(pic,temp)
BYTE pic[IMAGESIZE][IMAGESIZE];
BYTE temp[IMAGESIZE][IMAGESIZE];
{
  int flag, col, row;

  flag = NOTHINGDELETED;
  for (row=0;row<IMAGESIZE;row++)
    for (col=0;col<IMAGESIZE;col++)
```

```
      if (temp[row][col] == 1)
      {
         temp[row][col] = 0;                                  /* Keep temporary image clear    */
         pic[row][col] = 0;
         flag = DELETED;
      }

   return (flag);
}

/*****
Function : setupmask()

Description : Function to set up the 3x3 mask used to flag pixels for deletion. The mask is actually
              implemented as the 1-D array :
```

mask[0]	mask[1]	mask[2]	mask[3]	mask[4]	mask[5]	mask[6]	mask[7]	mask[8]

```
              where the mask elements are related to the arrangement illustrated in Figure 6.25, Chapter
              6, as follows :
                  q
              mask[0] = P1;      mask[1] = P2;    mask[2] = P3;    mask[3] = P4;    mask[4] = P5;
              mask[5] = P6;      mask[6] = P7;    mask[7] = P8;    mask[8] = P9;

Argument list : BYTE pic[][]     binary image to be skeletonized
                BYTE mask[]      3x3 mask to be set up
                int row, col     current row, column coordinates

Return value : void
*****/
void setupmask(pic,mask,row,col)
BYTE pic[IMAGESIZE][IMAGESIZE];
BYTE mask[9];
int row;
int col;
{
   mask[1] = pic[row-1][col];
   mask[2] = pic[row-1][col+1];
   mask[3] = pic[row][col+1];
   mask[4] = pic[row+1][col+1];
   mask[5] = pic[row+1][col];
   mask[6] = pic[row+1][col-1];
   mask[7] = pic[row][col-1];
   mask[8] = pic[row-1][col-1];
}

/*****
Function : condition1()

Description : Checks the condition  2 <= N(p1) <= 6.

Argument list : BYTE mask[]      3x3 mask used to flag points for deletion

Return value : PASSED if condition satisfied
               FAILED if condition not satisfied
*****/
int condition1(mask)
BYTE mask[9];
{
  int i, sum = 0;

  for (i=1;i<9;i++)
    if (mask[i]!=0) sum += 1;

  if ((sum>=2) && (sum<=6))
    return (PASSED);
  else
    return (FAILED);
}
```

```
/*****
Function : condition2()

Description : Checks the condition S(p1) = 1.

Argument list : BYTE mask[]      3x3 mask used to flag points for deletion

Return value : PASSED if condition satisfied
               FAILED if condition not satisfied
*****/
int condition2(mask)
BYTE mask[9];
{
   int sum = 0;

   if ((mask[1]==0) && (mask[2]==1)) sum+=1;
   if ((mask[2]==0) && (mask[3]==1)) sum+=1;
   if ((mask[3]==0) && (mask[4]==1)) sum+=1;
   if ((mask[4]==0) && (mask[5]==1)) sum+=1;
   if ((mask[5]==0) && (mask[6]==1)) sum+=1;
   if ((mask[6]==0) && (mask[7]==1)) sum+=1;
   if ((mask[7]==0) && (mask[8]==1)) sum+=1;
   if ((mask[8]==0) && (mask[1]==1)) sum+=1;

   if (sum == 1)
      return (PASSED);
   else
      return (FAILED);
}

/*****
Function : condition3()

Description : Checks the condition p2 AND p4 AND p6 = 0.

Argument list : BYTE mask[]      3x3 mask used to flag points for deletion

Return value : PASSED if condition satisfied
               FAILED if condition not satisfied
*****/
int condition3(mask)
BYTE mask[9];
{
   if ((mask[1] && mask[3] && mask[5]) == 0)
     return (PASSED);
   else
     return (FAILED);
}

/*****
Function : condition4()

Description : Checks the condition p4 AND p6 AND p8 = 0.

Argument list : BYTE mask[]      3x3 mask used to flag points for deletion

Return value : PASSED if condition satisfied
               FAILED if condition not satisfied
*****/
int condition4(mask)
BYTE mask[9];
{
   if ((mask[3] && mask[5] && mask[7]) == 0)
     return (PASSED);
   else
     return (FAILED);
}
```

```
/*****
Function : condition5()

Description : Checks the condition p2 AND p4 AND p8 = 0.

Argument list : BYTE mask[]      3x3 mask used to flag points for deletion

Return value : PASSED if condition satisfied
               FAILED if condition not satisfied
*****/
int condition5(mask)
BYTE mask[9];
{
   if ((mask[1] && mask[3] && mask[7]) == 0)
     return (PASSED);
   else
     return (FAILED);
}

/*****
Function : condition6()

Description : Checks the condition p2 AND p6 AND p8 = 0.

