

A Thesis Submitted for the Degree of PhD at the University of Warwick

Permanent WRAP URL:

<http://wrap.warwick.ac.uk/108898>

Copyright and reuse:

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it.

Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk

THE BRITISH LIBRARY
BRITISH THESIS SERVICE

Visual Inspection:

TITLE Image Sampling, Algorithms and Architectures.

AUTHOR Richard C. Staunton

DEGREE

AWARDING BODY The University of Warwick

DATE 1991

THESIS
NUMBER

THIS THESIS HAS BEEN MICROFILMED EXACTLY AS RECEIVED

The quality of this reproduction is dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction.

Some pages may have indistinct print, especially if the original papers were poorly produced or if the awarding body sent an inferior copy.

If pages are missing, please contact the awarding body which granted the degree.

Previously copyrighted materials (journal articles, published texts, etc.) are not filmed.

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no information derived from it may be published without the author's prior written consent.

Reproduction of this thesis, other than as permitted under the United Kingdom Copyright Designs and Patents Act 1988, or under specific agreement with the copyright holder, is prohibited.

1	2	3	4	5	6	REDUCTION X	21	
cm						CAMERA	1	
							No. of pages	

**Visual Inspection:
Image Sampling, Algorithms and Architectures.**

by
Richard C. Staunton

submitted for the degree of
Doctor of Philosophy

to
The University of Warwick
Department of Engineering

November, 1991.



Contents

Title page.	
Contents.	i
List of figures.	x
List of tables.	xviii
Acknowledgements.	xix
Declaration.	xx
Abstract.	xxi
Abbreviations.	xxii
1 Introduction.	1
1.1 Vision and Hexagonal Sampling Systems.	1
1.2 Computer Architectures for Industrial Image Processing.	4
1.3 The Structure of the Thesis.	7
2 Human Vision, Computer Vision, and Industrial Inspection.	11
2.1 Introduction.	11
2.2 Human Vision.	12
2.2.1 The Historical Background of the Study of Human Vision.	12
2.2.2 The Hexagonal Packing of Sensory Elements in the Eye.	12
2.2.3 Models of Human Vision.	14

2.2.3.1	The Mapping of Visual processes onto the Biological Visual system	16
2.3	Computer and Machine Vision	19
2.3.1	General Purpose Systems	19
2.3.2	Machine Vision	22
2.3.2.1	Robot Vision	22
2.3.2.2	Visual Inspection	23
2.3.2.3	Other Applications	27
2.4	Industrial Inspection, Engineering Considerations	27
2.4.1	Reliability, Processing Rate, and Cost Effectiveness	27
2.4.2	Automated Surface Inspection	29
2.4.2.1	Surface Defects and Finish Quality	30
2.4.2.2	Handwritten and Printed Character Reading	31
2.5	Conclusions	33
3	Image Sampling	35
3.1	Sampling Theory	35
3.2	Sampling Schemes	37
3.2.1	The Square Scheme	37
3.2.2	The Rectangular Scheme	39
3.2.3	The Hexagonal Scheme	40
3.2.4	Other Schemes	43
3.3	Image Sensors	44
3.3.1	TV Cameras	44
3.3.1.1	Raster Scanned Devices and Image Sampling	44
3.3.1.2	Camera Technology and Image Sampling	46

3.4	Image Storage and Display.	47
3.4.1	Memory Size.	47
3.4.2	Addressing of Memory in the Rectangular and Hexagonal Systems.	47
3.4.3	Computer Graphics.	50
3.5	Conclusions.	51
4	Image Processing.	54
4.1	Introduction.	54
4.2	High, Mid and Low Level Processes.	55
4.3	Global Low Level Processes.	55
4.3.1	General.	55
4.3.2	FFT of Square and Hexagonally Sampled Data.	57
4.3.3	A Comparison of Filters Operating in Each System.	57
4.4	Local Processes.	58
4.4.1	Processes Implemented on Square and Hexagonal Systems.	58
4.4.1.1	Connectivity.	58
4.4.1.2	Filters.	59
4.4.1.3	Edge Detectors.	65
4.4.1.4	Line Thinning and the Skeleton of an Object.	69
4.4.1.5	Morphological Operators.	71
4.4.1.6	Distance Functions.	74
4.4.2	Processes Only Implemented on the Square System.	74
4.4.2.1	Advanced Edge Detectors.	74
4.5	Mid Level Processes.	77
4.5.1	The Hough Transform.	77

4.6	Conclusions.	79
5	A Comparison Between Local Edge Detection Operators in Square and Hexagonal Data Structures.	83
5.1	System Implementation.	83
5.1.1	Overall Computational Processing Time.	85
5.1.2	Calculation of the Edge Operator Mask Weights.	86
5.1.2.1	Square Grid System.	86
5.1.2.2	Hexagonal Grid System.	89
5.1.2.3	Summary of Edge Detector Mask Weights.	91
5.1.3	Edge Accuracy.	91
5.1.4	Circular Band-Limiting of the Modeled Edge Data.	94
5.1.5	Edge Position Within a Pixel.	101
5.2	Conclusions.	105
6	Case Studies: Comparison of Square and Hexagonal System Algorithms.	106
6.1	The Simulation of Hexagonally Sampled Images.	107
6.2	Surface Defect Detection in Sand Cores used for Automobile Engine Castings.	107
6.2.1	Defect Modeling.	113
6.2.1.1	Circular Defects.	117
6.2.1.2	Rectangular Defects.	123
6.2.1.3	Long Thin Defects.	126
6.2.2	Alternative Methods.	128
6.2.3	Sand Core Study Conclusions.	129
6.3	Printed Character Recognition.	131
6.3.1	Printed Character Study Conclusions.	142

6.4	Final Conclusions.	142
7	Computer Architectures for Machine Vision.	143
7.1	Introduction.	143
7.2	The Parallel Processing of Images.	144
7.3	The Classification of Processors.	145
7.4	Two and Multi-Dimensional Processor Arrays.	147
7.4.1	Introduction.	147
7.4.2	Fine Grain Arrays.	149
7.4.3	Coarse Grain Arrays.	152
7.4.4	Pyramid Processors.	153
7.5	One Dimensional Arrays.	155
7.5.1	Vector Arrays.	156
7.5.2	Pipelined Processors.	158
7.5.2.1	Real-Time Pipelines.	160
7.5.2.2	Re-Circulating Pipelines.	163
7.6	The Choice of Architecture for an Industrial Vision System.	165
7.6.1	Introduction	165
7.6.2	Real-Time Processing.	168
7.6.3	The Reliability of the Image Processes.	169
7.7	Conclusions.	171
8	A Pipeline Processor Element.	173
8.1	Introduction.	173
8.2	A Basic PE Design.	174
8.2.1	The PE External Interconnections.	174

8.2.2	Processor Features.	176
8.2.2.1	Real-Time Processing.	176
8.2.2.2	Image Processes to be Implemented.	177
8.2.3	A Functional Description of the PE Hardware.	180
8.2.3.1	System Overview.	180
8.2.3.2	The Local Area, Line Delays, and Associated Modules.	181
8.2.3.3	The Multiplier Arrays.	184
8.2.3.4	The Selectable Operator.	185
8.2.3.5	The Output Post Processes.	198
8.2.3.6	The Control Unit.	199
8.3	Possible Options and Modifications That Enhance the Basic PE.	203
8.3.1	The Processing of Hexagonally Sampled Images.	203
8.3.1.1	Pipeline System Modification.	203
8.3.1.2	PE Modification.	204
8.3.1.3	Comment.	206
8.3.2	Alternative Circuits to Reduce the Number of Multipliers in the Multiplier Arrays.	207
8.3.2.1	Separable Operators.	207
8.3.2.2	Simplified Multiplier Array (B).	209
8.3.3	5x5 Pixel Local Image Areas.	209
8.3.3.1	In General.	209
8.3.3.2	A 5x5 Local Image Area for Parallel Binary Operations Within a Normal 3x3 PE.	210
8.4	Image Processing Pipeline Systems Using the Basic PE.	210
8.4.1	A Flexible Programmable Video Rate Pipeline.	210

8.4.2	Topological Interconnection Possibilities	211
8.4.3	Towards Adaptive Control	212
8.5	Conclusions	214
9	The Simulation of the Pipeline Processor Element	216
9.1	Introduction	216
9.2	The Simulation of Complete Pipelines	217
9.3	The Processor Element Simulation	218
9.3.1	Data Input	219
9.3.2	The Main Processes	220
9.3.2.1	The Scaling Routines	222
9.3.3	The Post Processes	223
9.3.4	Data Output	224
9.4	The Simulation Results	225
9.4.1	Convolution Operators	225
9.4.1.1	Square Sampled Images	226
9.4.1.2	Hexagonally Sampled Images	232
9.4.2	Edge Detection Operators	233
9.4.2.1	Square Sampled Images	234
9.4.2.2	Hexagonally Sampled Images	237
9.4.2.3	An Alternative Thresholding Strategy	240
9.4.2.4	Discussion	241
9.4.3	Sort and Select Operators	242
9.4.3.1	Median Filters	243
9.4.3.2	Grey Level Morphology	244

9.4.3.3	Discussion	245
9.4.4	Parallel Logical Operators	246
9.4.4.1	Binary Morphological Operations	246
9.4.4.2	Thinning Operations	247
9.5	Conclusions	250
10	Conclusions and Further Work	254
10.1	Conclusions	254
10.1.1	The Hexagonal Sampling of Images and Their Processing	254
10.1.1.1	Sampling	254
10.1.1.2	Processing	256
10.1.2	Image Processing in Industrial or Controlled Lighting Environments	257
10.1.2.1	Image Processing Operators	258
10.1.2.2	Surface Defect Detection In Sand Cores	258
10.1.2.3	Printed Character Processing	259
10.1.3	Computer Architecture	260
10.1.3.1	Pipeline Architectures	260
10.1.3.2	The Design of the PE	262
10.2	Further Work	264
10.2.1	Computer Vision and Hexagonal Sampling Grids	264
10.2.1.1	Operators Requiring Large Areas of Support	264
10.2.1.2	The Design and Response of Sensor Elements	265
10.2.1.3	Additional Case Studies	265
10.2.1.4	The Adoption of Hexagonal Image Sampling	266
10.2.2	Pipeline Processors	266

10.2.2.1	The Completion of the PE Design and Fabrication.	266
10.2.2.2	The Host Computer.	267
10.2.2.3	Adaptive Pipeline Control.	268
10.3	In Summary.	269
	Bibliography.	270

List of Figures

1.1	A Simple Pipeline Processor.	6
2.1	The Retina.	13
2.2	The Wavelike Appearance of Parallel Lines when Viewed Close to the Eyes Resolution Limit and the Hexagonal Sensor Pattern that Produces this Effect.	14
2.3	Tiling with Hexagonally Shaped Local Areas, a Seven Element Local Area That Produces one Value in the Reduced Resolution Image is Shaded.	17
2.4	The Hexagonal Pyramid Structure.	18
3.1	A Square Grid of Sampling Points ** , The Area is Tiled with Square Shaped Pixels.	38
3.2	A Tiling of the Frequency Plane with Square Band Limit Regions.	39
3.3	A Rectangular Grid of Sampling Points ** , The Area is Tiled with Rectangular Shaped Pixels.	40
3.4	Square and Rectangular Band Regions Containing Maximal Circular Band Regions.	41
3.5	A Hexagonal Grid of Sampling Points ** , The Area is Tiled with Rectangular Shaped Pixels.	41
3.6	A Hexagonal Grid of Sampling Points ** , The Area is Tiled with Hexagon Shaped Pixels.	42
3.7	A Hexagonal Band Region Containing A Maximal Circular Band Region.	43
3.8	The Array Positions of the 6 Hexagonal Nearest Neighbours. (a) The Logical Positions. (b) The Odd Row Array Positions, (c) The Even Row Array Positions.	48
3.9	Hexagonal Pixel and Array Subscript Mapping, Scheme 1.	49
3.10	Hexagonal Pixel and Array Subscript Mapping, Scheme 2.	49
4.1	3 Shell Hexagonal and 6 Shell Square Local operators.	60

4.2	3x3 Sobel Differential Operators.	66
4.3	Hexagonal Differential Edge Detection Operators.	68
5.1	Circular Neighbourhoods of Radius 1.500 and 1.581 Imposed on 3x3 Square Pixel Areas.	87
5.2	A Circular Neighbourhood Imposed on a Seven Element Hexagonal Grid Pixel Area.	89
5.3	Variation in the Magnitude of the Maximum Angular Error as a Function of the Radius of the Circular Neighbourhood.	90
5.4	Operator Edge Gradient Response, for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$	93
5.5	Operator Angular Error for Edges Oriented $0 \Rightarrow 90^\circ$	93
5.6	A Step Edge of Height 10 Units. (A) Profile. (B) The 2-D Image Data Resulting from Such an Edge, Oriented at 135° , Passing Through a 3x3 Pixel Area.	95
5.7	Circular Spatial Region, Diameter 1.00 Times Square Grid Horizontal Sampling Distance. Operator Edge Gradient Response, for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$	96
5.8	Circular Spatial Region, Diameter 1.00 Times Square Grid Horizontal Sampling Distance. Operator Angular Error for Edges Oriented $0 \Rightarrow 90^\circ$	96
5.9	Circular Spatial Region, Diameter 1.50 Times Square Grid Horizontal Sampling Distance. Operator Edge Gradient Response, for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$	98
5.10	Circular Spatial Region, Diameter 1.50 Times Square Grid Horizontal Sampling Distance. Operator Angular Error for Edges Oriented $0 \Rightarrow 90^\circ$	98
5.11	Circular Spatial Region, Diameter 2.00 Times Square Grid Horizontal Sampling Distance. Operator Edge Gradient Response, for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$	99

5.12	Circular Spatial Region, Diameter 2.00 Times Square Grid Horizontal Sampling Distance. Operator Angular Error for Edges Oriented $0 \Rightarrow 90^\circ$.	99
5.13	Sobel Operator Edge Gradient Responses, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Square Sampling Point Support Region.	102
5.14	Sobel Operator Edge Orientation Errors, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Square Sampling Point Support Region.	102
5.15	Hexagonal Operator Edge Gradient Responses, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Rectangular Sampling Point Support Region.	103
5.16	Hexagonal Operator Edge Orientation Errors, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Rectangular Sampling Point Support Region.	103
5.17	Hexagonal Operator Edge Gradient Responses, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Circular Sampling Point Support Region, Diameter 1.5 units.	104
5.18	Hexagonal Operator Edge Orientation Errors, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Circular Sampling Point Support Region, Diameter 1.5 units.	104
6.1	Rectangular Sampled Sand Core Image, 512x512 Resolution.	108
6.2	Rectangular Sampled Sand Core Image Zoomed, 64x64 Resolution.	109
6.3	Hexagonal Sampled Sand Core Image Zoomed, 64x64 Resolution.	109
6.4	Rectangular Sampled Sand Core Image Edge Detected, 64x64 Resolution.	111

6.5 Hexagonal Sampled Sand Core Image Edge Detected, 64x64 Resolution.	111
6.6 Rectangular Sampled Sand Core Image, Thinned, 64x64 Resolution.	112
6.7 Hexagonal Sampled Sand Core Image, Thinned, 64x64 Resolution.	112
6.8 A Simple Circular Defect Model.	115
6.9 A Simple Circular Defect Model with Noise Added and Edge Smoothing.	116
6.10 A 3D Plot of the Sand Core Large Circular Defect Image.	118
6.11 The Threshold Values for Circles, Diameter 6.0 Units, Centred at Various Positions.	119
6.12 The Threshold Values for Circles of Various Diameter, Centred at 7.2,8.0	119
6.13 The Threshold Values for Smoothed Circles, Diameter 6.0 Units, Centred at Various Positions.	121
6.14 The Threshold Values for Smoothed Circles of Various Diameter, Centred at 7.2,8.0.	121
6.15 The Threshold Values for 5x5 Unsmoothed Rectangles with no Added Noise.	123
6.16 The Threshold Values for 5x5 Smoothed Rectangles with no Added Noise.	125
6.17 Modelled Rectangular Defect, Square Sampled. Size: 8x2 Pixels, Major Axis at 60 Degrees to the Horizontal, and Noise Added with a Variance of 400. (a) Grey-level Image. (b) Edge Detected Image.	127
6.18 Modelled Rectangular Defect, Hexagonal Sampled. Size: Equal to the Correspond- ing Square Sampled Defect, and Noise Added with a Variance of 400. (a) Grey-level Image. (b) Edge Detected Image.	127
6.19 Isolation of Object Edges Using a Hough Transform Technique.	129
6.20 Thresholded Rectangular System Text.	132
6.21 Thresholded Hexagonal System Text.	132
6.22 (a) Chin's Thinning Templates. (b) Chin's Restoring Templates.	133
6.23 Rectangular System Text Thinned Without Morphological Smoothing.	133
6.24 The Square 3x3 Structuring Element.	134