Argument list : BYTE mask[]      3x3 mask used to flag points for deletion

Return value : PASSED if condition satisfied
               FAILED if condition not satisfied
*****/
int condition6(mask)
BYTE mask[9];
{
   if ((mask[1] && mask[5] && mask[7]) == 0)
     return (PASSED);
   else
     return (FAILED);
}
```

```
/*****
FILE : analysis.h

DESCRIPTION : Header file for the module analysis.c.
*****/
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "imagedef.h"

/* Constants used in genblob() */
#define MAXBLOB  1000     /* maximum allowed number of blobs         */
#define MAXRUN   65535    /* maximum allowed number of runs          */
#define MAXCRP   65535    /* maximum allowed number of critical runs */
#define MAXNUM   65535    /* maximum number of blob pointers         */
#define NROW     512      /* maximum number of rows in image         */
#define MAXROW   511      /* maximum row (starting from 0)           */
#define NCOL     512      /* maximum number of columns               */
#define MAXCOL   511      /* maximum col (starting from 0)           */
#define ENDLINE  0x40     /* a flag to test for the end of a row     */
#define PERFLAG  0        /* 0 is corrected perimeter calculation    */


/* Compiler directives that can speed up processing */
/* 0 : computation of the parameter is not required */
/* 1 : computation of the parameter is required     */
#define AREA  1           /* blob area                          */
#define PERIM 1           /* blob perimeter                     */
#define N1MOM 1           /* first order x and y moments        */
#define N2MOM 1           /* second order x, y and xy moments   */
#define BINARYMOD 1       /* 0 : grey level image as input      */
                          /* 1 : binary image as input          */
#define ENCLBOX 0         /* enclosing box coordinates          */
#define BORD  0           /* border point coordinate            */
#define TREE  0           /* hierarchical blob relationship     */
#define DEBUG 0           /* debugging information              */

/* structure that holds basic properties of a blob */
typedef struct blob {
                int flink;
                int rlink;
                int parent;
                int child;
                int sibling;
                BYTE color;
                BYTE xbord;
                unsigned int number;
                unsigned int perim;
                int nholes;
                int imin;
                int imax;
                int jmin;
                int jmax;
                   } BLOB;

/* Structure that holds additional statistics of a blob */
typedef struct slob {
                unsigned long area;
                unsigned long sumi;
                unsigned long sumj;
                unsigned long sumi2;
                unsigned long sumj2;
                float sumij;
                   } SLOB;

/* Structure that holds measurements for characterising blobs */
typedef struct glob {
                unsigned long area;                    /* Blob area                */
                int xcent;                             /* x centroid               */
                int ycent;                             /* y centroid               */
                float round;                           /* Circularity measure      */
                float eccent;                          /* Elipticity measure       */
                   } GLOB;
```

```
/* Structure that holds the statistical measurements for characterising surface froth structures */
typedef struct stats {
                float mean;
                float var;
                float stddev;
                float skew;
                     } STATS;

/* Function prototypes */
void image_analysis();
void zero_nonaoi();
void promfill();
int gencrp();
int genblob();
void getablob();
void update();
int alloc();
void surround();
void updperv();
void updperh();
void m_errmsg();
void characterise();
void moment();
void mark_centroid();



/*****
FILE : analysis.c

DESCRIPTION : Module that performs connected component analysis. This is modified consoft4.c code, that is
              used to analyze the "blobs" in segmented surface froth images and a complete discussion of
              the code is given in Appendix C.

              This code was modified to run on the SUN workstations so that segmented 512x512 images could
              be analyzed. Due to the quantity of global data that is defined, additional modifications are
              required if the software is to process 512x512 images on a pc running under MS-DOS.

AUTHOR : J. Knapp, Aspex Incorporated, 1989.

MODIFIED : S. Biro, January 1991.
           P. Symonds, November 1991.
*****/
#include "analysis.h"

/* ---------    External registers/counters used by labeller ---------- */
int i;            /* Current column counter - 9 bits  (max value of 511) */
int j;            /* Current line counter - 9 bits    (max value of 511) */
int leftend;      /* Left side of new run - 9 bits                       */
unsigned int bn;  /* Current blob counter - 16 bits                      */
int crun;         /* Current run number counter - 16 bits                */
int prun;         /* Previous run number counter - 16 bits               */
int curreg;       /* Pixel A's blob register - 16 bits                   */
int regabv;       /* Pixel D's blob register - 16 bits                   */
int qfree;        /* Next free blob number counter - 16 bits             */
int avail;        /* Usable blob in blob list register - 16 bits         */
int newflag;      /* 1 if new run                                        */
int dval;         /* Used to hold temp vert component of perim           */
int twobysix;     /* Holds pixel info of active CRP for perim and wstate */
int crptotal = 0; /* Total amount of CRP's used (for info only)          */

/* Variables in main module */
unsigned int bp[MAXNUM];           /* Blob pointer list (indexed by bn) */
struct blob b[MAXBLOB];
struct slob bs[MAXBLOB];
struct glob bg[MAXBLOB];

/* CRP list - bit assignments (D0 to D31).        */
/* D0 - D3  : window state                        */
/* D4 - D5  : perimeter info - vertical component */
/* D6 - D14 : crp column value - 9 bits (max 511) */
/* D31 : 1 = active blob; 0 = inactive blob       */
unsigned int cr[MAXCRP] = {0,0};
```