6.25	Rectangular System Text, Morphologically Smoothed.	134
6.26	Rectangular System Text, Morphologically Smoothed and Thinned.	134
6.27	The Hexagonal Structuring Element.	135
6.28	Hexagonal Thinning and Restoring Templates.	136
6.29	Hexagonal System Text Thinned Without Morphological Smoothing.	136
6.30	Hexagonal System Text Thinned Without Morphological Smoothing, but Utilising Simplified Restoring Templates.	137
6.31	The Letter "E" with Superimposed Skeletons. (a) Square Sampled. (b) Hexagonally Sampled.	139
7.1	A MISD Pipeline.	146
7.2	2-D Processor Array Interconnection Topology. (a) Rectangular. (b) Hexagonal.	148
7.3	A Fine Grain Array System.	149
7.4	A Pyramid Architecture.	153
7.5	Hartman and Tanimoto's Pyramid Structure.	155
7.6	A Typical 1-D Systolic Array.	156
7.7	A Typical Pipeline Processor.	156
7.8	A Clip7A Linear Array Operating on Two Sub-images.	157
7.9	A Pipeline Processor Element.	159
7.10	A Real-Time Pipelined Image Processor.	160
7.11	A Re-Circulating Pipeline System.	163
8.1	A Simple Pipeline Processor Consisting of two PEs.	174
8.2	PE Input and Output Signals.	175
8.3	A Block Diagram of the Pipeline Processor Element.	181
8.4	The PE Input Section, Line Delays and Local Area.	182

8.5	The Multiplier Arrays	185
8.6	The Selectable Operators	186
8.7	The Convolution Operator Module	187
8.8	The Edge Detection Selectable Operator	189
8.9	Danielsson's Circuit to Perform the Rank Ordering Algorithm	194
8.10	Danielsson's Logic Cell	195
8.11	The Corrected Logic Cell	195
8.12	A Corrected Stage of the Circuit to Perform the Rank Order Algorithm	196
8.13	The Parallel Binary Operator	197
8.14	The Output Post Processes	199
8.15	The Control Unit	200
8.16	A to D Converter Sample and Data Read Timing	204
8.17	The Relationship Between the 9 Input Pixels and the 6 Hexagonal Nearest Neighbours on Alternate Scan Lines	205
8.18	The Alternative Convolution Module Circuit	208
8.19	A Programmable Video Rate Pipeline	211
8.20	A Programmable Re-circulating Pipeline	211
8.21	A Multi-pipeline Re-circulating System	212
8.22	A Multi-pipeline Arrangement to Produce an Edge Overlay on an Image	213
9.1	64 x 64 image, Unsmoothed and Smoothed by G_1 with Distributed Scaling	227
9.2	The Low Contrast Test Image and the Brightness Values in the Vicinity of one of the Corners	228
9.3	The Application of G_1 and G_2 to the Image Without Distributed Scaling	228
9.4	The Application of G_1 and G_2 to the Image With Distributed Scaling	229

9.5	Edge Detected Rectangular Image, (b) High Threshold, (b) Low Threshold.	229
9.6	(a) The Low Contrast Test Image (b) After High Pass Filtering (c) Numerical Representation, No Scaling (d) Numerical Representation After Distributed Scaling and a Word Width Limit of 8 Bits.	231
9.7	Hexagonally Sampled Sand Core Image, Unsmoothed and Smoothed by G_4 with Distributed Scaling.	233
9.8	The Application of G_4 to the Hexagonally Sampled Low Contrast Rectangle Image, Without and With Distributed Scaling.	233
9.9	(a) The Low Contrast Rectangle Image. (b) Edge Image with High Threshold Level (c) Edge Image with High Threshold Level and Distributed Scaling (d) Edge Image with Low Threshold Level (e) Edge Image with Low Threshold Level and Distributed Scaling.	235
9.10	Square Operator Edge Gradient Response for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$	236
9.11	High Pass Edge Filtered Version of the Low Contrast Rectangle.	237
9.12	(a) The Low Contrast Rectangle Image. (b) Edge Image with High Threshold Level (c) Edge Image with High Threshold Level and Distributed Scaling (d) Edge Image with Low Threshold Level (e) Edge Image with Low Threshold Level and Distributed Scaling.	238
9.13	Hexagonal Operator Edge Gradient Response for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$	240
9.14	A Typical Gradient Magnitude Squared to Threshold Transformation.	241
9.15	An Edge Detected Low Contrast Rectangle Image, Alternative Threshold Procedure. 241	
9.16	Square Sampled 64×64 image, Unsmoothed and Smoothed by a Square 3×3 Median Filter.	243

9.17 Hexagonally Sampled 64 x 64 image, Unsmoothed and Smoothed by a 7 Element Median Filter.	244
9.18 Square Sampled Five Item Input Image Together With Grey Level Morphologically Closed Version.	244
9.19 (a) Hexagonally Sampled Five Item Input Image. (b) Grey Level Morphologically Dilated Version. (c) Grey Level Morphologically Closed Version.	245
9.20 (a) Binary Image. (b) Morphologically Dilated. (c) Morphologically Closed. (d) Morphologically Erudd. (e) Morphologically Opened.	247
9.21 Edge Detected Sand Core Image Together With Thin-OPTA Image.	248
9.22 Original Text Image Together With Thin-OPTA Image.	248
9.23 Hexagonally Sampled Edge Detected Sand Core Image Together With Thinned Image.	249
9.24 Original Hexagonally Sampled Text Image Together With Thinned Image.	250

List of Tables

5.1	Edge Magnitude Computation Time for Each Pixel. PC Computer: Olivetti M28 with Intel 80286 Processor. Clock rate 8 MHz.	86
5.2	Edge Detector Mask Weights for a 3x3 Operator, radius $r = 1.500$	88
5.3	Edge Detector Mask Weights for a 3x3 Operator, radius $r = 1.460$	88
5.4	Edge Detector Mask Weights for a Seven Element Operator as Indicated by theCir- cularity Principle.	90
5.5	Edge Detector Mask Weights for a Seven Element Operator as Indicated by a Numerical Method.	91
5.6	Final Edge Detector Mask Weights for Both Systems.	91
6.1	Threshold Value Statistics for 50 Random Circular Defects.	122
6.2	Threshold Value Statistics for 50 Random Rectangular Defects.	126
6.3	Threshold Value Statistics for 25 Random Long Thin Defects.	128
6.4	A Comparison of 150 Small Characters.	138
6.5	A Comparison of 150 Large Characters.	139
6.6	Analysis of the Skeletal Lines of Character Strokes.	140
6.7	Analysis of Skeletal Junctions.	140
6.8	Analysis of Tight Skeletal Curves.	141
8.1	The Arguments to be Sorted, A_i , and the Corresponding Bit Vectors, B_j	191
8.2	The Rank Ordering Process.	191
8.3	Control Signals for a PE Capable of Processing Square and Hexagonally Sampled Images. The Word Widths Given are for a 3x3 Pixel Local Area.	202

Acknowledgements

I would especially like to thank my supervisor, Dr. N. Storey for his help and advice throughout the course of this work. I would also like to thank Professor W. C. Cullyer for reading the thesis and his many valuable comments on its presentation.

Thanks are also due to the many other members of the Department of Engineering who have added their encouragement, and provided much valuable advice. Particular thanks go to Professor D. K. Bowen, Dr S. S. Lawson, Dr H. V. Shurmer, Dr S. Summerfield, Dr T. Tjahjadi, and Mr W. Wong.

Last but not least, I am grateful to my wife, Julie, for her encouragement, understanding and support.

Declaration

None of the original work in this thesis was started before the 1st January 1986, when the author registered for this degree. The following is a list of publications that have been made on work contained within this thesis together with the numbers of the sections to which they relate.

1. Staunton, R. C. "The Design of Hexagonal Sampling Structures for Image Digitisation and Their use With Local Operators", *Image and Vision Computing*, Vol. 7, No. 3, Pages 162-166, 1989.

Relevant Thesis sections:- 3.2, 3.3, 4.3, 4.4, 5.1, 12.1.1.

2. Staunton, R. C. "Hexagonal Image Sampling a Practical Proposition", *Proc. SPIE*, Vol. 1008, Pages 23-27, 1989.

Relevant Thesis sections:- 4.3, 4.4, 5.1, 12.1.1.

3. Storey, N and Staunton, R. C. "A Pipeline Processor Employing Hexagonal Sampling for Surface Inspection", *3rd Int. Conf. On Image Processing and its Applications, IEE Conference Publication 307*, Pages 156-160, 1989.

Relevant Thesis sections:- 6.2, 8.3, 10.2, 12.1.2, 12.1.3.

4. Storey, N and Staunton, R. C. "An Adaptive Pipeline Processor for Real-time Image Processing", *Proc. SPIE*, Vol. 1197, Pages 238-246, 1990.

Relevant Thesis sections:- 8.2, 8.3, 10.2, 12.1.3.

5. Staunton, R. C and Storey, N. "A Comparison Between Square and Hexagonal Sampling Methods for Pipeline Image Processing", *Proc. SPIE*, Vol. 1194, Pages 142-151, 1990.

Relevant Thesis sections:- 6.1, 6.2, 9.3, 12.1.1, 12.1.2, 12.1.3.

Abstract

The thesis concerns the hexagonal sampling of images, the processing of industrially derived images, and the design of a novel processor element that can be assembled into pipelines to effect fast, economic and reliable processing.

A hexagonally sampled two dimensional image can require 13.4% fewer sampling points than a square sampled equivalent. The grid symmetry results in simpler processing operators that compute more efficiently than square grid operators. Computation savings approaching 44% are demonstrated. New hexagonal operators are reported including a Gaussian smoothing filter, a binary thinner, and an edge detector with comparable accuracy to that of the Sobel detector. The design of hexagonal arrays of sensors is considered.

Operators requiring small local areas of support are shown to be sufficient for processing controlled lighting and industrial images. Case studies show that small features in hexagonally processed images maintain their shape better, and that processes can tolerate a lower signal to noise ratio, than that for equivalent square processed images. The modelling of small defects in surfaces has been studied in depth.

The flexible programmable processor element can perform the low level local operators required for industrial image processing on both square and hexagonal grids. The element has been specified and simulated by a high level computer program. A fast communication channel allows for dynamic reprogramming by a control computer, and the video rate element can be assembled into various pipeline architectures, that may eventually be adaptively controlled.

Abbreviations

1D, 2D, 2.5D, 3D	One, two, two and a half, three dimensional
A-D	Analogue to digital converter
ALU	Arithmetic and logic unit
ASCII	American standard code for information interchange
ASIC	Application specific integrated circuit
CAD - CAM	Computer aided design - Computer aided manufacture
CCD	Charge coupled device
CMOS	Complementary metal oxide silicon
DDFOM	Defect detection figure of merit
FFT	Fast Fourier transform
FIR	Finite impulse response
HFFT	Hexagonal fast Fourier transform
MIMD	Multiple instruction stream, multiple data stream
MISD	Multiple instruction stream, single data stream
PCB	Printed circuit board
PE	Processor element
Pixel	Picture element
SIMD	Single instruction stream, multiple data stream
SISD	Single instruction stream, single data stream
VLSI	Very large scale integration

Chapter 1

Introduction.

This thesis is concerned with vision. Vision involves the acquisition and processing of information about an environment by sensing electromagnetic or other radiation, emitted or reflected by the features within the environment. Some aspects of human vision and the more general principles of computer vision are discussed, with a special emphasis on hexagonal image sampling. The main thrust of the thesis is, however, in the areas of machine and industrial vision. Algorithms are developed which compute hexagonally sampled images efficiently. They compare favourably to their square system counterparts. A pipeline processor is designed to process hexagonal and square sampled images, of the kind often encountered in industry, at high speed.

1.1 Vision and Hexagonal Sampling Systems.

The first task of a vision system is to sense the image of the scene about which information is to be gathered. Sensors can be designed to respond to various frequency bands. Often they convert the light intensity received into an electrical brightness signal. Two dimensional (2D) arrays of such sensor elements, or a single element or 1D array scanned throughout a sensing area, can be used to build up a picture of the scene. Such a picture is referred to as a 2D monochrome or grey level

image. A human observer can distinguish the boundaries between the objects within the image by the discontinuities in the brightness field that occur at them. Then, by using other clues such as the position of shadows, or prior knowledge, the observer can recognise and categorise the various objects depicted in the image. Computers can also be programmed to provide image processing and object recognition, but at present humans can cope with a far wider range of scene types, objects, and lighting conditions than any single computer vision system.

Further information can be obtained to ease the task of object recognition, but more information can mean that more processing is required. The 2D array of sensors can include sensors that respond maximally to differing frequencies in the visible spectrum, thus adding colour information. A sequence of images can be obtained over a time period, and from the relative positions of the edges of moving and stationary features, the objects recognised. The sensors can be moved to a new position, and a second image acquired. This can, with further processing, add depth or 2.5D information. A 2.5D image differs from a 3D image in that only the distances to the visible surfaces are known.

A 2D array of sensors samples the image. Each sensor, at an individual position in the array, provides information on that part of the image. The distance between two excited sensors within the array provides relative distance information. Sampling is achieved with single sensor systems and 1D arrays by moving them to a new position and then taking a new set of intensity measurements.

In real life images, it is only on rare occasions that only two sensors in the array will be excited coincidentally, and that all the required information on the scene can then be obtained. Various processes have to be performed on the grey level images to extract relevant information about, for example: brightness differentials, brightness constancy, or texture patterns. From this low level information, higher levels of knowledge on, for instance, object structure can be obtained. One of the topics explored in this thesis concerns the hexagonal arrangement, or packing, of sensors. With such an arrangement six neighbouring sensors are situated equidistantly from a central sensor, and

this pattern is then repeated throughout the sensing area. Previous researchers [124] have proved these hexagonal sampling grids to be the most efficient, in that fewer sensors are required to provide equal spatial resolution than with alternative grid patterns. An alternative, and more commonly used, grid pattern is the square sampling grid. Here horizontal rows and vertical columns of samples are assembled. Each sensor has four nearest neighbours, two horizontally and two vertically. Generally the square grid requires more sensors than an equivalent hexagonal grid to cover a given area if the resolution of the image is equal in all directions.

Previously a gap existed in the techniques available for image processing on the hexagonal grid in that processes were available only for images containing two brightness levels (binary images), and for general 2D signal processing. These general signal processing techniques included Fourier transforms, fast Fourier transforms, and filters, but the filters available were not necessarily optimum for image processing. A set of grey level image processing operators have been developed and are reported in the thesis. The set is not comprehensive as it was not the intention to produce a hexagonal grid operator equivalent to every existing square grid operator. The set is limited to the low level operators that perform the preliminary operations on the image data, as these are more dependent on the sampling grid geometry. It is also limited to grey level images and only operates within the plane of one image, as opposed to a sequence of images. A third restriction of the set is that it contains only small local area operators. The local area contains a region of samples surrounding a point of interest.

The use of small local area operators with the square grid system is well established. They have been found to be adequate for many image processing requirements where the image has been captured under controlled, or near perfect lighting conditions. It is often less expensive to create these conditions in an industrial situation than to provide more powerful computers to process the larger and more complicated operators required for poorly lit images. Industrial image processing has been reviewed and a set of widely differing operators chosen for implementation on both square

and hexagonal sampling grids. It will be shown that due to the higher symmetry of the hexagonal grid, a computational advantage can exist in addition to that gained because of the reduction in the number of sampling points required to cover the image.

For some operator types it was simply necessary to modify the existing square grid operators for the hexagonal grid, but for others a new design was required. A new design was required for a hexagonal edge (boundary) detector. The test results and comparison of this detector, with the square system equivalent, occupy the whole of Chapter 5 of the thesis.

In addition to comparisons between the individual operators for the two grid systems, some comparisons between processes operating on images captured in an industrial environment, and requiring the use of several operators have been undertaken. With these processes, not only is the efficiency compared, but also the overall quality of the process. It was found that for some processes fewer operators are required with the hexagonal system. The industrial examples compared include the detection of small surface defects and the processing of printed characters. Comments are made on the suitability of hexagonally sampled images for human viewing, and it is shown in Chapter 6, that in some circumstances it is easier to interpret them than an equivalent square sampled image.

Hexagonally sampled images have advantages for low level image processing in that fewer samples are required, the operators are simpler, and sometimes fewer operators are required. For industrial image processing, a system must be cost effective, as fast as the production line on which it is to be used, and reliable in that it must produce few incorrect decisions or measurements. It is in this area that the processing of hexagonally sampled images is likely to have most impact.

1.2 Computer Architectures for Industrial Image Processing.

Once a list of local image processing operators for use on hexagonal and square grids of sampling points has been defined, the next task is to identify a hardware system on which they can be

implemented. The operators are so simple that almost any modern digital computer can be used. It is the rate at which the images need to be processed for a particular application that primarily determines the choice of computer. Two application areas have been identified. In the first, images are processed at the rate at which they are captured, usually the video rate, and displayed for use by a human observer. In the second, applicable to machine vision, images are processed at a rate determined by the rate at which the scene changes. An industrial production line is an example of the second application area. If a machine vision system has to inspect parts as they pass along the line, then the processing on one part must be completed before the next comes into view. The processing rate is determined by the part rate.

The video rate is determined by the video standard in use. In Europe the usual image rate is 25 frames a second. With production lines, part rates can be several parts a second, but other applications may require an even faster update rate. However, the highest rate is likely to be the video rate, and the computer for a combined video and part rate processing system, must therefore be capable of processing 25 pictures in one second. Video rate processing is advantageous as it can be arranged that no frame store buffer memory is required between the TV camera and the computer.

The data rate at which the computer must operate is set by the number of picture elements (pixels) in the image and the frame rate. For a 2D sensor array the number of pixels is usually equal to the number of sensors. Images typically contain up to 512×512 pixels, although for some applications, considerably more. For a video rate application described in Chapter 8 of the thesis, the image data rate is $13.0 \text{ MBytes} \cdot \text{s}^{-1}$. Several operations are often required on each datum, and so processing rates can be multiples of the data rate. Other factors that determine the choice of computer are the cost and its flexibility of use.