```
/* Run list */
unsigned int rs[65536];              /* Run start                          */
unsigned int rn[65536];              /* Run number                         */

unsigned int lptrans[64] = {         /* Corrected perimeter left LUT       */
/*  E=0, F=0 */ 0,0,5,4,0,0,5,5,5,5,0,0,4,5,0,0,
/*  E=1, F=0 */ 0,0,5,4,0,0,5,5,4,4,0,0,3,4,0,0,
/*  E=0, F=1 */ 0,0,4,3,0,0,4,4,5,5,0,0,4,5,0,0,
/*  E=1, F=1 */ 0,0,5,4,0,0,5,5,5,5,0,0,4,5,0,0};

unsigned int rptrans[64] = {         /* Corrected perimeter right LUT      */
/*  E=0, F=0 */ 0,0,5,4,0,0,5,5,5,5,0,0,4,5,0,0,
/*  E=1, F=0 */ 0,0,5,5,0,0,5,5,4,4,0,0,3,4,0,0,
/*  E=0, F=1 */ 0,0,4,3,0,0,4,4,5,5,0,0,4,5,0,0,
/*  E=1, F=1 */ 0,0,5,5,0,0,5,5,5,5,0,0,4,5,0,0};

/* Prom stuff */
unsigned int proma[512];             /* Right sum prom (area)              */
unsigned int promb[512];             /* Left sum prom (area)               */
unsigned long promc[512];            /* Right sum x prom                   */
unsigned long promd[512];            /* Left sum x prom                    */
unsigned long prome[512];            /* Right sum x^2 prom                 */
unsigned long promf[512];            /* Left sum x^2 prom                  */
unsigned int promg[512];             /* Multiplication table               */

/*****
Function : image_analysis()

Description : This function analyses a segmented (binary) surface froth image. Only blobs within a specified
              area of interest and in the range minarea <= area <= maxarea are processed to generate the
              required statistical distributions.

Argument list : BYTE image[][]    binary image to be analyzed.
                long minarea      minimum blob area to be considered.
                long maxarea      maximum blob area to be considered.
                int x1, y1        top left-hand coordinates of AOI.
                int x2, y2        bottom right-hand coordinates of AOI.
                char *filename    file to which the analysis results are written.

Return value : void
*****/
void image_analysis(image,minarea,maxarea,x1,y1,x2,y2,filename)
BYTE image[IMAGESIZE][IMAGESIZE];
long minarea;
long maxarea;
int x1;
int y1;
int x2;
int y2;
char *filename;
{
   int backblob;                     /* Root or background blob            */
   int err;
   int threshold = 1;
   float pixelwidth = 1.0;

   zero_nonaoi(image,x1,y1,x2,y2);   /* Zero non-area of interest          */
   promfill(pixelwidth);             /* Generate LUTs                      */
   err = gencrp(image,threshold);    /* Generate run-length list           */
   if (err)                          /* err = 1, there is a problem        */
   {
     m_errmsg("Error in generating critical run-length list.\n");
     exit(0);
   }
   err = genblob(&backblob);         /* Generate blobs                     */
   if (err)                          /* err = 1, there is a problem        */
   {
     m_errmsg("Error in generating the blobs.\n");
     exit(0);
   }
   characterise(image,backblob,minarea,maxarea,x1,y1,x2,y2,filename);
}
```

```
/*****
Function : zero_nonaoi()

Description : This function zeros the area surrounding the area of interest of the image. This region is
              then the background or root blob of the image. See Appendix C, section C.3, for a discussion
              on this.

Argument list : BYTE image[][]    image to be processed
                int x1, y1        top left-hand coordinate of AOI
                int x2, y2        bottom right-hand coordinate of AOI

Return value : void
*****/
void zero_nonaoi(image,x1,y1,x2,y2)
BYTE image[IMAGESIZE][IMAGESIZE];
int x1;
int y1;
int x2;
int y2;
{
   int col, row;

   for (row=0;row<y1;row++)
     for (col=0;col<NCOL;col++)
        image[row][col] = 0;

   for (row=y2;row<NROW;row++)
     for (col=0;col<NCOL;col++)
        image[row][col] = 0;

   for (row=y1;row<y2;row++)
     for (col=0;col<=x1;col++)
        image[row][col] = 0;

   for (row=y1;row<y2;row++)
     for (col=x2;col<NCOL;col++)
        image[row][col] = 0;
}

/*****
Function : promfill()

Description : This function generates the LUTs that are used to speed up the calculations for the area,
              first moments and second moments of a blob. The function uses the variable pw to define the
              aspect ratio of the image when generating the LUTs, so that the geometric parameters are
              rotation invariant.

Argument list : float pw          pixel width.

Return value : void
*****/
void promfill(pw)
float pw;
{
   unsigned int i = 0;
   double fi = 0.0;
   float tmp2,tmp3,tmp12;

   tmp2 = pw/2.0;
   tmp3 = SQUARE(pw)/3.0;
   tmp12 = SQUARE(pw)/12.0;

   while (i<512)
   {
     promg[i] = i*i;

#if AREA
     proma[i] = (unsigned int)ROUND(fi*pw+tmp2) ;
     if (i)
        promb[i] = (unsigned int)ROUND(fi*pw-tmp2);
     else
        promb[i] = 0;
#endif
```

```
#if N1MOM
     promc[i] = (unsigned long)ROUND(((fi+0.5)*(fi+0.5))*tmp2);
     if (i)
       promd[i] = (unsigned long)ROUND(((fi-0.5)*(fi-0.5))*tmp2);
     else
       promd[i] = 0;
#endif

#if N2MOM
     prome[i] = (unsigned long)ROUND((((fi+0.5)*(fi+0.5)*(fi+0.5))*tmp3)-((fi+0.5)*tmp12));
     if (i)
       promf[i] = (unsigned long)ROUND((((fi-0.5)*(fi-0.5)*(fi-0.5))*tmp3)-((fi-0.5)*tmp12));
     else
       promf[i] = 0;
#endif

#if DEBUG
     printf("I: %d, proma: %lu, promb: %lu, promc: %lu, promd: %lu, prome: %lu, promf:
             %lu\n",i,proma[i],promb[i],promc[i],promd[i],prome[i],promf[i]);
#endif

     ++i;
     fi += 1.0;
   }
}

/*****
Function : gencrp()

Description : This function generates a critical run-length list from the binary image and computes the
              window state in a 3x2 neighbourhood, at the end of each run.