For industrial applications the cost of the computer and its ability to process 512×512 images at the video rate are important. These two considerations can result in the use of parallel computers.

in which the task is divided up and shared by several processor elements (PEs). Parallel computer architectures are reviewed, and a pipeline architecture, in which the image data is streamed serially through a string of processors, has been chosen for further investigation. The pipeline processes local area operators efficiently, is fast enough for video rate calculations, and the individual PEs can be readily mass produced and assembled into inexpensive flexible computers.

A simple pipeline processor is shown in Figure 1.1. Such a pipeline is constructed by connecting a serial stream of image pixel data to the input of the left most PE. This provides the first processing operation of the task on each data, and then passes the partially processed data on to the next PE for further processing. This procedure continues until, at the end of the line, a completely processed image emerges. The value output by each PE is a function of each pixel value in the local area. The required local area is assembled within each PE as each pixel passes through. The parallelism is achieved by providing a separate PE for each image sub-process. Raster scanned image capture devices such as TV cameras produce a stream of data that, after digitisation, is compatible with the input to such a pipeline. Computer disk, computer memory, and frame store interfaces, can also readily provide a suitable input data stream to the pipeline. Many other parallel processor architectures require the image to be reformatted before it can be loaded into the processor elements.

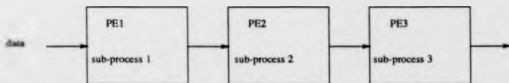


Figure 1.1: A Simple Pipeline Processor.

A single PE component that can be programmed to provide any of the local operators thought to be needed for controlled lighting applications has been designed as a part of this project. These

PEs can be assembled into pipelines constructed on single circuit boards, and configured by a host computer to provide many low level image processing tasks. The host computer can either accept data from the pipeline and continue with high level processing tasks, in which case the pipeline acts as a pre-processor, or it can direct the output to a TV monitor for observation.

A fast control communication channel between the PEs and the host computer enables the PEs to be reprogrammed during the inter-image period, and allows control information to be fed back from the PEs to the host computer. As discussed in Section 8.4.3, this will enable the adaptive control of pipelines in future developments. By making the delay in the passage of data through each PE independent of the image process, the branching and merging of pipelines, or the use of parallel lines can be achieved. These and other pipeline connection topologies are explored in Chapter 8 of the thesis.

The PE, specified in Chapter 8, will process both hexagonal and square sampled data. This dual standard device has been simulated and the results of the simulation reported in Chapter 9. Comparisons are made between results obtained from the simulated PE operating on equivalent images in both systems. The advantages and disadvantages of hexagonal processing are stated.

1.3 The Structure of the Thesis.

Vision and hexagonal sampling systems are explored in Chapters 2 to 6, and computer architectures for industrial image processing in Chapters 7 to 9. The overall conclusions reached during the project are presented in Chapter 10. A summary of the contents of each chapter follows.

Chapter 2: Human Vision, Computer Vision, and Industrial Inspection. The general areas of biological, computer and industrial vision are surveyed. Comparisons are made between the complexity of processes, the definition of real-time processing, the image types that require processing, and the adaptability of systems, in each area.

Chapter 3: Image Sampling. Initially the general aspects of image sampling are discussed. These include the sampling theory, and quantisation errors in the pixel brightness and position. Aspects of image reconstruction are explored, and circularly band limited 2D signals are defined. Square, rectangular and hexagonal grids of sampling points covering 2D images are compared. Popular image sensing, display and storage devices are investigated, together with computer graphics techniques.

Chapter 4: Image Processing. In this chapter image processing is reviewed, and then the discussion centres on low level processes, that is processes on data that are associated with points on the sampling grid. Low level processes are divided further into those that are global or require a large area of support, and those that are local and require a small area of support. Those requiring a small area of support are more relevant to the remainder of the thesis. Many algorithms have been developed for operators in both of these groups for square sampled images. For hexagonally sampled images, only binary local algorithms, and some general purpose 2D signal processing global algorithms previously existed. Some local grey level processes, developed as a part of this project, are reported and compared with their square grid system counterparts.

Chapter 5: A Comparison Between Local Edge Detection Operators in Square and Hexagonal Data structures. The design of a hexagonal grid edge detector that is equivalent to the square grid Sobel detector is reported. Comparisons are made between the edge magnitude and angular accuracies of the two detectors in addition to comparisons of computational efficiency.

Chapter 6: Case Studies, a Comparison of Square and Hexagonal System Algorithms. Two industrial case studies have been undertaken to identify the relevance of the set of local operators assembled in Chapter 4. Redundant operators identified at this stage, lead to the simplification of the design of the pipeline PE. Comparisons are made between solutions for the case study problems on

the hexagonal and square sampling grids, from the points of computational efficiency and process reliability.

Chapter 7: Computer Architectures for Machine Vision. This chapter contains a survey of the computer architectures that have been commonly used for computer vision. The various possible architectures are classified, and their advantages explored. Finally, architectures suitable for processing images captured under controlled lighting conditions, such as is often the case with industrially derived images, are discussed. Pipelined systems of processor elements, within which small local image areas are assembled and operated on, show advantages for this class of image processes.

Chapter 8: A Pipeline Processor Element. The design of a processor element that can be assembled into a variety of image processing pipeline architectures is discussed. It will be able to perform the low level local image processing operations that have been identified in previous chapters as being useful for images that have been captured in industrial and controlled lighting conditions. It will be able to process both square and hexagonally sampled images at picture rates up to the video rate. Some novel pipeline image processors that use this re-configurable processor element as a building block are discussed.

Chapter 9: The Simulation of the Pipeline Processor Element. The results from a simulated pipeline of PEs are presented. The pipeline processing of industrial and controlled lighting environment images with PEs that accumulate only small local areas is shown to be feasible, on both the square and hexagonal sampling grids. The PE is simulated largely as it is described in Chapter 8, and internally, the circuit blocks identified there are realised by individual subroutines within the simulation. Some choices are then made as to which circuit blocks need to be implemented. The word widths of the data paths within the PE are chosen. Minimising these reduces the area of

silicon required by the device, but this must be balanced against a reduction in the accuracy of the calculations.

Chapter 10: Conclusions and Further Work.

Chapter 2

Human Vision, Computer Vision, and Industrial Inspection.

2.1 Introduction.

In this chapter the fields of human vision, computer vision, and industrial inspection are surveyed. The complexity of the human visual process is noted and the hexagonal packing of the retinal cells and the hexagonal interconnections between processing cells discussed. The scientific basis of computer vision is briefly introduced and the interaction between it and human vision research which has lead to the development of robust general image processing algorithms noted. With industrial image processing, however, the lighting conditions and the number of possible image scenes are strictly limited, and this has been exploited in the development of simple processing algorithms that can execute on inexpensive computer hardware. Some examples of industrial vision systems are given.

2.2 Human Vision.

2.2.1 The Historical Background of the Study of Human Vision.

Historically, the study of human vision was probably a necessary prerequisite to the study of computer vision. However, our knowledge of how biological vision systems operate is still fragmentary and with the recent development of the field of computer vision, a fresh impetus has been given to the study of human vision.

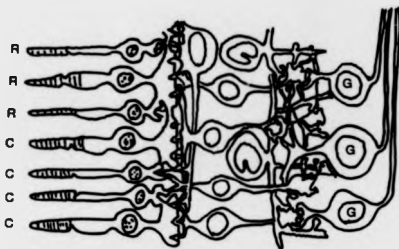
Biological and ophthalmic observations on the human eye indicate that a hexagonal packing of retinal sensory elements has evolved in nature. This is a motivation for the study of hexagonal sampling schemes for computer vision which is one of the topics covered in this thesis. Behind the eye, ganglion cells and neurons connect to the retinal sensory elements and to each other to provide processing of the image focused on the retina. Models of biological image processing have lead to the development of computer architectures such as artificial neural networks and pyramid processors for computer vision. Finally, studies of the "higher levels" of human vision have suggested algorithms for object detection and location, stereopsis and movement detection amongst others.

2.2.2 The Hexagonal Packing of Sensory Elements in the Eye.

Helmholtz in his "Treatise on Physiological Optics, 1909" [75], begins with an anatomical description of the eye. The higher orders of life have eyes capable of distinguishing both light and darkness and also form. To enable this, the eyes can have one of two forms. The first, common among insects, is a composite eye, in which sensory elements separated by opaque septa cover the surface of the eye. The elements at the surface of the eye, are usually of a hexagonal or square shape. The second form of eye, as with the eyes of many vertebrates, has a lens which focuses light onto a retina. A section of a retina is shown in Figure 2.1. The retina can be comprised of rod and

cone sensory elements. In the human eye most elements are of the smaller rod type. Cones are distributed amongst the rods in varying densities depending on the particular part of the retina. In the so called "yellow spot" only cones are found, whereas towards the periphery of the retina there are only rods. Behind the surface layer of rods and cones are layers of fine fibres connecting these elements to a layer of ganglion cells. These cells perform many processes, one of which is to pass information to the optic nerve. Thus the retina is a complicated array of different types of sensory element and has a number of layers associated with detection, interconnection, and possibly some image processing is also performed.

Nerve Fibres



R.....Rods C.....Cones G.....Ganglion Cells

Figure 2.1: The Retina.

From the anatomical drawings in Helmholtz treatise, it can be observed that the roughly circular sensory elements tend to pack together efficiently, which leads to a closely packed hexagonal lattice. Ophthalmic experiments reported in volume two of his work prove this to be the case. In one experiment, Helmholtz set up a grating of equal thickness light and dark lines, which was viewed

at various distances and under differing lighting conditions to measure the spatial resolution of the eye. His results indicated that two bright lines could only be distinguished if an unstimulated retinal element existed between the elements on which the images of the lines fell. This is in accordance with Nyquist's Sampling Theorem [134]. He also noted that for grid spacings close to the resolution limit of the eye, the lines appeared wavelike or modulated with repeated thick and thin sections as shown in Figure 2.2. From this effect he inferred that the cone sensors, the only type of sensor in the high resolution part of the retina, were packed in a hexagonal pattern.

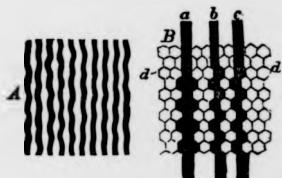


Figure 2.2: The Wavelike Appearance of Parallel lines when Viewed Close to the Eye's Resolution Limit and the Hexagonal Sensor Pattern that Produces this Effect. *Reproduced from Helmholtz Treatise [75].*

In a later experiment, Helmholtz noted that the resolution of the eye can be increased if the eye or the grating were allowed to move. Similar techniques are used today to measure the thickness of thin wires and filaments [125][11].

2.2.3 Models of Human Vision.

By the 1970's many advances had been made in neurophysiology and psychophysics. Interconnections between neurons could be seen physically and measured electrically. Researchers were able to determine the functions of single elements of the brain and the mechanism of vision was expected to be solved by study of the central nervous system. However, as reported by Marr [113],

this expectation was incorrect, as an analysis of the eye as an information processing system was also required. He also noted that visual information processing is very complicated even though, as humans, we are able to process information apparently instantaneously, and for a wide range of scenes under greatly varying lighting conditions.

Marr states [113] "*Vision is a process that produces from images of the external world a description that is useful to the viewer and not cluttered with irrelevant information*". He then continues to analyse vision as a system to which the input is the image of the external world and the output is the description. The description could be the spoken word, but more fundamentally, evidence points to a description that is a three dimensional model of objects within the image as being the output of the vision system. This model is independent of the vantage point of the observer in that all external surfaces are described, and Marr identifies a three stage representational framework to enable the three dimensional shape information to be derived from the original image. In his book, Marr [113] provides illustrated examples at the various stages.

The first stage is the *Primal Sketch*. Here, important two dimensional information in the image is identified, for example edge detection, where edges are defined as the boundaries between segments in the image, and may be found from discontinuities between the brightness of points within the image. The second stage is the *2.5-D Sketch*. Here depth information on the surfaces visible from the viewers reference point is calculated and discontinuities in depth and orientation noted. The final stage is the *3-D Model Representation*. Shapes and their spatial organisation are described in an object centered coordinate system. Volume and shape primitives are identified and attached to a wire frame model of the object to provide the final description. Marr produced many papers within this representational framework and developed many algorithms for the data processing primitives at the various stages. These are listed in the bibliography of [113] and his paper on the "Theory of edge detection" [114] is discussed further in Section 4.4.2.1. These algorithms can of course be modelled on digital computers. The algorithms and models developed

by Marr have been used as the basis of many of the current algorithms and models used in modern image processing research. Some of them, for example his edge detector and the 3-D model, appear to be difficult to implement efficiently on computers.

2.2.3.1 The Mapping of Visual processes onto the Biological Visual system.

More recent researchers have attempted to map vision processes more exactly onto the biological visual system. As an example, the following work by Watson and Ahumada [184][183] on a model of image representation in the visual cortex is included.

Anatomically, behind the hexagonally packed retinal sensors are a layer of retinal ganglion cells which, in the centre of the retina, connect one to one with the sensors [140]. The ganglion cells can also be considered to be connected on a hexagonal grid. The $2 \cdot 10^6$ ganglion cells connect to the visual cortex which contains approximately 10^9 neurons. Physiological experiments have shown that between the retina and the visual brain, the image undergoes a sequence of transformations. Sets of cells in the cortex can be identified with these various transforms. Watson and Ahumada consider a transform performed by the ganglion cells and a subsequent one performed within the cortex. The ganglion cells transfer spatial and brightness information. Their transfer function is broad-band and they provide local adaptive gain control. The transform within the cortex is different. The cells are narrow-band and employ a so called hybrid space-frequency code to convey the position, spatial variation and orientation of a region. Watson and Ahumada model the process in this set of cells by a structure, described below, which they refer to as a hexagonal orthogonal oriented quadrature pyramid. This is considered further here to indicate the complexity of the human visual system compared to current machine vision systems, and the necessity for the large number of cells involved to process a wide range of image types as they are presented in real-time to the human being. Real-time infers here that the images are processed quickly enough for us to continue to function as we do. Some machines may be able to process specific visual information

more quickly, but cannot cope with the variety of scenes and lighting conditions that a human can.

Simple pyramid processing structures, commonly used in computer vision, are described in Section 7.4.4. The image transform performed can be considered as image coding and performed by an image pyramid. Pyramids, introduced by Tanimoto and Pavlidis [170], filter the image into several levels of resolution. At the bottom is the highest resolution image. This is subsampled to produce a lower resolution image at the next highest level in the pyramid, and so on, until at the highest level is an image with the lowest resolution. The aim of Watson and Ahumada was to model the transform in the human cortex with a pyramid constructed from elements that were themselves modeled on known physiological components. The pyramid had a hexagonal lattice input layer, the transform was invertible, and the overall process was found to be efficient.

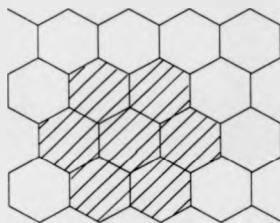


Figure 2.3: Tiling with Hexagonally Shaped Local Areas, a Seven Element Local Area That Produces one Value in the Reduced Resolution Image is Shaded.

The input image is passed on by the retinal ganglion cells to the lowest level of the pyramid. This level can be considered to be tiled with hexagonally shaped picture elements, known as pixels, as shown in Figure 2.3. The transformation to the next highest level in the pyramid involves taking a group of seven of these pixels, the shaded area in Figure 2.3, and producing one output pixel that contains a vector of values from a set of seven kernels, one of which produces the average brightness value of the local area, and the other six of which are bandpass and localised in space.

spatial frequency, orientation, and phase. Each low level pixel only contributes to one next level pixel, so the next level contains only $1/7$ the number of pixels, and so on until the apex of the pyramid is reached. The resulting hexagonal pyramid structure is shown in Figure 2.4. In this figure, the input image lattice is represented by the vertices and centres of the smallest hexagons and the highest level, which is also the lowest resolution image, is represented by the largest, thickest line hexagon.

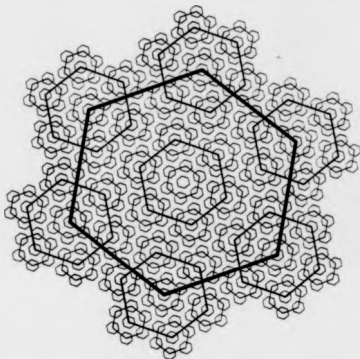


Figure 2.4: The hexagonal pyramid structure. *Generated using the program listed in the appendix of Watson and Ahumada's paper [184].*

At the highest level there may only be one pixel, but the vector associated with it encodes all the image information and can be decoded back down the pyramid to reconstruct the original image.

Watson and Ahumada have developed a model of an image process that occurs within the visual cortex. The model produces results that agree reasonably closely with physiological measurements, but some modifications, such as using larger kernels are needed. These will increase the complexity

of the model. The transform developed is a low level visual process, and is only part of the process of producing Marr's *Primal Sketch*. Human vision is very general purpose and processes are accomplished in real-time. The retina, visual cortex and visual brain have evolved to be highly developed, complicated organs. In the past studies of human vision have prompted the development of algorithms for computer vision, however, only a small consideration has been given to the hexagonal packing of machine vision sensors and the processing of the resulting hexagonally sampled images.

2.3 Computer and Machine Vision.

2.3.1 General Purpose Systems.

A general purpose computer vision system would include all the functions of human vision, and ideally, would compute in real-time. Here, real-time means that the machine of which the vision system is a part, can operate at its designed rate. Additionally, as noted by Horn [81] it would also have to reason about the physical world. There has been much research specifically towards the goal of the general purpose system, one approach has been to implement a system within Marr's processing framework (See Section 2.2.3). Such a study was undertaken in the UK as part of the Alvey Programme [3].

The core of a general purpose vision framework should be implementable on any hardware system, whether a biological system or an electronic computer. Differences occur at the input and output stages. In computer vision, the image input sensor could be one or more from a widely divergent set including:- monochrome and colour TV cameras, line scan cameras and multiple frequency band detectors such as those used for remote sensing [68]. System output will depend on the environment, and on the overall function of the machine of which the vision processor is a part.

For a computer vision system, the choice of architecture is an important consideration. The more

general purpose a system is, the higher the computation load. Additional constraints are imposed by the time available for the computation and the financial cost of the system. Image input systems usually provide a 2-D image, exceptions include line and raster scanned input devices which produce 1-D information, but lines of data are associated with one another and the overall data set is 2-D. Some devices such as nuclear magnetic resonance scanners and laser range finders [91, 90] produce 3-D or 2.5-D information directly. Processes within the system can produce multidimensional data sets and other data sets such as knowledge bases will also need to be operated on. Rosenfeld [153] lists commonly used architectures. The basis of the list is included here with some additions:-

- Single processor systems, these operate on one data byte at a time.
- Multi-processor systems, these operate on several bytes in parallel. A sub-list of types follows:-
 - Pipelined systems. 1-D arrays of processors performing a sequence of operations on a stream of data passing along the array.
 - Systolic arrays. One or more dimensional arrays of processors performing the same operation on every datum in the array [55]. Note, alternative definitions exist.
 - Mesh connected systems. 2-D arrays that can perform local operations on the whole image simultaneously.
 - Trees and Pyramids. These can perform global and local kernel operations on the data. The lowest level processors in the architecture are associated with small sections of the image. Groups of these processors communicate upwards with a reduced number of processors at the next highest level, and so on, until at the highest level a single processor contains the final result of the image process. A variation of the architecture also allows data flow in a downwards direction. Such systems are often used for multi-resolution

image processing. Multi-resolution processing forms a part of Marr's [113] overall processing framework.

- Hypercubes. These combine the advantages of meshes and pyramids. The topology of the network consists of a Boolean n cube.
- Shared memory machines. Use an interconnection network between the processors and common memory. They can simulate other architectures.
- Artificial neural networks.

Some of these architectures employ three or multidimensional arrays of processors, but there is an interconnection limitation with such systems. Closely packed 3-D arrays of interconnected neurons exist within the human visual cortex, but it is not possible to effect such interconnections on integrated circuit devices. Connections are limited to a 2-D plane. Nudd [130] reports a compact system using current technology to interconnect a stack of wafer scale integrated processors. Each wafer contains an array of 32×32 processors, and typically seven wafers containing different processor types may be required for a simple image operation. Data is loaded serially into the top wafer in the stack, processed, and then transferred vertically to the corresponding processor in the wafer below. It may be possible to make the top wafer a CCD device that could accept direct light input. With systems for larger, say 512×512 images, the number of interconnections may make such a stack difficult to implement.

A general purpose computer vision system would be most likely to succeed, if implemented on a highly parallel machine, as these architectures maximise processing rate. From the above list, pyramid, hypercube and artificial neural networks appear most suitable. However, practical systems tend to have more modest specifications, and the requirement to be cost effective leads to implemented systems with a wide variety of architectures.

2.3.2 Machine Vision.

A machine vision system must produce a symbolic description of the scene being imaged. The description can then be used to direct the operation of a robot, a production machine, a production line, or a vehicle etc. If machine vision is classed as a sub-category of computer vision then there are several other sub-categories relating to this field. These are listed by Horn[81] and are included with explanation here:-

- **Image Processing** The input image is transformed to a new output image. The output image is often enhanced in some respect, for example several images may be averaged to remove noise, or the contrast may be stretched to spread grey levels more evenly through the available range[65]. These types of operation make it easier for humans to obtain information from the image.
- **Pattern Classification.** Features in a preprocessed input image are extracted and then classified by the required metric. For example a surface defect can be classified by size, or a printed character by a code such as ASCII[30].
- **Scene Analysis.** This is the transformation of simple analysis such as the output of an edge detector into a more elaborate one suitable for the particular task. For example, the analysis may need to determine if a particular set of edges represents a particular object.

Research described in this thesis is relevant to each of these sub-categories. Machine vision encompasses a range of applications, some of which will be further explored below, and draws on a number of computer vision techniques. The major classes of machine vision are now reviewed.

2.3.2.1 Robot Vision.

A robot vision system is defined as being a vision system that is part of the control mechanism for the robot[81]. Consider a robot arm and manipulator that has to pick parts from a conveyor belt.

The vision system images the part and obtains information on its type, its position and orientation in 3-D space, and its rate of movement. This information is used to position the robot's manipulator and to enable it to grasp the part. The vision system will be continually updating this information as the arm moves the manipulator to intercept the part. It can then guide the arm to deliver the part to its destination.

The vision system must employ many algorithms to effect these tasks. An overview of these is given by Hall and Nurre [133]. At the lowest level, the scene must be separated into its component segments, i.e. to distinguish the part from the background and other unwanted parts. 3-D positional information must be obtained (see [128, 142]), perhaps from stereo pairs of images and a model of the camera geometry (see [176]). This positional information must be related to the robot's geometry and the results communicated to the rest of the system. All these results have to be calculated in real-time. The time available is a function of the operating speeds of the robot and the conveyor.

Mobile robots can be "on the road", for an example see [104], or "all terrain", see [50]. The first category includes robots that use the sides of a road, or markers, to sense their position and plan their routes along the road, whereas the all terrain vehicle can move freely in a 2-D plane. Additional algorithms are required to detect the robots position, for navigation, and obstacle avoidance etc.

2.3.2.2 Visual Inspection.

The topic has been reviewed by several authors:- Chin and Harlow[26] (1982), Wallace[182] (1988), and Chin[25] (1988), are comprehensive works. An introductory textbook containing industrial examples has been written by Batchelor, Hill and Hodgson[11] (1985).

Two classes of visual inspection system can be distinguished. In the first, the inspection machine reports information to a human operator, and in the second, known as automated visual inspection, the machine acts on the information autonomously. The first class includes medical applications

such as digital radiography, computerised axial tomography, the gamma camera, and ultrasound imaging devices. Other examples can be found in industry. In manufacture, where many of the techniques above are also used, there is an example of a real-time system to aid glass blowers in maintaining the quality of their product[129]. In the security industry, there are systems using movement detection to scan for the presence of intruders. Many of these systems could be classed as image processing machines in that they enhance images or reconstruct the image from a set of transmission measurements, but often facilities are provided so that measurements can be made. For example, plots of kidney function can be obtained from repeated gamma camera images [136].

With automated inspection, the vision system makes the choice between a good and a bad artifact. More complex machines are able to grade at more than two levels, while others can be part of a feedback loop controlling a process.

An example of binary selection is given in [173] where a vision system selects potatoes for marketing as baking quality. Such a potato must be large, of good shape, and free of surface defects. Letter post code reading machines grade the letters at many levels, so that a letter is directed to one of several million locations[60]. The following are examples where the inspection system forms part of a feedback mechanism:-

- Piece work quality checking. With piece work, a human assembly worker is paid for each piece produced. The incentive is to produce as many items as possible, but quality must be maintained. In one system, involving mechanical sub-assemblies, the operator is prevented from removing the item from the assembly jig until the vision system has checked the correct location of each component[6].
- Seam welding. As welding progresses, the vision system identifies the gap position and checks the width of the weld[149].

Some examples such as those above could be classified as robot vision applications, but others

involve the control of a complete plant, in that quality information obtained from the vision system may effect automatic adjustments to a number of machines.

The following is a short survey of practical automated visual inspection systems listed under the industrial area in which they occur. Only a few industries are considered, but hopefully an appreciation of the wide range of applications can be obtained. For each system the most important considerations in the inspection process are the defect detections reliability, the operating speed, and the system cost.

The electronics industry.

- Printed circuit board (PCB) tracks. The vision system inspects the boards for broken and thin tracks, missing pads, unwanted connections, and missing drilled holes. Line scan cameras are often used and images containing 10^6 or 10^8 picture elements (pixels) accumulated. Processing can involve referencing the captured image with an electronic data base image[172] or searching for generic defects[110]. Input images are thresholded to produce binary representations which are then processed by parallel processors. Pipeline processors comprising hundreds of processor elements are often used[147].
- PCB component position checking. Correct insertion and labelling of components can be checked[139].
- VLSI photomask checking. Masks contain many elements, but patterns are often repeated many times. Huang[83] and Akiyama[2] describe techniques for alignment and inspection of the mask based on cross correlation with a reference image.
- Wire thickness. Wire is spooled at high speed past a line scan camera and thickness measurements performed. System optics are critical[125].

The food industry. A wide variety of system applications exist, a few are listed here:-

- To check packaging. For example the health warning labels on cigarette packets[23].
- To check the quality of fruit and vegetables. For example kiwi fruit[79] and potatoes[173]. Size, shape and surface blemishes are checked.
- To check production line foodstuffs. For example biscuits[38] for such parameters as size, defects, and thickness of chocolate. Production line products may overlap or touch which complicates processing.

The automobile industry. The applications in this area are very varied, ranging from small part location and inspection[138], to inspection of complete assemblies[177]. A few examples are listed:-

- Car body dimensions. In one application developed by Automatix[6], several laser stripes are directed onto the body work and from their position and deformation certain critical dimensions can be checked.
- Correct assembly of drum brakes. With this system, developed for Volkswagen and described in[80], the system checks for correct component insertion and positioning. A Fourier technique can even check for the correct return spring by checking for the correct frequency transformed from the spatial position of the coils.
- Inspection of cylinder bores. West and Stocker[187] describe a system using laser light to check the surface quality of cylinder bores.
- Paint finish. The characteristic "orange peel effect" of incorrect paint spraying can be spotted [11].

- Instrument and warning light operation. Systems have been developed from meter reading techniques, as described in [175], to check the cars' instrumentation.
- Component cracks. Many techniques exist, for example by magnetic liquid penetration [62] and by Xray [68].

2.3.2.3 Other Applications

Some applications may not fit into the above categories or be better classified under fields closely related to machine vision (see Section 2.3.2). Such applications include document reading where documents are imaged and transformed to computer text files. Visual database acquisition, for example the building of finger print libraries, is another application.

2.4 Industrial Inspection, Engineering Considerations.

2.4.1 Reliability, Processing Rate, and Cost Effectiveness.

Practical automated visual inspection systems need to be reliable, quick in operation, and cost effective. Reliability refers here to the success rate of the system in performing the required task. For example will a particular defect be detected every time it occurs?

The image processing must be performed at least at the operating speed of the machine or production line. This is by no means easy, as after digitisation a typical image may contain a quarter of a million data bytes, and be updated every 20ms. Together with the low level processes of extracting information on actual objects from within the scene, the high level tasks of characterisation and measurement of objects will require a considerable amount of computation.

A vision system will enhance a production machines operation, but total system cost is a very important factor. In the UK the average price of an installed industrial robot was £20,000 in 1985. This figure is higher in other countries, but even at the research stage the eventual system cost needs

to be considered. An automated visual inspection system will not be installed in a competitive industry unless it is cost effective.

Processing reliability can be increased by a careful choice of illumination, processing algorithms and image sampling scheme. To obtain an acceptable reliability at the real-time part rates of a typical production line will probably require a computer system employing a parallel architecture. The choice of architecture has an important effect on the cost of the system and is considered further in Chapter 7. The illumination technique can also be used to simplify the processing task, for example, back lighting could be used to enable efficient thresholding of the image of an item that is being checked for perimeter defects. Thresholding produces a binary image that is easier to process [1]. The illumination method for a particular task is usually decided on after some experimentation, and depends on the experience of the designer. Batchelor[8] has devised an expert system lighting advisor. Arranging for the part to be presented at a known position and orientation can also reduce the image processing task. This could involve the use of a simple mechanical deflector on a conveyor belt, or a more expensive robot manipulator [1].

As with illumination, processing algorithms tend to be chosen by the system designer from experience. Vision development systems such as the Automatrix A190[6] and the Context Vision GOP300[29] are widely used in industry. These contain many image processing algorithms that were installed by the manufacturer, and also macros of these that have been assembled by experienced vision system designers. New algorithms will be incorporated into the development systems when the existing ones fail to solve problems. In the same way more powerful computers, which may contain parallel processors, can be added if the existing hardware does not operate quickly enough.

The transfer from the development system to an embedded system forming part of the production system usually involves running the application software on a rugged version of the same hardware. If parallel architectures are to be used, they must be flexible for general inclusion within the development system, or available as a variety of plug in optional extras. As described in Chapter 8

of the thesis, the flexibility of the programmable pipeline processing element proposed in this project makes it suitable for inclusion in both development and embedded image processing systems.

The sampling schemes most widely used are square and rectangular arrays of sampling points. Each point usually has a small tile or picture element (pixel) associated with it. The pixels fit together to enable a continuous display of the image. In the square scheme, samples are equally spaced on horizontal and vertical lines. The pixels, and often the overall image, are square in shape. In the rectangular scheme, samples are again constrained to lie on horizontal and vertical lines, but the horizontal and vertical spacings differ. The horizontal spacing is usually the larger, enabling images with the same number of horizontal and vertical samples to fill a 4x3 aspect ratio TV screen. Hexagonal sampling schemes show advantages over other schemes (these are discussed in Chapters 3 and 4) but when they will be able to break in to the commercial development system evolutionary cycle is uncertain.

2.4.2 Automated Surface Inspection.

In Section 2.3.2.2, automated inspection examples were classified according to industry. Here a classification is presented according to function. This classification is useful as within a class, certain parameters such as the method of illumination, basic algorithms, and the presentation of data tend to be common. The classification follows:-

- Inspection for item identification.
- Inspection for item shape.
- Inspection for correct assembly.
- Dimensional inspection of items.
- Surface inspection. For defects, finish quality, and surface features such as PCB tracks and printed characters.

- Sub-surface inspection

Surface inspection is the area from which the processing examples in the remainder of this thesis originate, and so will now be reviewed under the following sub-headings.

2.4.2.1 Surface Defects and Finish Quality.

A surface defect can be a pit, scratch, crack, or raised bump on an otherwise smooth surface, or, a discontinuity in a pattern or texture. Defects can occur randomly or in groups, can be of a characteristic size and shape, or of a random dimension. The surface may be flat or curved. Curved surfaces cause problems with specular reflections, shadows, and illumination gradients across the surface are more difficult to accommodate. Graham[66] has designed an illumination system to compensate partly for these problems. Several surfaces may need to be inspected on a particular 3-D object, requiring multiple views or manipulation of the part[1]. Additionally the inspection may be of an internal surface, as with cylinder bores[187].

A defect may be detected by comparing the image of an object with a perfect model of the object. The model may be obtained during a teaching phase[76, 115], or by direct input from a computer aided design - computer aided manufacture (CAD-CAM) system[132]. With this approach a defect is any feature that is not included in the perfect model of the object and false defect detections may result from image noise being detected. An alternative is to compare the image with models of known defects, but this assumes the defects are always of a constant size and orientation. Another detection approach is to search the image for generic features associated with the defect. With this approach there is a possibility that unexpected defects will be missed, however, detection is independent of the defect's size and the technique is widely used in industry. This final approach is the basis of methods using mathematical morphology, statistical, and Fourier plane techniques. Examples using morphology are given by:- (a) Serra[157], in which he describes analysis of the crystalline surface structures in materials, (b) Batchelor and Cotter[10] who describe a system for

analysing cracks in noisy surfaces, and (c) Mandeville[110] who describes a system for inspection of PCBs.

An introduction to statistical analysis in pattern classification is given by Duda and Hart[46]. Examples of its use for defect detection can be found in the following:- (a) Suresh et al.[169] have developed a complete system including architecture and algorithms for detecting and classifying imperfections in the surfaces of hot steel slabs. A statistical binary tree classifier makes decisions such as:- Is this feature noise or a defect? Or, is this crack horizontal or vertical? (b) A statistical analysis is used by Connors et al.[28] in the inspection of wood for knots. Grey level tonal qualities are measured in local areas and the mean, variance and skewness of the levels obtained. Textural methods based on co-occurrence matrices[73] are also employed. Fourier plane analysis can be used to detect repeating defects in patterns printed on textiles[178] and ripple effects in spray painted surfaces[11].

3-D information will be required for certain applications, for instance, the volume of a pit, or the depth of a scratch can be measured using laser range finder[90], stereo pair[160], depth from shading[81], or structured lighting[132] techniques.

Surface finish quality measurements can be made. In an example involving metal surface roughness measurements, Don and Fu et al.[44] inspect for surface profile parameters. A mathematical model of a surface profile is developed which relates image variables to the surface parameters.

Finally, Ozaki et al.[137] has produced an automatic visual inspection system for checking print quality. This leads to the next sub-section on optical character recognition.

2.4.2.2 Handwritten and Printed Character Reading.

This is one of the lower level processes that form part of the operation of document reading. It involves transforming character shapes to, say, ASCII codes. Previous processes will have divided the scanned document into blocks. Each block contains one word and the relative block positions are

known within the page. Words are then separated into letters. This is easier to achieve with machine printed characters as gaps are maintained between them. With handwritten text, clues are used, such as the way two cusps join[60, 15] to separate the letters. Even with printed characters, there exists large variations in typeface and size, and simple template matching would not accommodate these.