Argument list : BYTE image[][]    image to be processed
                int thr           threshold level

Return value : int 0 : critical run-length list generated OK.
                   1 : error generating critical run-length list.
*****/
int gencrp(image,thr)
BYTE image[IMAGESIZE][IMAGESIZE];
int thr;
{
   BYTE *pixb;
   BYTE *pixd;
   BYTE *pixe;
   int row = 1;
   int wstate;                          /* Window state                 */
   int z = 0;
   int i;
   unsigned int *crpptr;                /* Pointer to the array cr[]    */

   crpptr = (unsigned int *)cr;
   crptotal = 0;

   for (row=1; row<MAXROW; row++)       /* Go from row = 1 to 510       */
   {
     pixb = &image[row-1][1];
     pixd = &image[row][1];
     pixe = &image[row+1][0];
     for (i=0, wstate=0; i<MAXCOL; i++) /* Go from col = 0 to 510       */
     {
       wstate &= 0xC;

#if BINARYMOD                           /* Image already binary ?       */
       z = *pixd++ & 8;
       z |= (*pixb++ & 4);
#else                                   /* If not, threshold it         */
       if (*pixd++ >= thr) z = 0x8;
       else z = 0;
       if (*pixb++ >= thr) z |= 0x4;
       else;
#endif
```

```
        if (z ^ wstate)
        {
          wstate >>= 2;                    /* Divide by 2                    */
          wstate |= z;

#if BINARYMOD
          wstate |= (*pixe++ & 0x20);
          wstate |= (*pixe & 0x10);
#else
          if (*pixe++ >= thr) wstate |= 0x20;
          else;
          if (*pixe >= thr) wstate |= 0x10;
          else;
#endif
          *crpptr++ = (i<<7) | wstate;     /* Save column and window state information       */
          if (++crptotal >= MAXCRP)        /* Too many CRP's                                 */
            return(1);
        }
        else ++pixe;
      }
      ++pixb;
      ++pixd;
      ++pixe;
      *crpptr++ = 0x0000FFC0;              /* The SUN workstation works with 32 bit integers */
      if (++crptotal >= MAXCRP)            /* Too many CRP's                                 */
      {
        crptotal = 0;                                                                        #
        return(1);
      }
      else;
    }
    return(0);
}

/*****
Function : genblob()

Description : This function uses the critical run-length list generated by gencrp(), to generate the
              individual blobs and to compute the blob parameters.

Argument list : int *backblob     pointer to the root blob.

Return value : int 0 : blobs generated OK.
                   1 : error in generating blobs.
*****/
int genblob(backblob)
int *backblob;
{
  int crp = 0;
  int err = 0;

  *backblob = 1;          /* Set up background blob                                        */
  bp[1] = 1;              /* Set pointer of current blob number to run number              */
  b[1].number = 1;        /* Remember blob pointer offset in blob data for future use      */
  b[1].color = 0;         /* Background                                                    */

#if PERIM
  b[1].perim = 0;
#endif

#if AREA
  bs[1].area = 0.0;
#endif

#if N1MOM
  bs[1].sumi = 0;
  bs[1].sumj = 0;
#endif

#if N2MOM
  bs[1].sumi2 = 0;
  bs[1].sumij = 0;
  bs[1].sumj2 = 0;
#endif
```

```
#if ENCLBOX
  b[1].imin = 0;
  b[1].imax = 0;
  b[1].jmin = 0;
  b[1].jmax = 0;
#endif

#if BORD
  b[1].xbord = 0;
#endif

  b[1].rlink = 1;               /* Backward link to itself since it's the only blob      */
  b[1].flink = 1;               /* Forward link to blob 1 for now                        */

#if TREE
  b[1].parent = -1;             /* Parent unknown                                        */
  b[1].sibling = -1;            /* Sibling unknown                                       */
  b[1].child = -1;              /* Child background blob NA                              */
  b[1].nholes = 0;              /* No holes yet                                          */
#endif

  rs[1] = 0;
  rn[1] = 1;
  prun = 1;
  crun = 2;
  bn = 2;                       /* Next blob                                             */
  rs[2] = 0;                    /* Run start                                             */
  qfree = 2;                    /* Set qfree to next available blob in list              */
  avail = -1;                   /* No blobs merged and freed up yet                      */
  newflag = 0;                  /* Zero newflag at beginning of line                     */
  dval = 3;                     /* Diagonal value at beginning of line                   */
  regabv = 1;                   /* Get above run's blob number                           */
  curreg = 1;                   /* Always starts with background run                     */