Most recognition algorithms firstly thin the character to a single pixel wide skeleton representation, and then investigate this for generic features such as end points, junctions and deflections. Knowing the relative position of these, a tree structure can be searched and the character recognised. Thinning algorithms for characters have been extensively surveyed by Smith[164], and character structure is investigated by Kerrick and Bovik[95]. This structural recognition method has been extended to hand printed characters, but more feature types have to be recognised[24]. Recent alternative approaches include:- (a) Masih and Stonham's[115] proposed artificial neural network. The network would be trained to produce the ASCII code for each character. (b) Gilles et al.[60] have investigated the relative position of cavities in the characters as the starting point for recognition. A cavity is a space enclosed on three or four sides. Open cavities are classed as north, south, east or west facing. They applied the method to hand written ZIP code (Post codes as used in the USA.) reading. Note that ZIP codes contain only numerical characters.

Commercial systems usually employ line scan cameras, and allow the document to be digitised with a typical resolution of 300 lines per inch. Parallel processor architectures are often used to enable reading at a reasonable page rate. Crimmins and Brown[31] have described the implementation of morphologic character recognition algorithms on the "Cyto Computer", a pipeline processor. Gillies'[60] ZIP code reading algorithms were also implemented on this machine. Chin[27] described a skeletonising algorithm for a pipeline processor, and developed a processor element design based on combinational logic elements.

2.5 Conclusions.

The first section of this chapter concerned human vision. In particular the hexagonal packing of sensors in the human eye, and the result of this spatial arrangement on the low level image processing was noted. This evolutionary evidence is the first evidence presented in the thesis for the possible superiority of the hexagonal scheme. Much computer vision research is based on studies of human vision, and research in these two branches of science tend to complement each other.

In the section on computer and machine vision the idea of a general purpose machine, combining both low and high level processes, that aims to perform as the human eye does is presented. Such a machine would have to extract information from many different types of scene, under various lighting conditions, and operate at the same speed as a human does. The discussion on hexagonal sampling schemes in later chapters of the thesis are applicable to such general computer vision research, but the results presented on the performance of various image processing operators, and the pipeline processor architecture are more suitable for industrial image processing systems.

The final section of this introductory chapter concentrates on examples of image processing found in industry. Even in this relatively narrow area of applications there are systems providing many different types of high level information. Each has as input 1, 2, or 2.5D images, but the output may be controlling a robot that operates in a 3D space, making quality adjustments to a production line, or be a computer coded version of a section of printed text. Many of the applications reported here utilise 2D arrays of sensors or 1D arrays scanned across the picture to build up a 2D image.

Common to applications from all of the industries surveyed here, researchers have tried to maximise the signal to noise ratio of the image by arranging for the scenes to be optimally lit and for the sensors to be the most appropriate. In industrial situations the number of different types of object that will be presented to the image processor in any one application is usually from a limited set, and if possible, objects are presented to the image processor in a known orientation

and separated from other objects. When the above points are considered both low and high level operations are simplified and inexpensive image processors that reliably perform their tasks can result. Low level image operators that are often used for industrial image processing are explored further in Chapter 4. In industry, the time available for the image processing to be completed is often limited by the speed of a production line. An appropriate computer architecture, a pipeline, for industrial image processing is investigated in Chapters 7, 8, and 9.

Chapter 3

Image Sampling.

3.1 Sampling Theory.

Before a digital computer can process an image, the image must be sampled, and then the quantity sampled digitised. The images considered here are 2D brightness fields, Nyquist's sampling theorem [134, 146] applies, and is given here in terms of wavelength, as the more usual frequency definition can strictly only apply to time varying signals. The theorem states that the sampling interval must be less than or equal to half the shortest wavelength present in the image, for the analogue image to be exactly reconstructed from the sampled values. However many textbooks and papers concerning image processing and multidimensional signal processing define the term "frequency" to be the reciprocal of the wavelength multiplied by π . This definition will be used in the remainder of this thesis to simplify cross references to these other works. All future references to band-limited signals, low pass filters, and high pass filters use this definition of frequency.

In practice, the highest required image frequency is chosen and the analogue signal is frequency band limited to that frequency. Ideal filters are not realisable and so the sampling frequency will always be greater than twice the highest required image frequency. Quantisation error is introduced by the digitisation of the sampled quantity. The precision of the quantised word is an important

design consideration. Other errors in the process include sample point position error, and error in determining the quantity being sampled. For instance, is the quantity sampled the average value of the signal in a local area surrounding the sampling point? In the reconstruction of the analogue brightness signal from the digital representation, the binary word is converted back to an analogue value to represent the signal at the particular sampling point. The signal between sampling points is reconstructed by an interpolation process involving contributions from up to n surrounding points. Such an interpolation was used by Shannon[159]. Workers such as Nyquist and Shannon were concerned with 1-D functions of time, and, in the early nineteen sixties, Petersen and Middleton[141] extended the theory for the sampling and reconstruction of frequency band limited functions in N-Dimensional spaces. More recently (1990), Oakley and Cunningham[135] have designed a set of reconstruction filters for N-D images. An ideal filter would reproduce the original signal exactly between the sampling points. Petersen and Middleton's reconstruction filter is optimum for randomly occurring images, whereas Oakley and Cunningham's is optimised for each individual image. However for image processing applications, the reconstruction process is usually limited to simply holding the present sampling point value until the next is available, or to the use of a simple low pass filter to smooth the brightness transition between sampling points. These simple methods are often adequate for human viewing as images are usually reconstructed from a large number of sampled values. Further consideration here is limited to 2-D space.

A system for processing analogue images will comprise a sensor to convert, say, the brightness to an electrical signal, followed by an analogue to digital converter, and then a digital processor. Eventually, a digital to analogue converter can reconstruct an analogue signal, which can be viewed on a display system.

The image will be sampled at a number of points, at known positions, distributed throughout the image. It is easier to record the sampling point's positions and to address them if they are arranged in a simple pattern. Various patterns are examined in Section 3.2.1 and image sensors are discussed

in Section 3.3.

Image processing is performed on the arrays of numbers that comprise the digitised images. Each number is a measurement of brightness at a particular sampling point. Usually for image display, and sometimes conceptually for processing algorithm development, the image is considered to be tiled by pixels (picture elements). Pixel shapes should fit together to produce a continuous tiling of the image, and for simplicity, the sampling point should be located at the centre of the pixel. The need to tile the image constrains the variety of sampling schemes considered here. Display systems are considered further in Section 3.4.

3.2 Sampling Schemes.

3.2.1 The Square Scheme.

Figure 3.1 shows the conventional square grid pattern of sampling points. The points are arranged in equally spaced rows and columns, and the row spacing is equal to the column spacing. Mersereau[124] and Preston[144] suggest that this system was the initial choice for the sampling of 2-D signals as it is conceptually easy to generalise processing algorithms from the 1-D case. This grid, and its more general case, the rectangular grid (Section 3.2.2), have been chosen for virtually all implementations in image processing. However, it is not the most efficient scheme in terms of the number of sampling points required for equal high frequency resolution[109, 124, 126, 135, 141, 144, 188].

The definition of a circularly frequency band limited 2-D spatial signal, is that the highest frequency within the signal is equal for all directions within the spatial plane in which it lies. If the signal is transformed to the Fourier plane, and zero frequency, ω_0 , is shifted to the centre of the plane, then such band limiting is represented by a circle, centre ω_0 , radius ω_{max} . All frequency components of the signal lie within this circle.

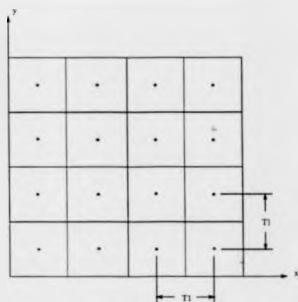


Figure 3.1: A Square Grid of Sampling Points $T_1 \times T_2$. The Area is Tiled with Square Shaped Pixels.

As shown in Figure 3.2, square sampling permits a square tessellation of the Fourier plane [124]. The first square, centred at the origin, represents the non-aliased band region permitted by square sampling. The extent of the square along the Ω_1 axis represents the maximum non-aliased frequency in the spatial x (Figure 3.1) direction. Likewise, the extent along the Ω_2 axis represents the maximum non-aliased frequency in the spatial y direction. It can be seen that higher non-aliased frequencies could be represented in all other directions. The surrounding squares in Figure 3.2 are periodic extensions to the fundamental square band region. If an image is circularly frequency band limited, then high frequency information will be processed equally, independently of the scene or object orientation. This is important for repeatable results in industrial inspection applications.

For an analogue signal, $F(x, y)$, square sampling can be represented by Equation 3.1. All of the equations included in this section are proved by Mersereau [124].

$$f(n_1, n_2) = F(n_1 T_1, n_2 T_2) \quad (3.1)$$

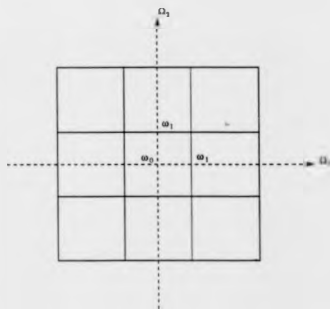


Figure 3.2: A Tiling of the Fourier Plane with Square Band Limit Regions.

Where T_1 is the sampling interval as shown in Figure 3.1, and n_1 and n_2 are the horizontal and vertical sampling addresses. $F(x, y)$ is frequency band limited by a band region R if its transform to the Fourier plane $\mathcal{F}(\Omega_1, \Omega_2)$, satisfies Equation 3.2.

$$\mathcal{F}(\Omega_1, \Omega_2) = 0, \quad (\Omega_1, \Omega_2) \notin R \quad (3.2)$$

For square sampling, if $T_1 < \pi/\omega_1$ is satisfied, then $F(x, y)$ can be exactly recovered from the sampled values. Where ω_1 is the horizontal and the vertical bandwidth (see Figure 3.2).

3.2.2 The Rectangular Scheme.

The square sampling scheme is a special case of the general rectangular scheme. With rectangular sampling, the horizontal and vertical sampling intervals are not necessarily equal. Figure 3.3 shows an x interval of T_1 and a y interval of T_2 . The sampling equation generalises to Equation 3.3, and $F(x, y)$ can be exactly recovered from the sampled values if $T_1 < \pi/\omega_1$ and $T_2 < \pi/\omega_2$

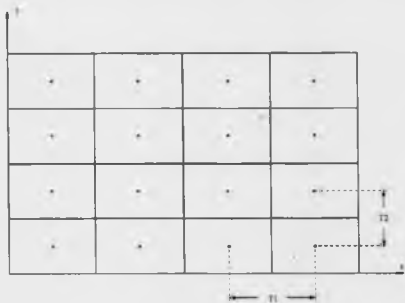


Figure 3.3: A Rectangular Grid of Sampling Points^{*}. The Area is Tiled with Rectangular Shaped Pixels.

$$f(n_1, n_2) = F(n_1, T_1, n_2, T_2) \quad (3.3)$$

Where w_1 is the horizontal and w_2 is the vertical bandwidth. Square and rectangular band regions are shown in Figure 3.4, an optimum circular band region is superimposed on each. For circularly band limited signals, it can be seen that square sampling is optimal in the sense of the density of sampling points required to reconstruct the signal exactly. This follows from the source having equal x and y dimensions. Rectangular sampling schemes are sometimes used for digitising 4:3 aspect ratio TV images. The scheme allows for an equal number of horizontal and vertical points.

3.2.3 The Hexagonal Scheme.

Figure 3.5 shows a regular hexagonal grid of sampling points. With a regular hexagonal grid, each point is equidistant from each of its six nearest neighbours, and points on alternate lines are shifted

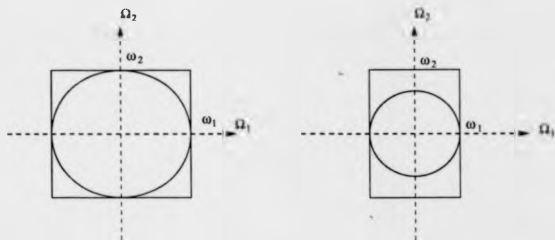


Figure 3.4: Square and Rectangular Band Regions Containing Maximal Circular Band Regions.

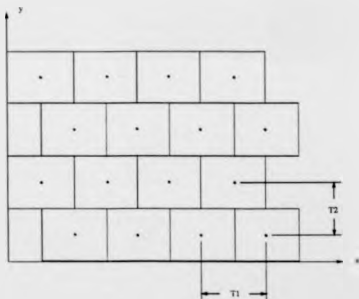


Figure 3.5: A Hexagonal Grid of Sampling Points ω^* . The Area is Tiled with Rectangular Shaped Pixels.

by half the horizontal sampling distance. In this arrangement, horizontal nearest neighbours exist, but not vertical ones. Alternatively, the pattern can be rotated to give vertical, but not horizontal, nearest neighbours. Here, rectangular pixels are shown. Such a tiling is advantageous if raster scanned input and output devices are employed, as described in Section 3.3.

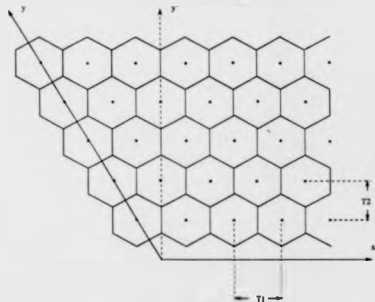


Figure 3.6: A Hexagonal Grid of Sampling Points. The Area is Tiled with Hexagon Shaped Pixels.

A hexagonal tiling is shown in Figure 3.6. Here a non-orthogonal y axis has been chosen to ease indexing of the grid points. An alternative axis at $\pi/3$ to the $+z$ axis could be chosen, but for positive indexing of overall rectangular scenes, the $2\pi/3$ axis is required. Using this y axis, hexagonal sampling can be described by Equation 3.4, and $F(x, y)$, for a general hexagon, can be exactly recovered from the sampled values if Equation 3.5 is satisfied.

$$f(n_1, n_2) = F\left(\frac{2n_1 - n_2}{2}, T_1, n_2, T_2\right) \quad (3.4)$$

$$T_1 < \frac{4\pi}{2\omega_1 + \omega_2} \quad T_2 < \pi/\omega_2 \quad (3.5)$$

Variables ω_1 , ω_2 , and ω_3 are defined in Figure 3.7. If regular hexagonal sampling is employed, then $\omega_1 = \omega_2 = 2\omega/\sqrt{3}$. If a signal is circularly band limited to frequencies below ω radians s^{-1} , then for a rectangular system the maximum sampling period is $T_1 = \pi/\omega$, $T_2 = \pi/\omega$, and for a regular hexagonal system, $T_1 = 2\pi/\sqrt{3}\omega$, $T_2 = \pi/\omega$. For the hexagonal system, period T_1 is increased. Mersereau [124] calculated that 13.4% fewer samples were required to meet the Nyquist criterion.

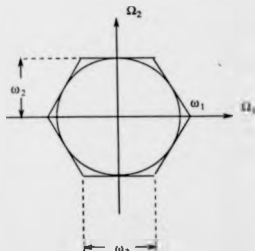


Figure 3.7: A Hexagonal Band Region Containing A Maximal Circular Band Region.

3.2.4 Other Schemes.

In Section 3.2.3, the hexagonal scheme was shown to be more efficient than the square. The optimum scheme will have a band region which is as close as possible to the band limit region of the 2-D input signal. For a circular band limited signal, Petersen and Middleton [141] show the regular hexagonal scheme to be the optimum scheme, and calculate its efficiency to be 90.8%. The comparable square lattice efficiency is 78.5%.

Other schemes exist. Whitehouse and Phillips [188] describe a trigonal arrangement, where each point is surrounded by three nearest neighbours. Their application was the measurement of surface

roughness with point contact probes. The scheme enabled a mechanical probe to be positioned and a measurement taken in a minimum time, but it is not as efficient as the hexagonal scheme for the raster scanning of image processing sensors.

3.3 Image Sensors.

In a computer vision system, the image sensor converts electro-magnetic radiation (often light) into an electrical signal that represents the image. This signal is sampled and then digitised. Sensors can be single point sensors that are mechanically scanned across the image, or arrangements that are electronically scanned, such as in the TV camera. Other systems may use a single sensor and a scanned light beam, as with the laser range finder[90].

Sensors have finite cross sectional areas. The position of the sampling point within the sensor is uncertain. Kamgar-Parsi and Kamgar-Parsi[92] have developed a model for estimating the average error due to quantisation with square sampling systems. They have extended this so that the probability of an error being within a particular range can be obtained. It is then possible to determine whether the quantisation noise is acceptable for a particular application. Subsequently[93], they have extended the model to include hexagonal sampling, and from their mathematical analysis have concluded that hexagonally shaped sensor elements yield smaller quantisation errors.

3.3.1 TV Cameras.

3.3.1.1 Raster Scanned Devices and Image Sampling.

With these devices, the 2-D image sensor is scanned a row at a time and the information transferred sequentially. In the terminology of TV technology, the scan lines are equal to the rows of the image array. Each line is initialised electrically by a synchronisation pulse. The set of lines that completely scan the image in one pass are referred to as a frame. In the European standard[118], 625 lines

constitute a frame, and a new frame is initialised by a frame synchronisation pulse. Many cameras operate in an interlaced line scan mode in which the set of odd numbered lines, known as the odd field, are scanned first, before the even field. This facilitates lower band-width transmission.

It is possible to construct cameras with differing numbers of lines and different scanning geometries. It is usual for the overall sensing area to be of rectangular shape with a 4:3 aspect ratio, but 1:1 ratio cameras are available, and designs exist for programmable picture warp cameras[67].

The raster scan camera effectively initiates the sampling process as it sections the image into rows. The scanning electronics are simplified if the line spacing is constant. All the sampling schemes considered in Section 3.2 can be sampled by a constant line spaced raster. Sampling is completed by a 1-D process, synchronised by a master clock, that divides each line into a number of samples. This clock may increment the address to a CCD (Charged coupled device) sensor array and time the A-D (Analogue to digital) conversion process, or for a fully analogue camera, simply time the A-D converter. Converted words are then stored or processed. An alternative camera, the line scan camera, consists of a 1-D array of CCD sensors. The object is moved in front of the array so that a 2-D image is built up.

For equi-spaced scanning of rectangular arrays, scan lines are normally arranged to coincide with the horizontal rows of required sampling points. The value at the first point is converted at a fixed time after the line synchronisation pulse for each line. Horn[81], notes that hexagonal sampling can be realised in the same way, but that the delay between the synchronisation pulse and the first sampling point, needs to be increased by half the sampling period on alternate lines. Regular hexagonal and square geometries are realised by calculating the sampling period so that the required horizontal physical spacing results. For hexagonal sampling, an alternative scanning scheme would be to scan parallel to the non-orthogonal y axis, as shown in Figure 3.6.

3.3.1.2 Camera Technology and Image Sampling.

In the iconoscope vacuum tube camera, light falls on a photosensitive mosaic formed by the deposition of millions of silver globules. Each globule forms a capacitor with a common electrode provided in the construction of the sensor. A charge is induced in each capacitor proportional to the light intensity falling on it. The charges are then interrogated by an electron beam scanned across the mosaic target. Modern tubes have more sensitive targets than the iconoscope[11]. The image has been sampled by the mosaic of capacitors, but the relatively wide electron beam reconstructs a 1-D analogue signal that is circularly band limited.

With solid state cameras, the sensor is a 2-D array of say CCD (Charge coupled devices). A typical array size is 512x512 elements, although a more sophisticated device may have a programmable resolution of up to 3000x2300[96] pixels. The overall sensor area is typically rectangular or square. Rows of analogue pixel values are transferred, in turn, to a line shift register, and then shifted out at a master clock rate to produce a raster scanned TV signal to a particular standard. CCD array resolutions are comparable to the resolutions required for computer vision. Two image sampling strategies exist.

In the first strategy, the line shift register (or pixel) clock can be synchronised with the A-D converter so that the analogue pixel value is digitised while it is stable. Sampling is effected by the CCD array and depends on the array geometry. Many arrays are available for rectangular and square sampling, but these geometries should not be assumed to always be the case, as it is less expensive to fabricate devices on a hexagonal grid[120]. The Kontron ProgRes 3000 [96] camera oversamples the image. This allows for the output samples to be programmed for rectangular, square or hexagonal grids. The acquisition system from light input to digital output has been studied by McClellan[121]. He finds non-ideal characteristics which include variations in transfer function from pixel to pixel, non-linear pixel response to illumination, and the diffusion of light within the

array. He suggests a correction procedure, but it is limited to correcting linear differences in the brightness responses of the individual pixels. At present this would appear to be the only correction procedure available for applications operating at the video rate.

In the second strategy, the CCD array pixel clock and the A-D converter are not synchronised. The signal is resampled at the converter. Errors will occur as the camera output signal will not be perfectly reconstructed (Oakley and Cunningham[135]), but the choice of sampling scheme is independent of the array geometry.

For CCD sensors the output signals cannot be considered to be circularly band limited as the array elements are often square or rectangular in shape. However, errors will be introduced if they are considered to provide a complete square or rectangular tiling of the image plane as gaps often exist between elements, and the response to illumination of different areas within the element may be uneven.

3.4 Image Storage and Display.

3.4.1 Memory Size.

After digitisation, the sampled values are often stored in image frame sized blocks. One or more memory locations are required for each value. As shown in Section 3.2.3, hexagonal sampling is the most efficient scheme. As stated by Meserieu [124], for equal frequency information, 13.4% fewer locations will be required for a hexagonal than for a square sampled image.

3.4.2 Addressing of Memory in the Rectangular and Hexagonal Systems.

If the computer's memory is sufficient, the image can be read by the programme into a 2-D array. Increments along one image axis can be mapped "one to one" onto the first array subscript, and increments along the other axis onto the second array subscript. With the rectangular and the square

sampling systems, an increment of the array subscript corresponds to a step to the next sampled value. The corresponding distance traversed on the image is proportional to the number of times the subscript is incremented.

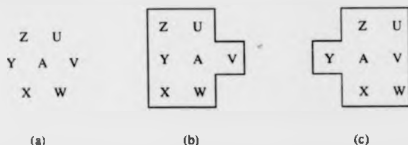


Figure 3.8: The Array Positions of the 6 Hexagonal Nearest Neighbours. (a) The Logical Positions, (b) The Odd Row Array Positions, (c) The Even Row Array Positions.

Reading a hexagonally sampled image efficiently into an array that maintains the sampling grid geometry is more difficult. The image could be read into a double sized array and half the locations left redundant, or two other schemes can be followed. In the first scheme, samples are mapped into a rectangular array, equal in size to the number of sampling points, in such a way as to fill the leftmost element of each row first. This fills memory efficiently, but the indexing of the array subscripts to step to a particular nearest neighbour is different, depending on whether the neighbour is on an odd or even row. Figure 3.8 shows the array positions of the six nearest neighbours on odd and even rows. Calculating the distance between two image points will be equally complicated. Figure 3.9 shows the image pixels annotated with their corresponding array subscripts.

In the second scheme, samples are again mapped into a rectangular array, but the indexing is referred to a 60 degree or 120 degree oblique axis. The scheme is illustrated in Figure 3.10. If the input image is rectangular in overall shape, some addresses close to the y axis (120 degrees) will be redundant and the address of the first pixel in each row must be calculated. Another disadvantage occurs if the picture boundary position must be checked during image processing. The scheme has the advantage that general pixel address calculation is independent of row number.

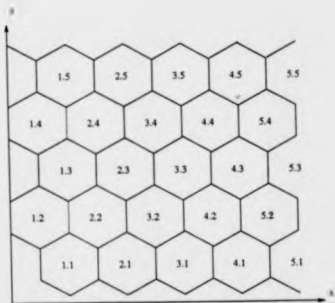


Figure 3.9: Hexagonal Pixel and Array Subscript Mapping, Scheme 1.

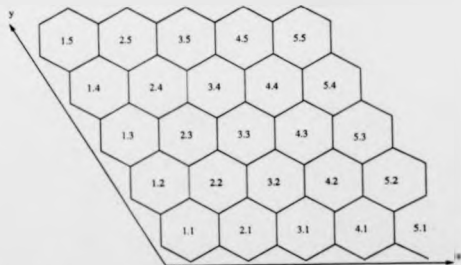


Figure 3.10: Hexagonal Pixel and Array Subscript Mapping, Scheme 2.

As seen in Figure 3.10, neighbouring pixel addresses can be straightforwardly found.

2-D arrays may not be the best way of associating adjacent pixels especially with the hexagonal sampling system. There is a solution using list processing, in which pointers are used in a 1-D array to associate hexagonal neighbours. This scheme was used in the processor element simulation presented in Chapter 9.

3.4.3 Computer Graphics.

Image processing on a hexagonal grid is covered in Chapter 4. Graphics is included here as part of the section on display. Images are often displayed by mapping their pixels one to one onto an output plane, but some image processes, such as the Hough transform (Section 4.5.1), can output position vectors, and line or curve equations. Translation and rotation transformations are also considered here.

The graphical output plane can be considered to contain a rectangular or hexagonal grid of points onto which the image locks. As noted by Serra [157], with a square grid, only image rotations of $n\pi/2$, where n is an integer, can be locked, without distortion, onto the new grid points. With a hexagonal grid, distortionless rotations of $n\pi/3$ are permissible.

With both grids, an aliasing problem can produce aberrations such as edge staircasing and the scintillation of small objects. These aberrations can be reduced if computations are made on a high resolution grid, and if the resulting image is low pass filtered and displayed on a standard resolution grid. Lestor and Sandor [101] compare the implementation of this technique on rectangular and hexagonal grids and conclude that, for negligible additional computational effort, hexagonal processing results in a reduction in the aliasing of vertical and near vertical features. Aliasing of features at other orientations remain unchanged.

Bell et al [13] have developed a method, based on complex numbers, that permits integer addressing of hexagonal grids with 60 degree oblique axes. They continue to describe a general

method for producing geometrical algorithms for such axes. Algorithms for simple shape drawing, translation, rotation and scaling are presented. These could easily be modified for 120 degree axes.

A combination of these techniques, together with the reduced number of points required with hexagonal grids, could result in a graphics system with reduced aliasing artifacts and higher computational efficiency than can be achieved with a rectangular grid.

3.5 Conclusions.

Sampling. Various sampling grid schemes can be employed when 2D images are digitised. In accordance with the sampling theorem, the spacing of the samples determines the highest "spatial frequency" that can be correctly processed by the system. For an image processing system that is used to process fine detail it can be important to circularly band-limit the image before sampling. A 2D circular band-limiting of the signal will result in the spatial resolution of the system being equal for all orientations of the object being imaged. For example in a system designed to detect scratches in a surface, fine scratches will be equally detectable, independently of their orientation.

If 2D signals are circularly band-limited then the hexagonal sampling grid is more efficient than the square as 13.4% fewer samples are required for equal high frequency information.

Sometimes it is convenient to consider images to be tiled by pixels where each pixel contains a sampling point at its geometric centre. Pixels can be conceptually useful during processing algorithm development and can be used as simple models of scenes or display transducers. They are only simple models as it is assumed that they exhibit uniform intensity throughout their area, and that they join together at zero thickness opaque boundaries. In Chapters 5 and 6 computer generated images are used to test edge and defect detectors. In each case the initial computer generated images are produced by a program that tiles a continuous image field with pixels, and then calculates the sampling point value by averaging the brightness field contained within the pixel. Significantly

different results are obtained when a more sophisticated sensor model is used in the sampling point value calculation.

Sensors. Some image sensors have been reviewed. In the vacuum tube TV camera the sensor comprises of a large number of phosphor dots that are scanned by an electron beam with a circular cross sectional area. Such a process is likely to produce a circularly band-limited signal. However, the frequency response of this band-limiting is difficult to determine. The phosphor dot density is high, and so the major factor limiting the frequency response is likely to be the width of the scanning electron beam. It is likely that the camera manufacturer will design the diameter of the beam to be similar to the scan line spacing. The band-limiting produced is likely to be most suitable for systems with a sampling point spacing equal to the scan line spacing.

A simple model of a CCD camera is one in which the sensing area is tiled by pixels. The band-limiting is then a function of the pixel shape. For instance a rectangular CCD cell would produce a rectangular band-limit region in the Fourier plane similar to the one shown in Figure 3.4. The shape of the cell is likely to have the greatest effect on the shape of the band-limit region. Other researchers have shown that the boundaries between CCD cells are not opaque, that several cells would respond to a narrow beam of light falling on a single cell, and that the response of cells are not uniform over their entire area. CCD manufacturers could strive to produce devices which circularly band-limit the image, or band-limit the image as a perfect function of pixel shape. A hexagonal sensor shape will enable a good approximation to a circular band-limited signal, whereas a square element that produces a square band-limit region would be difficult to achieve. Kamgar Parsi et al's [93] proof that the hexagonal pixel shape results in a lower positional quantisation error is also an important consideration in sensor design. Some cameras reconstruct a 1D analogue signal for output. Such a signal can be re-sampled to produce a hexagonal grid of points. CCD cells must be arranged on a hexagonal grid to enable a direct output of the sampled values for a hexagonal

system.

Storage. Hexagonally sampled images can be efficiently stored as 13.4% fewer memory locations are required. Various schemes have been presented to enable easy indexing of the hexagonal grid. The complex integer number scheme is best suited to global image or graphics operations as distance calculations between points can more easily be achieved. The simple scheme that recognises the offset of pixels on alternate lines is best suited to local image processes as membership of the local area can more easily be defined, and is used in the pipeline processor proposed in Chapter 8 of the thesis.

Display. Hexagonal grids of points can lead to advantages for graphics display systems. There are six as opposed to four locations on to which the image can lock, and researchers have shown there can be less aliasing of features with the hexagonal grid for equal computational effort. This can be important for image processing as often a system's output will be a binary map of features presented for human observation.

Chapter 4

Image Processing.

4.1 Introduction.

In this chapter image processing is reviewed. Firstly processes are classified as being either high, mid, or low level. These levels are defined below. The discussion then centres largely on low level processes; that is those processes that operate on the image pixels and produce modified pixel outputs, as opposed to symbolic outputs such as a description of a feature.

Low level processes are then divided again into those that are global or require a large area of support, and those that are local requiring only a small area of support. Work presented by other researchers on global techniques for square and hexagonally sampled images is reviewed and the advantages of the hexagonal system processes that can produce memory savings of up to 58% and result in increased computational efficiency noted.

Local processes are widely used for industrial applications as lighting levels can be controlled to ensure a good signal to noise ratio and reliable processing with these simpler operators. Again the advantages of operators designed for hexagonally sampled images are considered. The hexagonal implementation of a small local area Gaussian filter was designed as a part of this project, as was a hexagonal system edge detector. The edge detector was compared extensively with the square

system Sobel detector, which has similar design criterion with respect to computational efficiency, edge magnitude, and angular accuracy. The results of this study can be found in the next chapter. Finally some advanced edge detectors that are more suitable for general computer vision applications and require areas of support varying in size from 11×11 to 500×500 pixels are reviewed together with some mid level processes that are useful for small defect detection and categorisation.

4.2 High, Mid and Low Level Processes.

Marr [113] presents a four level framework for vision, and this is discussed in Section 2.2.3. In this chapter, image processing, a subset of vision, is partitioned into three levels:- High, Mid, and Low level processes. Luck [108] also presents this partition and gives examples of processes that fit each level. Low level processes include the primal sketch and image pre-processing. Mid level includes the 2.5D sketch and feature extraction. High level includes 3D representation and feature classification. This partitioning is also useful in the discussion of computer architecture (Chapter 7), where certain architectures are suitable for processes at a particular level, and others for all levels.

Several low level processes are considered in the remaining sections of this chapter, and comparisons made between implementations on square and hexagonal sampling grids. Some mid level processes concerned with feature extraction, such as the Hough transform, are discussed in relation to the specific case studies presented in Chapter 6.

4.3 Global Low Level Processes.

4.3.1 General.

A global process requires information from all points within the image. For a process that calculates a new value for a particular pixel, the new value will be a function of all the image pixel values. Processes requiring large areas of support are also described in this section. The transform of an

image to the Fourier plane is a global process. It may be more computationally efficient to use the following FFT (Fast Fourier transform) processes if convolutions which require large local areas for support, are to be applied to the image.

Image filtering can be a convolution process. Consider an image to be a discrete function $f(x, y)$, that is processed by a filter with global support, $h(x, y)$. The filter output will be a function $g(x, y)$, where, if \otimes represents convolution:-

$$g(x, y) = f(x, y) \otimes h(x, y)$$

If $g(x, y)$, $f(x, y)$, and $h(x, y)$, are transformed to the Fourier plane, then:-

$$G(u, v) = F(u, v)H(u, v)$$

Using efficient FFT algorithms, it may be possible to reduce the computational problem. For example, if a 2-D, 3×3 low pass filter template such as:-

$$h(x, y) = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

is convolved with the image, $f(x, y)$, in the spatial plane, and assuming that the image borders are padded with zeros so that the complete image can be processed, then $K \cdot L \cdot M \cdot N$ multiplications are required for the convolution, where $f(x, y)$ contains $K \cdot L$ pixels and $h(x, y)$ contains $M \cdot N$ pixels. Here $M = N = 3$. For a particular, efficient, FFT algorithm employing sharing of intermediate terms, Horn [81] reports that $4K \cdot L \log_2 K \cdot L$ multiplications are required to perform the process using the Fourier plane. Comparing the two methods for a $K = L = 512$ image, if $M \cdot N > 72$ then the FFT method will require fewer multiplications than the direct method. However, integer multiplications are required by the spatial plane method, and complex floating point multiplications by the Fourier plane method. The choice is complicated by the type of computer architecture. Firstly, integer multiplications are likely to be performed more quickly than complex floating point

multiplications. Secondly, many parallel hardware configurations often favour the spatial plane convolution method [81]. Thirdly, if semiconductor memory is limited, data swapping to disk will slow the process. The efficiency of the spatial plane methods can be increased if $h(x, y)$ is separable, as it is in this example. The convolution can be performed by firstly applying

$$h_1(x) = \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}, \quad \text{and then} \quad h_2(y) = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

to each image point. The number of multiplications is then reduced to $K \cdot L(M + N)$.

4.3.2 FFT of Square and Hexagonally Sampled Data.

A hexagonal Fourier transform and HFFT (Hexagonal fast Fourier transform) have been developed by Mersereau [124, 47]. The HFFT requires 25% less storage of complex variables than the RFFT (Rectangular fast Fourier transform) and also computes more efficiently. The algorithm is based on the Rivard [150] procedure, rather than the more conventional decomposition of the 2-D kernel into 1-D FFTs method. Decomposition to 1-D FFTs is not possible in the hexagonal case. This alternative procedure is a direct extension of the 1-D FFT algorithm to the 2-D case, and can increase the computational efficiency of the RFFT by 25%. Mersereau has shown that his HFFT increased computational efficiency by a further 25% in comparison to the Rivard RFFT.