  j = 1;
  while (j<MAXROW)              /* Loop for row = 1 to 510                               */
  {
    i = cr[crp]>>7;             /* Extract column                                        */
    if (cr[crp] & ENDLINE)      /* End of the line ?                                     */
    {
      ++prun;
      update();
      ++crun;
      getablob();               /* Get above run's blob number                           */
      curreg = regabv;          /* Always starts with background run                     */
      rs[crun] = 0;
      newflag = 0;              /* Zero newflag at beginning of line                     */
      dval = 3;                 /* Only vertical component                               */
      ++j;                      /* Next line now                                         */
    }
    else
    {
      twobysix = (cr[crp] & 0x3F);      /* Mask on the first 6 bits                      */
      switch (twobysix & 0x0F)          /* Mask on the 4 bits for the window state and   */
      {                                 /* check for a particular state that has occurred*/
         case 14:                       /* (see Appendix C, section C.2.2 (C))           */
         case 1 : ++prun;               /* Pixel B & D are different so increment prun    */
                  surround(backblob);
                  break;
         case 13:
         case 2 : if (newflag)          /* End of a new run, allocate a new blob         */
                  {
                    newflag = 0;
                    err = alloc();
                    if (err) exit(0);
                  }
                  else;
                  update();
#if PERIM
                  updperv();
                  updperh();
#endif
                  curreg = regabv;
                  rs[++crun] = ++i;     /* Not end of line yet                           */
```

```
                break;
        case 12:
        case 3 : ++prun;                /* Point at next prun                   */
                 update();              /* Update crun data                     */
#if PERIM
                 updperv();
#endif
                 getablob();            /* Get above run's blob number          */
                 curreg = regabv;       /* Set equal                            */
                 rs[++crun] = ++i;      /* Not end of line yet                  */
                 break;
        case 11:
        case 4 : ++prun;
                 getablob();
                 leftend = ++i;
                 break;
        case 6 : ++prun;
                 surround(backblob);
                 update();
#if PERIM
                 updperv();
#endif
                 rs[++crun] = ++i;      /* Not end of line yet                  */
                 leftend = i;
                 newflag = 1;
                 break;
        case 7 :
        case 8 : update();              /* Update crun                          */
#if PERIM
                 updperv();
#endif
                 rs[++crun] = ++i;      /* Not end of line yet                  */
                 leftend = i;
                 newflag = 1;
                 break;
        case 9 : ++prun;
                 if (newflag)           /* End of a new run, allocate a new blob */
                 {
                   newflag = 0;
                   err = alloc();
                   if (err) exit(0);
                 }
                 else;
                 update();              /* Update crun                          */
#if PERIM
                 updperv();
                 updperh();
#endif
                 curreg = regabv;       /* This connects pixel A to D           */
                 getablob();            /* Get above run's blob number          */
                 rs[++crun] = ++i;      /* Not end of line yet                  */
                 leftend = i;
                 break;
       default : break;
     }
     if (err) return(1);
     else;
   }

   if (crun >= MAXRUN)
   {
     m_errmsg("Run list overflow.");
     return(1);
   }
   else
     ++crp;
 }
 return(0);
}
```

```
/*****
Function : getablob()

Description : This function searches through the blob/run tracking list for an active blob record
              and returns the active blob associated with the variable prun.
              MSB = 0 indicates an active record
              MSB = 1 indicates that the blob has been merged and released

Argument list : void

Return value : void
*****/
void getablob()
{
   unsigned int n = rn[prun] & 0x0000FFFF;

   while ((n = bp[n]) & 0x80000000) n &= 0x0000FFFF;
   regabv = n;
}

/*****
Function : update()

Description : This function updates the statistics for a completed run.

Argument list : void

Return value : void
*****/
void update()
{
  int il = rs[crun];              /* Starting column of current run                    */
  unsigned long tsum;
  unsigned long tsumi;
  unsigned long tsumi2;

  rn[crun] = b[curreg].number;    /* Set the run number to the allocated blob number   */

#if AREA
  tsum = (proma[i]-promb[il]);
  bs[curreg].area +=  tsum;
#endif

#if N1MOM
  bs[curreg].sumi +=  (tsumi = (promc[i]-promd[il]));
  bs[curreg].sumj +=  tsum * j;
#endif

#if N2MOM
  bs[curreg].sumi2 += (tsumi2 = (prome[i]-promf[il]));
  bs[curreg].sumj2 += promg[j] * tsum;
  bs[curreg].sumij += (float)((unsigned long)j * (unsigned long)tsumi);
#endif

#if ENCLBOX
  if (il < b[curreg].imin) b[curreg].imin = il;
  else;
  if (i > b[curreg].imax) b[curreg].imax = i;
  else;
  b[curreg].jmax = j;
#endif

#if BORD
  b[curreg].xbord = i;
#endif
}
```

```
/*****
Function : alloc()

Description : This function allocates and initialises a new blob.