4.3.3 A Comparison of Filters Operating in Each System.

In his paper, Mersereau [124], also develops a series of hexagonal FIR (finite impulse response) filters, and compares these to rectangular filters with comparable frequency responses. He found the hexagonal filters to be superior in terms of computational efficiency, and also, since they could be designed with twelve fold symmetry, they had a more circular frequency response.

Mersereau was considering 2-D signal processing in general as opposed to just image processing. He adapted many FIR filter design algorithms to the hexagonal sampling system. Mersereau considered in detail window, equiripple (Chebyshev) and transform designs. Window designed Gaussian filters are widely used for image processing applications, these can easily be adapted to operate on hexagonally sampled images. The 12 fold symmetry inherent in the sampling system eases the design procedure and leads to efficient computation. FIR filters can be implemented using direct convolution or FFT algorithms. Mersereau reports savings of up to 58% in memory and similar gains in computational efficiency for hexagonal filters compared to their rectangular counterparts.

4.4 Local Processes.

A local process is one that requires support from only a small part of the image. These include convolutions that are more efficiently processed by a direct spatial plane method, some classes of non-linear transforms, a class of distance metrics, and a set of connectivity operators. These processes are listed in the following sections under the function for which they are used. Operators for both binary and grey level images are considered here.

4.4.1 Processes Implemented on Square and Hexagonal Systems.

4.4.1.1 Connectivity.

In determining if a group of, say, binary valued pixels are connected together to form an object, a definition of connectivity must first be stated. On a hexagonal grid, all neighbouring sampling points, with associated pixels touching a central pixel, are equidistant from the central sampling point. If the pixel shape is hexagonal then all the nearest neighbours touch the central pixel along equal length sides. This scheme is known as six-connectedness. Hexagonal grids with rectangular

pixels, as shown in Figure 3.5 can also be defined as six-connected.

On a rectangular grid, there are four nearest neighbouring pixels, but four additional pixels touch the central pixel at each corner. There are two definitions of connectivity:-

- Four-connectedness, where only edge adjacent pixels are neighbours.
- Eight-connectedness, where corner adjacent pixels are also considered as neighbours.

A problem arises since the connectivity of the background pixels can also be considered. Now, if the four-connected definition is used on both foreground and background, some pixels will not appear in either set. A simple closed curve should be able to separate the background and object into distinct, connected regions, but this is not the case. Again, if the eight-connected definition is used, some pixels will appear in both sets. One solution is to use four-connectedness for the object, and eight connectedness for the background. Another is to define a six connectedness that involves just two corners.

The hexagonal systems' unambiguous definition is more convenient. Connectivity is an important consideration in many image processes, especially where groups of pixels are being considered for membership of a particular feature, or the edges of a feature are being traced out and coded. Rosenfeld [152] explores the use of connectivity in shrinking and edge following algorithms. In a more mathematical paper, Mylopoulos and Pavlidis [126] consider the more general topological properties of digitised spaces, and in particular connectivity and the order of connectivity.

4.4.1.2 Filters.

Filters can be classified according to the algorithm used to generate them, the band of frequencies passed, the filter order, the region of support, linearity, whether they are discrete or continuous, FIR or IIR, etc etc.

Some of the types, commonly used for image processing, are listed here by image processing

function. Comparisons are made between rectangular and hexagonal implementation. In general, the regular hexagonal structure leads to easy spatial plane local operator design. The local area can be defined to include the central pixel and any number of concentric "shells" of pixels at increasing distances from the centre. All the members of a particular shell are equidistant from the centre and can be assigned equal weighting factors in many local operator designs. For example, consider a local Gaussian approximation filter. The application of such a filter to an image was considered earlier in Section 4.3.1. For a three shell filter operating on a hexagonal grid, three weighting factors are initially calculated, as shown in Figure 4.1, and the final algorithm will be of the form of Equation 4.1. In comparison a similar filter on a square grid (5x5) requires six different weighting factors and a correspondingly more complicated algorithm of the form of Equation 4.2.

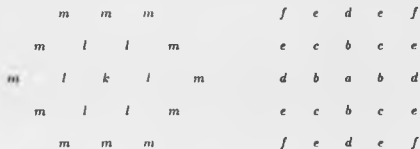


Figure 4.1: 3 Shell Hexagonal and 6 Shell Square Local operators.

$$P_H = k i_{1,1} + l \sum_{p=1}^{12} i_{2,p} + m \sum_{q=1}^{18} i_{3,q} \quad (4.1)$$

Where *k*, *l*, and *m* are filter weights associated with the three shells, and *i* are image points. Three multiplications and nineteen additions are required for the computation of each output pixel.

$$P_S = a i_{1,1} + b \sum_{p=1}^4 i_{2,p} + c \sum_{p=1}^4 i_{3,p} + d \sum_{p=1}^4 i_{4,p} + e \sum_{p=1}^4 i_{5,p} + f \sum_{p=1}^4 i_{6,p} \quad (4.2)$$

Where *a*, *b*, *c*, *d*, *e*, and *f* are filter weights associated with the six shells. Six multiplications

and twenty five additions are required for the computation of each output pixel.

Both filters have a similar region of support, but, in general, 13.4% fewer points will be required for the hexagonal filter than for the square. However, in this case, the square system operator kernel is separable, giving an alternative computation algorithm of the form of Equation 4.3.

$$P_5 = [a_{1,1} + b(t_{2,1} + t_{2,3}) + d(i_{4,1} + i_{4,3})] \otimes [a_{1,1} + b(t_{2,2} + t_{2,4}) + d(i_{4,2} + i_{4,4})] \quad (4.3)$$

Now, six multiplications and eight additions are required for the computation of each output pixel. The hexagonal operator requires only three multiplications, compared with six for both rectangular algorithms, but the number of additions is larger than that for the separable kernel rectangular method. Computational efficiency will be determined by the architecture of the computer's arithmetic and logic unit, and depend upon whether the filter coefficients are integer or real numbers.

Low Pass Linear Filters. Low pass filters are used to suppress image noise, and to remove detail in multi-resolution images, such as those used in pyramid processors (see Section 7.4.4). In the simplest low pass filter, the new output pixel is the average of the points in a local area surrounding the old pixel value. However, as illustrated graphically by Gonzalez and Wintz [65], this results in "ringing", a wavlike intensity superposition, within the image. More sophisticated designs, for example the Gaussian filter, remove this effect. Examination of the Gaussian filter profile indicates that the filter requires infinite support, but here, we are concerned with the identification of approximate filters for local areas of both square and hexagonal grids. Davies [37] has designed optimum Gaussian filters for both 3x3 and 5x5 square grids. His methods can be adapted for different sized regions, and to design filters for hexagonal regions. The effect of a larger filter can be achieved by repeated application of the smaller filter. However, any errors in the smaller filters

will be compounded

Davies observes that the 2-D, isotropic, Gaussian operator does not match well to a digital lattice. Two problems occur. Firstly, the filter support outside the local area must be truncated, and secondly, the Gaussian profile can vary rapidly within the space of the individual pixels. The magnitude of these problems depends on the width or standard deviation of the profile. If a general 3×3 Gaussian filter is represented by G_0 , then a measure of the width can be given by the ratio a/b .

$$G_0 = \begin{matrix} c & b & c \\ b & a & b \\ c & b & c \end{matrix}$$

Davies calculates an optimum filter, with a moderate standard deviation, that minimises the effect of these two divergent problems. This filter has an a/b ratio of 2.22. By relaxing these optimum filter coefficients, Davies identifies an optimum, integer coefficient only, filter. This is included here:-

$$G_1 = \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}$$

Filter isotropy is an important consideration. The filter must cause minimal distortion of image features, such as edges and lines. Davies continued to calculate a set of filter coefficients that will minimise anisotropic errors. This filter has an a/b ratio of 2.33. The integer form of this filter is included here:-

$$G_2 = \begin{matrix} 3 & 7 & 3 \\ 7 & 16 & 7 \\ 3 & 7 & 3 \end{matrix}$$

Davies states that either G_1 or G_2 would be sufficient for most applications. In the case studies presented in Section 6, G_1 was used mostly, as defect detection was paramount, rather than defect measurement. Two non-optimised hexagonal filters were used, the first had the same a/b ratio as $G_1 = 2.22$:

$$G_3 = \begin{array}{ccc} & 2 & 2 \\ 2 & 5 & 2 \\ & 2 & 2 \end{array}$$

It should be noted that the pixel spacing is a factor of $2/\sqrt{3}$ larger on the hexagonal grid. The second filter was less accurate but slightly quicker to compute as only one multiplication was required:-

$$G_4 = \begin{array}{ccc} & 1 & 1 \\ 1 & 3 & 1 \\ & 1 & 1 \end{array}$$

The filter outputs $G_1 - G_4$ were usually divided by the sum of the coefficients to ensure that the output image values remained within the same range as those of the input image. Filter G_1 can be applied more efficiently if its kernel is separated into the one dimensional array $\begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$. This is then applied sequentially, horizontally and vertically to the image. This requires 2 multiplications and 4 additions per pixel. G_3 requires 2 multiplications and 6 additions, whereas, G_4 requires one less multiplication. Here, the G_1 square filter is more computationally efficient per pixel than the hexagonal, although there are 13.4% fewer pixels required with the hexagonal grid. Anisotropic errors will be less for the hexagonal design as there are more axes of symmetry.

Hexagonal equivalents to Davies 5×5 filters were not developed, instead G_3 and G_4 were multiply applied to produce the same result.

High Pass Linear Filters. Local high pass linear filters are discussed in text books such as Gonzalez and Wintz [65] and Batchelor et al [11], and can be used to enhance object boundaries to make features stand out from the background. Designs presented there can be readily adapted for use on hexagonal grids.

Non-Linear Filters. This class of filters includes designs such as the Median filter, discussed below, and Morphologic filters, which are discussed in Section 4.4.1.5.

Median Filters. Pre-filtering noisy image data with this type of filter is a useful technique, but they can only be applied in the spatial domain. False edge detections due to image noise are reduced, and distortions of edge position minimised. A small area surrounding the central pixel is investigated and the members of the area ordered in terms of brightness. The median or central element of the ordered string is used to replace the central pixel value. Bovik et al [16] have reported the history of the technique and investigated the shape and size of the local area required for square sampled images. For images containing a preponderance of edges that do not line up preferentially with any particular direction, they found an isotropic, square area to be best. Whereas, a "+" shaped area was better for images with a preponderance of vertical and horizontal edges, and a "x" shaped area for images with a preponderance of diagonal edges.

Sorting the pixel values to find the median is a sequential procedure for each value, and is computationally intensive. Pseudo median algorithms, that avoid the search, such as one developed by Pratt [143], have been proposed and found to work reasonably well. Various authors have suggested algorithms to speed up sorting. Huang [84] developed a running median method, that starts with a value based on previous pixel calculations. However, if there is a large high frequency component in the image, the algorithm may be no faster than a conventional sort. Algorithms based on examining the bit content of variables, and successive approximation techniques, have been

refined by Danielsson [33]. His algorithm speeds computation significantly and can be implemented on special purpose parallel hardware. This hardware is discussed further in Section 8.2.3.4.

Hexagonal grid, median filters, should be more computationally efficient than their square grid counterparts, as, for the same area of support, 13.4% fewer values exist. This will significantly simplify the sorting procedure.

4.4.1.3 Edge Detectors.

Edges correspond to intensity discontinuities in the image. These discontinuities may correspond to the edges of an object, but unfortunately, sometimes they do not. For example the edge of a shadow is likely to be detected

Differential Operators. Differential operators model local edges by fitting the best plane over a convenient size of neighbourhood. In square arrays two orthogonal operators are applied to a pixel and from the response of these, the magnitude, m , of the gradient of the plane and the edge angle, α , can be calculated. The magnitude depends on the brightness difference from one side of the edge to the other and measures the significance of the edge. The edge angle is the angle the edge of the feature makes with the horizontal image axis.

$$m = (th^2 + tv^2)^{1/2} \quad (4.4)$$

$$\alpha = \arctan(tv/th) \quad (4.5)$$

Where tv and th are the responses of operators designed to respond maximally to vertical and horizontal edges. Figure 4.2 shows two such operators designed to be convolved with a 3x3 area of the image. For edge detection, the response magnitude is compared with a threshold to determine if

a significant edge exists. Automatic thresholding techniques have been comprehensively surveyed by Weska [186], for both global and local image areas. She discusses the application of the methods to the automatic thresholding of the edge detector magnitude output. For the edge detection results presented later in this thesis a manual technique was used in which the threshold level was set so that connected edges resulted that were of the minimum width possible. This technique allowed the edge detector to be evaluated in isolation from any automatic thresholding algorithm.

$$\begin{array}{ccc}
 1 & 2 & 1 \\
 0 & 0 & 0 \\
 -1 & -2 & -1
 \end{array}
 \qquad
 \begin{array}{ccc}
 1 & 0 & -1 \\
 2 & 0 & -2 \\
 1 & 0 & -1
 \end{array}$$

Figure 4.2: 3x3 Sobel Differential Operators.

Much has been published on the design of square local operators [35, 71, 58]. Each author offers an optimal set of weighting factors. In particular, Davies [35] develops a principle for the design of accurate edge orientation operators and proves the Sobel [154] operator to be very nearly optimal. The Sobel operator was extensively used during this project and was found to consistently produce good edge detection results. It would appear to be the optimum square system operator for use in controlled lighting conditions. The advanced edge detectors discussed in Section 4.4.2.1 perform better in lighting conditions that are not ideal or where sub-pixel edge positional accuracy is required. The Sobel operator has a computational processing time advantage over some other operators as only integer arithmetic is required and the local area in which it operates is relatively small.

Template Matching Operators. A set of masks which respond maximally to edges at various orientations are applied to the local area. The mask with the maximum response, m , gives the magnitude and orientation of the edge.

$$m = \max\{|m_i| : i = 1 \rightarrow n\} \quad (4.6)$$

Where n is the number of masks. For the square system, a set of square masks based on the Sobel operator, exist which will respond maximally to edges at 0, 45, 90 and 135 degrees to the horizontal [186]. The angular error can be 22.5 degrees and, even for a well designed template, the magnitude response can be a $1/(2^{1/2})$ underestimate. As reported in Section 5.1.3 differential operators can have an angular accuracy that is an order of magnitude better than this, and enable an appreciably more accurate magnitude response measurement.

Hexagonal Local Edge Detection Operators. The regular hexagonal data structure leads to easy local operator design. The central element of the local area is surrounded by shells of elements. Figure 4.3 shows a set of edge detection operators exploiting only the inner shell of neighbours, and these are of a comparable order to the 3×3 operators in Figure 4.2. These hexagonal operators will respond maximally to edges at 60 degree angular intervals from the horizontal. The weighting functions of the shell elements are chosen as 1 or -1 to reflect the regular structure of the grid of sampling points. As calculated in Section 5.1.2.2, Davies' [35] design principle also indicates "1" to be nearly optimal. Again only integer arithmetic is required for computation. If these masks are used as differential operators, the slope magnitude, m becomes relatively complicated compared with Equation 4.4. The equation of m is derived as follows.

The output of each of the three hexagonal operators, as shown in Figure 4.3, can be represented as a vector. An edge can be modelled by a plane [71], and the three vectors, \vec{r}_1 , \vec{r}_2 , \vec{r}_3 , lie within this plane. Assuming orthogonal x and y axes, \vec{r}_3 is aligned with the y axis, \vec{r}_1 is at 60 degrees to \vec{r}_3 , and \vec{r}_2 at 60 degrees to \vec{r}_1 . The resultant vector, \vec{m} , can be found:

$$\vec{m} = \vec{r}_1 + \vec{r}_2 + \vec{r}_3 \quad (4.7)$$

Examination of Figure 4.3 indicates the simple relationship $t_3 = t_1 - t_2$, giving:

$$\vec{m} = \frac{\sqrt{3}}{2}(t_1 + t_2)\vec{x} + \frac{3}{2}(t_1 - t_2)\vec{y} \quad (4.8)$$

The slope magnitude, m is:

$$m = [3(t_1^2 + t_2^2 - t_1 t_2)]^{1/2} \quad (4.9)$$

The angle that \vec{m} makes with the x axis is known as the edge angle, α :

$$\alpha = \arctan \left(\frac{\sqrt{3}(t_1 - t_2)}{t_1 + t_2} \right) \quad (4.10)$$

In template matching operations the three masks give an angular accuracy of between +30 degrees and -30 degrees and the magnitude response can as much as $\sqrt{3}/2$ times underestimated.

$$\begin{array}{cccccc} 0 & 1 & & -1 & 0 & & 1 & 1 \\ -1 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & & 0 & 1 & & -1 & -1 \end{array}$$

Figure 4.3: Hexagonal Differential Edge Detection Operators.

A comparison between the computational efficiency and accuracy of local edge detection operators in the two systems is made in Chapter 5. The hexagonal system detector was found to compute more efficiently than the square system Sobel detector as the mask weights are fewer in number and all unity. The accuracy of the two detectors were found to be equivalent, with the hexagonal being more accurate with one type of sensor model, and the square more accurate with a second.

Visual Appearance of Edges. An edge can be displayed by highlighting the pixels through which it passes. In a square system, highlight pixels will connect edge to edge, or corner to corner, whereas

in a hexagonal system connection is either end to end, or half edge to half edge. This can be seen in Figure 3.5. A human observing a vertical edge in the hexagonal system will see a ragged edge, but the easily defined pixel connectivity will simplify machine edge extraction.