Argument list : void

Return value : int 0 : new blob allocated OK.
                   1 : error allocating new blob.
*****/
int alloc()
{
  int tmp;

  if (avail < 0)                /* No freed up merged blobs to re-use                  */
  {
    if (qfree >= MAXBLOB)
    {
      m_errmsg("All blobs used.");
      return(1);
    }
    else
      curreg = qfree++;         /* Set curreg to next available blob                   */
  }
  else                          /* Re-use blob indicated by avail                      */
  {
    curreg = avail;
    avail = b[curreg].flink;    /* Chain to next blob                                  */
  }

  if (bn >= MAXNUM)             /* This initializes the data structure for a new blob  */
  {
    m_errmsg("Blob chain index list overflow.");
    return(1);
  }
  else;
  bp[bn] = curreg;              /* Set tracking memory to current blob                 */
  b[curreg].number = bn;        /* Give blob next sequential tracking number           */
  ++bn;                         /* Increment tracking number */

#if TREE
  b[curreg].parent = -1;        /* Parent unknown                                      */
  b[curreg].child = -1;         /* Child unknown                                       */
  b[curreg].sibling = -1;       /* Sibling unknown                                     */
#endif

  b[curreg].color = ((twobysix>>1)&1);    /* Set color to match run                    */
  bs[curreg].area = 0.0;                  /* No area yet                               */

#if PERIM
  b[curreg].perim = 0;          /* No perimeter yet                                    */
#endif

  b[curreg].nholes = 0;         /* No holes yet                                        */
  bs[curreg].sumi = 0;
  bs[curreg].sumj = 0;

#if N2MOM
  bs[curreg].sumi2 = 0;
  bs[curreg].sumij = 0;
  bs[curreg].sumj2 = 0;
#endif

#if ENCLBOX
  b[curreg].imin = rs[crun];
  b[curreg].imax = i;
  b[curreg].jmin = j;
  b[curreg].jmax = j;
#endif

#if BORD
  b[curreg].xbord = i;
#endif
```

```
  tmp = b[regabv].flink;          /* Plug new blob into linked list                */
  b[regabv].flink = curreg;
  b[curreg].flink = tmp;
  b[tmp].rlink = curreg;
  b[curreg].rlink = regabv;
  return(0);
}

/*****
Function : surround()

Description : This function checks to see if a blob has closed on itself.

Argument list : int *backblob     Background or root blob.

Return value : void
*****/
void surround(backblob)
int *backblob;
{
  int length = (i - leftend)<<2; /* 4/3                                          */
  int tmp = regabv;              /* Temp hole register set to pixel D's run #    */
  int tmp2;

#if PERFLAG
  length +=4;
#endif

#if PERIM
  b[tmp].perim += length;
#endif

  getablob();                    /* Get above run's blob number                  */
  if (newflag)                   /* What was originally thought to be a new blob */
  {                              /* is merged into an existing blob (pixel B's)  */
    newflag = 0;
    curreg = regabv;

#if PERIM
    b[curreg].perim += length;   /* Add new blob's run length to perim           */
#endif
  }
  else if (curreg == regabv)     /* Background has surrounded pixel D            */
  {
    b[tmp].sibling = b[curreg].child;     /* Pixel D's sibling is at the same level as      */
    b[tmp].parent = curreg;               /* current child, it's parent must be the current */
    b[curreg].child = tmp;                /* blob and this blobs child must be the new      */
    ++b[curreg].nholes;                   /* found hole. Add one to this blob's hole amount.*/

#if PERIM
    b[curreg].perim -= b[tmp].perim;      /* Subtract the holes perimeter from the blob's   */
#endif

    if (b[curreg].parent<0)
      *backblob = curreg;
    else;
  }
  else                                    /* Merge pixel B's blob into current blob         */
  {
#if PERIM
     b[curreg].perim += (b[regabv].perim + length);       /* Add blob's perimeter        */
#endif

#if ENCLBOX
      if (b[regabv].imin<b[curreg].imin) b[curreg].imin = b[regabv].imin;
      else;
      if (b[regabv].imax>b[curreg].imax) b[curreg].imax = b[regabv].imax;
      else;
      if (b[regabv].jmin<b[curreg].jmin) b[curreg].jmin = b[regabv].jmin;
      else;
#endif
```

```
#if AREA
      bs[curreg].area += bs[regabv].area;
#endif

#if N1MOM
      bs[curreg].sumi += bs[regabv].sumi;
      bs[curreg].sumj += bs[regabv].sumj;
#endif

#if N2MOM
      bs[curreg].sumi2 += bs[regabv].sumi2;
      bs[curreg].sumij += bs[regabv].sumij;
      bs[curreg].sumj2 += bs[regabv].sumj2;
#endif

      if (b[regabv].nholes)              /* Any holes in the blob?                          */
      {
         b[curreg].nholes += b[regabv].nholes;
         tmp = b[regabv].child;
         while (tmp > 0)
         {
           b[tmp].parent = curreg;
           tmp2 = tmp;
           tmp = b[tmp2].sibling;
         }
         b[tmp2].sibling = b[curreg].child;
         b[curreg].child = b[regabv].child;
      }
      else;
      b[b[regabv].rlink].flink = b[regabv].flink; /* Skip over merged blob by setting its   */
                                                  /* back link to its forward               */
      b[b[regabv].flink].rlink = b[regabv].rlink; /* Also have to set the new forward's     */
                                                  /* back link to skip back over merged     */
                                                  /* blob.                                  */

      /* Indicate that blob has been freed*/
      bp[b[regabv].number & 0x0000FFFF] = b[curreg].number | 0x80000000;
      /* Point the forward link at the next available blob */
      b[regabv].flink = avail;
      b[regabv].child = -1;
      b[regabv].parent = -1;
      avail = regabv;             /* regabv was freed so it's now available for re-use      */
      regabv = curreg;
  }
}

/*****
Function : updperv()

Description : Function that updates the vertical component of the perimeter of a blob.

Argument list : void

Return value : void
*****/
void updperv()
{
#if PERFLAG
   b[curreg].perim += 8;                          /* Crack mode                             */
#else
   b[curreg].perim += (rptrans[twobysix] + dval); /* Add right perimeter                    */
   dval = lptrans[twobysix];                      /* Save new left perimeter                */
#endif
}
```

```
/*****
Function : updperh()

Description : Function that updates the horizontal component of the perimeter of a blob.

Argument list : void

Return value : void
*****/
void updperh()
{
   int length = (i - leftend)<<2;

#if PERFLAG
   length += 4;
#endif

   b[curreg].perim += length;
   b[regabv].perim += length;
}

/*****
Function : m_errmsg()

Description : Function that prints an error message on the screen.

Argument list : char *str        error message

Return value : void
*****/
void m_errmsg(str)
char *str;
{
   printf("%s\n",str);
}

/*****
Function : characterise()

Description : This function calculates the parameters used to characterise the blobs in segmented surface
              froth structures and saves the analysis results, in ASCII format, to a file on disk. It also
              marks the centroids of the detected blobs.

Argument list : BYTE image[][]    analyzed segmented binary image
                int backblob      background or root blob
                long minarea      minimum blob area that is of interest
                long maxarea      maximum blob area that is of interest
                int x1, y1        top left-hand coordinates of AOI
                int x2, y2        bottom right-hand coordinates of AOI
                char *fn          filename to which the analysis results are to be written

Return value : void
*****/
void characterise(image,backblob,minarea,maxarea,x1,y1,x2,y2,fn)
BYTE image[IMAGESIZE][IMAGESIZE];
int backblob;
long minarea;
long maxarea;
int x1;
int y1;
int x2;
int y2;
char *fn;
{
  int totalblobs=0;              /* Total number of blobs, excluding the background blob   */
  int blobsinrange=0;            /* Number of blobs in the range minarea<=area<=maxarea    */
  int blobsbelow=0;              /* Number of blobs of area < minarea                      */
  int blobsabove=0;              /* Number of blobs of area > maxarea                      */
  int cb;                        /* Current blob                                           */
  float perimeter;               /* Blob perimeter                                         */

  float area_d[MAXBLOB];         /* Array for holding valid area data                      */
  float round_d[MAXBLOB];        /* Array for holding valid circularity data               */
  float eccent_d[MAXBLOB];       /* Array for holding valid elipticity data                */
```

```
int x_centroid[MAXBLOB];        /* Array for holding x centroid values                  */
int y_centroid[MAXBLOB];        /* Array for holding y centroid values                  */

struct stats area_s;            /* Area statistics                                      */
struct stats round_s;           /* Circularity statistics                               */
struct stats eccent_s;          /* Elipticity statistics                                */

float xmom, ymom, xymom, x2mom, y2mom;
float sumxyc=0.0, sumx2c=0.0, sumy2c=0.0;                    /* Central moments     */
float temp1, temp2, t1, t2;                                  /* Dummy variables     */
float major, minor;

FILE *fptr;
int i;

cb = backblob;                  /* Note that the background blob is included            */

do {
  if (b[cb].color == 0)          /* Calculate blob characteristics. Black blobs are only */
  {                             /* of interest                                          */
    totalblobs += 1;            /* Increment running blob total                         */

    /* Area */
    bg[cb].area = bs[cb].area;

    /* First and second order x and y moments */
    xmom = (float)bs[cb].sumi;
    ymom = (float)bs[cb].sumj;
    xymom = (float)bs[cb].sumij;
    x2mom = (float)bs[cb].sumi2;
    y2mom = (float)bs[cb].sumj2;

    /* Centralised second order x, y and xy moments */
    sumx2c = x2mom - ((xmom*xmom))/((float)(bg[cb].area));
    sumy2c = y2mom - ((ymom*ymom))/((float)(bg[cb].area));
    sumxyc = xymom - ((ymom*xmom))/((float)(bg[cb].area));

    /* Centroid calculations */
    bg[cb].xcent = (int)(xmom/bg[cb].area);
    bg[cb].ycent = (int)(ymom/bg[cb].area);

    /* Circularity calculations */
    /* If the aspect ratio is 1.0, the perimeter value must be divided by 3    */
    /* If the aspect ratio is 1.26, the perimeter must be multiplied by 0.3125 */
    /* The reasons for doing this are explained in the CONSOFT V4.0 Reference  */
    /* Manual.                                                                 */
    perimeter = (float)b[cb].perim/3;
    bg[cb].round = (SQUARE(perimeter))/(4.0*PI*(float)bg[cb].area);

    /* Elipticity calculations */
    temp1 = sumx2c + sumy2c;
    temp2 = sqrt(SQUARE(sumy2c-sumx2c)+4.0*SQUARE(sumxyc));
    t1 = (temp1+temp2)/2.0;
    t2 = fabs((temp1-temp2)/2.0);
    if (t1>t2)
    {
      major = t1;
      minor = t2;
    }
    else
    {
      major = t2;
      minor = t1;
    }
    bg[cb].eccent = sqrt(major/minor);

    if (INRANGE(minarea,bs[cb].area,maxarea))   /* Process all blobs within the required */
    {                                           /* range                                 */
      blobsinrange += 1;                        /* Another blob in the specified range   */

      area_d[blobsinrange] = (float)bg[cb].area;
      round_d[blobsinrange] = bg[cb].round;
      eccent_d[blobsinrange] = bg[cb].eccent;
      x_centroid[blobsinrange] = bg[cb].xcent;
```

```
        y_centroid[blobsinrange] = bg[cb].ycent;

        mark_centroid(image,bg[cb].xcent,bg[cb].ycent);
      }
      else if (bs[cb].area<minarea)
      {
        blobsbelow += 1;                        /* Blob smaller than minarea          */
      }
      else
      {
        blobsabove += 1;                        /* Blob larger than maxarea           */
      }
    }

    cb = b[cb].flink;                           /* Next blob                          */
  } while (cb != backblob);

  moment(area_d,blobsinrange,&area_s);          /* Calculate area statistics          */
  moment(round_d,blobsinrange,&round_s);        /* Calculate circularity statistics   */
  moment(eccent_d,blobsinrange,&eccent_s);      /* Calculate elipticity statistics    */

  if ((fptr = fopen(fn,"wt")) == NULL)
    m_errmsg("Error opening %s\n",fn);          /* Error opening the output file      */
  else
  {
    fprintf(fptr,"FILE : %s\n",fn);
    fprintf(fptr,"\n");
    fprintf(fptr,"Area of interest : (x1,y1) = (%3i,%3i)\n",x1,y1);
    fprintf(fptr,"                   (x2,y2) = (%3i,%3i)\n",x2,y2);
    fprintf(fptr,"Total number of blobs : %lu\n",totalblobs);
    fprintf(fptr,"Blobs in the range %lu <= area <= %lu : %lu\n",minarea,maxarea,blobsinrange);
    fprintf(fptr,"Blobs of area < %lu : %lu\n",minarea,blobsbelow);
    fprintf(fptr,"Blobs of area > %lu : %lu\n\n",maxarea,blobsabove);
    fprintf(fptr,"#\tAREA\t\tROUND\tECCENT\tX CENT\tY CENT\n");
    for (i=1;i<=blobsinrange;i++)
    {
      fprintf(fptr,"%4i\t",i);
      fprintf(fptr,"%6.2f\t\t",area_d[i]);
      fprintf(fptr,"%2.2f\t",round_d[i]);
      fprintf(fptr,"%2.2f\t",eccent_d[i]);
      fprintf(fptr,"%3i\t",x_centroid[i]);
      fprintf(fptr,"%3i",y_centroid[i]);
      fprintf(fptr,"\n");
    }
    fprintf(fptr,"\n");

    fprintf(fptr,"mean :");
    fprintf(fptr,"\t%6.2f\t\t%2.2f\t%2.2f\n",area_s.mean,round_s.mean,eccent_s.mean);
    fprintf(fptr,"var  :");
    fprintf(fptr,"\t%6.2f\t%2.2f\t%2.2f\n",area_s.var,round_s.var,eccent_s.var);
    fprintf(fptr,"sdev.:");
    fprintf(fptr,"\t%6.2f\t\t%2.2f\t%2.2f\n",area_s.stddev,round_s.stddev,eccent_s.stddev);
    fprintf(fptr,"skew :");
    fprintf(fptr,"\t%6.2f\t\t%2.2f\t%2.2f\n",area_s.skew,round_s.skew,eccent_s.skew);
    fclose(fptr);
  }
}

/*****
Function : moment()

Description : This function calculates the mean, variance, standard deviation and skewness for an array of
              data.

Argument list : float data[]      array of data to be processed
                int n             number of data points
                struct stats *sm  structure of statistical measurements

Return value : void
*****/
void moment(data,n,sm)
float data[MAXBLOB];
int n;
struct stats *sm;
```

```
{
  int j;
  float s, p;

  if (n<=1)
  {
     m_errmsg("Number of points must be at least 2 for moment calculations.\n");
     exit(0);
  }

  s = 0.0;
  for (j=1;j<=n;j++)                    /* First pass to get the mean                    */
     s += data[j];
  sm->mean = s/n;

  sm->var = sm->stddev = sm->skew = 0;
  for (j=1;j<=n;j++)                    /* Second pass to get the first, second and      */
  {                                     /* third moments of the deviation from the mean  */
    sm->var += (p = s*s);
    sm->skew += (p *= s);
  }

  sm->var /= (n-1);                     /* Put it all together                           */
  sm->stddev = sqrt(sm->var);
  if (sm->var)
    sm->skew /= (n*(sm->var)*(sm->stddev));
}

/*****
Function : mark_centroid()

Description : Marks the centroid positions of the detected blobs on binary image.

Argument list : BYTE image[][]    image to be marked
                int x             column coordinate
                int y             row coordinate

Return value : void
*****/
void mark_centroid(image,x,y)
BYTE image[IMAGESIZE][IMAGESIZE];
int x;
int y;
{
  int i;

  for (i=-2;i<=2;i++)
  {
    image[y][x+i] = 1;
    image[y+i][x] = 1;
  }
}
```