4.4.1.4 Line Thinning and the Skeleton of an Object.

The skeleton or medial axis of a shape can be used as a basis for object recognition. In particular it is often used in optical character recognition systems. There are several steps involved in the process:-

- **Thresholding:** The grey-level image is converted to a binary image in such a way as to maintain the shape.
- **Thinning:** The shape, which may have a width of several pixels, is analysed, or eroded, to find a one pixel thick line that fits centrally within it.
- **Line tracking:** The thinned lines are chain coded.
- **Line segmentation:** The chain coded information is converted to vector form. This point is the limit of the skeleton forming process. Subsequent processes analyse the vectors to identify junctions and then the object.

Variations on this procedure exist and a large number of algorithms have been developed for the processes at each step. Smith [164] has surveyed and categorised many of these algorithms. In the remainder of this section, algorithms developed for hexagonal grids are reviewed, with particular reference to thinning algorithms. An example of equivalent thinning algorithms operating on square and hexagonal grids is included in Section 6.3. Two main classes of thinning algorithm exist [164], iterative and methodical. In iterative methods, such as Chin's [27], a local area on the edge of the object is examined, and the central pixel of the area removed if certain rules, designed to

preserve the connectivity of the final skeleton are obeyed. The process is repeated on the image in a way that removes pixels equally from both sides of the object until no further pixels are changed. The resulting skeleton is connected, a subset of the original objects pixels, and can be sufficient for many recognition tasks. Jang and Chin [89] have used mathematical morphology to formally define thinning, and have produced a set of operators that are proved to produce single pixel thick, connected skeletons. However, this resulting skeleton may lie only approximately in the correct place. A second set of algorithms, known as methodical algorithms, aim to ensure a correctly positioned skeleton.

Methodical algorithms start with a formal definition of a skeleton. The shape of the perimeter is found, and then rules applied to identify skeletal points. However, the resulting skeleton may be disconnected, and so Davies and Plummer [39] have developed an algorithm with an initial methodical stage, followed by an iterative process on the original image to fill in gaps in the skeleton. The purely iterative methods can produce sufficiently accurate skeletons for most applications and as they compute efficiently and can be easily realised using local operators, they have been applied to many of the problems reported later in the thesis.

Deutsch [43] reports similar thinning algorithms developed for use with rectangular, hexagonal, and triangular arrays, and has compared their operation. The "triangular" algorithm produced a skeleton with the least number of points, but was sensitive to noise and image irregularities. The "hexagonal" algorithm was the most computationally efficient, produced a skeleton with fewer points than the "rectangular", and was easily chain coded. He concluded that of the three algorithms, the "hexagonal" was optimal. His conclusions are confirmed in Section 6.3, where Chin's [27] algorithm implemented on a rectangular grid, is compared with a new version implemented on a hexagonal grid that was developed as a part of this project. A further conclusion that the "hexagonal" algorithm can be less sensitive to image irregularities is made.

4.4.1.5 Morphological Operators.

These operators provide important additions to the tool box of local image operators available for machine vision applications. In this thesis their only application is to provide a smoothing filter prior to a line thinning process reported in Chapter 6, but their wider use in applications such as printed circuit board analysis [110] is well established.

Mathematical morphology is an approach to computer vision based on searching for the shape of an object or the texture of a surface. Morphological operators are repeatedly applied to the image to remove irrelevant information and enhance the essential shape of the objects within the scene. These methods are based on set theory. Highly mathematical justifications of the methods are presented by Matheron [116] and by Serra [157, 158]. An easier introduction is provided by Haralick et al [72] which concentrates on computer vision and visual inspection. Haralick's formalism, which uses a minimum of the available set operator symbols is reported here.

The scene, continuous or discrete, binary or grey-level, is one set. A typical morphological transform will take a small set, the members of which form a particular shape, known as a structuring element, and apply this as an operator throughout the image. Typical set operations performed at these points include dilation and erosion.

In dilation, two sets A and B , are combined using vector addition. Assuming the sets are in E^N space (E^N), and have elements $a \in \{a_1, \dots, a_N\}$ and $b \in \{b_1, \dots, b_N\}$, then the dilation of A by B , $A \oplus B$, is defined by:-

$$A \oplus B = \{c \in E^N | c = a + b \text{ for some } a \in A \text{ and } b \in B\} \quad (4.11)$$

As an example, consider $A = \{(0, 1), (1, 0), (2, 1), (2, 2), (3, 2)\}$ dilated by $B = \{(0, 0), (0, 1)\}$.

$$A \oplus B = \{(0, 1), (0, 2), (1, 0), (1, 1), (2, 1), (2, 2), (2, 3), (3, 2), (3, 3)\}$$

This can be shown graphically ((0, 0) is at the top left corner of each array):-



The dilation has added an extra • to the right of each original •. With binary image processing, the structuring element, " B ", is often square or disc shaped, in a 3×3 square or similar sized hexagonal area. This allows an isotropic expansion of image objects.

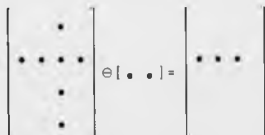
Erosion is the morphological dual of dilation, here the two sets A and B , are combined using vector subtraction. Erosion, $A \ominus B$, is defined as:-

$$A \ominus B = \{x \in E^N | x + b \in A \text{ for every } b \in B\} \quad (4.12)$$

As an example, consider $A = \{(1,0), (1,1), (1,2), (1,3), (0,2), (3,2), (4,2)\}$
eroded by $B = \{(0,0), (0,1)\}$.

$$A \ominus B = \{(1,0), (1,1), (1,2)\}$$

This can be shown graphically ($(0,0)$ is at the top left corner of each array):-



The erosion has removed each • that does not have a • to the right of it. Again, with binary image processing, the structuring element, " B ", is often square or disc shaped, in a 3×3 square or similar sized hexagonal area. This allows an isotropic contraction of image objects.

Openings and closings are morphological filters that eliminate detail, but leave the major image

features unaltered. Opening an image with a disc structuring element smooths an object's perimeter, breaks narrow isthmuses, and removes small blobs and spikes. Such a filter, implemented for both a hexagonal and a square system, is used in the printed character recognition example, in Section 6.3, to remove spikes prior to skeletonisation. Opening is performed by eroding and then dilating the image, and is defined, for an image A , and a structuring element K , as:-

$$A \circ K = (A \ominus K) \oplus K \quad (4.13)$$

Closing an image with a disc structuring element smooths an object's perimeter, bridges narrow gaps, and fills small holes and perimeter cracks. Closing is performed by dilating and then eroding the image, and is defined, for an image A , and a structuring element K , as:-

$$A \bullet K = (A \oplus K) \ominus K \quad (4.14)$$

Grey-level Morphology. The binary operations listed previously can be extended to grey-level images. Haralick et al [72] have introduced the basic concepts, and Sternberg [165] provides a comprehensive review of grey-level operators. Some operators, for instance the local maximum and minimum operators, have similar implementation algorithms to rank order filters. Maragos [111] presents a unifying theory for some morphological and signal processing operators. Shih and Mitchell [161] note that some grey-level operators are difficult to implement in real-time. They present a method for decomposing the image into multiple binary signals, which allows for parallel processing of the signals with binary logic gates integrated on VLSI devices. The results are then recombined to produce a grey-level result.

Morphology and Hexagonal Sampling Grids. Hexagonal sampling grids and morphological image processing have been strongly linked since their first inception. Golay [63] and Preston [144]

present work published in 1969 and 1971 on hexagonal parallel pattern transformations involving morphological operations. Their main reason for choosing hexagonal sampling was to avoid the ambiguous connectivity definitions between pixels on a square array. Serra [157, 158] makes extensive use of the hexagonal grid, preferring it to the square owing to the connectivity definition, its large possible rotation group on the grid, and the simple processing algorithms that result.

4.4.1.6 Distance Functions.

Distance metrics on digital pictures have been devised by Rosenfeld and Phaltz [155]. The functions are local operations that can be performed in parallel on every pixel, and are used to identify blobs and dissect a region into sub-regions. In a following paper, Luczak and Rosenfeld [109] extend the theory to the hexagonal grid and conclude that the resulting simpler functions compute more efficiently and produce more accurate results.

4.4.2 Processes Only Implemented on the Square System.

4.4.2.1 Advanced Edge Detectors.

Analysing the edge detection problem, it can be seen that an edge, a boundary between different segments of an image, is required to be accurately known, to be detected in the presence of noise introduced by the sensor and digitisation process, and to be continuous along the boundary, even as the surface illumination changes throughout the image. Edges due to specular reflections, shadows, or changes in surface reflectance may not be required. The spatial position may need to be known to sub-pixel accuracy.

The operators described in Section 4.4.1.3 were first order operators that included a certain amount of smoothing. They require support from only a small local area and compute efficiently, but they tend to produce false edges, require a thresholding stage, and produce double pixel thick lines even when the threshold value is optimum. Industrially, the signal to noise ratio of the image

can usually be improved by careful illumination design, and so these operators are widely used. More advanced edge detectors are introduced in the remainder of this section so that their implication to computer architecture can be assessed.

Recent authors have studied the edge detection problem and suggest a variety of solutions. However there are some common themes. Many authors suggest the use of optimum smoothing before or in conjunction with the detection process. Marr and Hildreth [114], and Canny [21, 20] describe the Gaussian filter to be optimum, and Torre and Poggio [174] describe it as being nearly optimum. As noted by Horn [81], it is the only rotationally symmetric operator that is the product of two 1-D operators, and so is computationally convenient. Conversely, Nalwa and Binford [127] claim that simple averaging is more efficient for smoothing when, as is usually the case, additive Gaussian distributed noise is present in the image. However, to avoid large errors resulting from Gibbs's phenomenon the averaging must be performed parallel to the edge and not across it. This approach then has the added advantage that the edge profile itself is not smoothed. However, another stage has been added to the edge detection process, the orientation and position must be estimated before the filtering and final detection stages.

Second derivative edge operators are widely used in advanced detectors as their output is zero where the edge is located. Marr and Hildreth [114] use a Laplacian operator combined with a Gaussian smoothing filter to produce a rotationally symmetric operator. Subsequent researchers [81, 174] claim that the detected edges do not locate correctly with the relevant image features. Marr and Hildreth recommend support for the operator from 500 pixel wide regions. Huertas and Medioni [85] developed the Laplacian of Gaussian mask further by realising an exactly separable operator. A typical operator would require a support region with a width of 12 pixels and sub-pixel accuracy is obtained by using a facet model. This model is applied after the zero crossings, which give good localisation of the edges, have been detected. The facet operators are interpolation functions. With its relatively small support region and exactly separable operator, their detector would appear to be

more easy to implement efficiently than the others reported in this section.

Nalwa and Binford [127] use a variant of a surface fitting approach in which 1-D oriented surfaces are fitted, in a least squares sense, to the edge. Edge-lets, that are not locked to the pixel grid, are then identified. The method permits sub-pixel resolution, and computationally efficient implementation. Their paper covers the entire edge detection process and contains novel processes for filtering, measurement and exact position finding stages.

Canny [20] developed an operator with an optimal trade off between localisation and detection. Detection of an ideal step edge is favoured by a broad filter, whereas localisation is favoured by a small filter. For 2-D step edges, Canny uses directional operators of varying width, location, and length, together with an adaptive thresholding scheme. These operators cope with different signal to noise ratios in the image. The operator outputs are then fine to coarse integrated to produce the final output information.

Micheli et al [41] analyse two aspects of edge detection, localisation accuracy and noise sensitivity. In particular they investigate the detection of corners and trihedral vertices. They confirm that for indoor scenes, or scenes where the lighting conditions are controlled, the signal to noise ratio of the images are high, and that small support region filters are sufficient for smoothing.

Many of the authors of the papers discussed in this section have claimed their particular edge detection method to be optimum. They may mean optimum in the accuracy of the position of the edge, the height of the edge, or the angular orientation of the edge. For an application requiring sub-pixel positional accuracy, the most computationally efficient method that will give the required accuracy and be robust in the presence of the expected image noise level must be chosen. The Canny [20] detector would appear to be a likely candidate to investigate initially, as the reasons for his choice of operator at the individual processing stages have been compellingly argued with regard to practical image problems.

However, none of these advanced detectors have been implemented as a part of this project.

Micheli et al's [41] argument was accepted that in a controlled lighting environment, the signal to noise ratio of the image is high enough for detectors such as the Sobel, which require only small support regions, to be sufficient. The practical results presented in later chapters of this thesis indicate this view to be correct.

4.5 Mid Level Processes.

The mid level processes surveyed in this thesis are all concerned with feature extraction. In Chapter 6, which includes two case studies, mid level processes concerning optical character recognition and surface defect detection are discussed. Connected component counting and the Hough transform were implemented in the defect detection example.

4.5.1 The Hough Transform.

The Hough transform is a method for detecting lines, curves or complicated shapes in binary or grey-level image data. When this data is transformed to the Hough plane, with a transform specific to the shape being sought, the x-y plane pixels, comprising the shape, form a peak point in the Hough plane. Detection is then a matter of peak or point finding in this plane. The method is tolerant of image noise and missing data. The Hough transform has been surveyed by Illingworth and Kittler [86]. Some points relevant to this thesis are included here.

The Detection of Straight Lines in Binary Images. The original method due to Hough [82] involves transforming each x-y point into a straight line in the Hough plane. If a set of x-y points forming a straight line, are transformed to the Hough plane, the resulting set of lines will intersect at a point. The Hough plane is implemented as an array of accumulators, so that each line that passes through an accumulator will increment it. Points of intersection are detected by thresholding the Hough plane accumulators. Hough's original transform was based on an m-c parameterisation,

where m and c are given by the straight line equation $y = mx + c$. However, this has a singularity for lines where $m \rightarrow \infty$. Duda and Hart [45] suggest a $p - \theta$ parametrisation to avoid this problem. Where p is the length of the normal to the line in the $x-y$ plane, that passes through the origin, between the origin and the line, and θ is the angle of the normal to the horizontal. θ can be found from the angle of the line, and p is given by equation 4.15.

$$p = x \cos \theta + y \sin \theta \quad (4.15)$$

Now each point in the $x-y$ plane transforms into a sinusoid in the $p - \theta$ plane. Again, intersection points in the $p - \theta$ plane indicate straight lines in the $x-y$ plane. Design problems include the choice of resolution of the $p - \theta$ plane and the peak finding technique. A simple threshold was used to detect peaks in the sand core example in Section 6.2. Detected peaks can then be transformed back to the $x-y$ plane and overlaid on the original image. Figure 6.19, which highlights the detected long straight lines of an object's edges to distinguish them from the shorter lines comprising some surface defects, is an example of such a display. With the length of line, $x-y$, and $p - \theta$ resolutions in this example, the detected line angle was accurate to ± 5 degrees and the line position to within ± 2 pixels.

The Detection of Curves and Shapes In Binary Images. Duda and Hart [45] continue to consider circle detection using a parametric representation that transforms $x-y$ points to surfaces in a 3-D Hough plane. A point where several surfaces interconnect will indicate a circle. Yuen et al. [190] present a comparative study of Hough circle finding methods. Sklansky [163] developed the technique further to the detection of any specified subset of points or an arbitrary translation of the subset. He continued to generalise this procedure to grey-level images. Stockman and Agrawala [166] have shown that Hough curve detection is equivalent to template matching.

The Generalised Hough Transform. The transform has been generalised to detect lines, circles, and parabolas in grey-level images, and Ballard [7] presents a generalisation to detect arbitrary shapes. Ballard starts with the binary edge detected image, but retains the magnitude and orientation information obtained by the edge detection operator. The directional information makes computation more efficient and improves the accuracy of the technique. The grey-level gradient components g_x and g_y enable one to one mappings between the x-y and Hough planes, as opposed to the one to multi-dimensional surface mappings common in previous schemes. Davies [36] presents a transform for straight edge detection that produces a parameter space that is congruent with the image space. By exploiting various techniques, he obtains a line orientation accuracy of ± 1 to 2 degrees and line location to within one pixel accuracy. However Ballard's original, or another suitable transform must be reverted to if features that are not straight are to be detected.

4.6 Conclusions.

Image processes can be arranged into high, mid, or low level groups. Higher level processes provide descriptions and classifications of image features, whereas low level processes calculate new values for pixels within the original, or a different resolution spatial grid. Mid level processes, such as the Hough transform, take pixel values as input and produce information that high level processes can efficiently operate on. Higher level processes have only been included in this chapter if they are used in examples reported later in this thesis. This chapter concentrates on low level processes as their operation and performance depends on the sampling grid employed.

Low level processes can be divided into global and local groups. Global processes require information from all the pixels within an image to calculate each new pixel value. Previous researchers have developed hexagonal grid global processes, such as an FFT and various filters, that are more efficient than their square system equivalents. Local processes only require information

from a small area of pixels surrounding the pixel to be modified. Many square grid local operators exist, but previously gaps existed in the range of hexagonal grid operators available. Several new hexagonal grid operators have been reported here. Local operators are appropriate for use in controlled lighting conditions such as are often found in industrial environments. In these conditions the signal to noise ratio of the image is likely to be high and the changes in illumination throughout the scene minimal. Local operators are usually easier to calculate than their global equivalents and this results in lower computational overheads.

Generally, hexagonal low level local operators were found to be easier to design than their square grid equivalents as there is only one connectivity definition, there are fewer equidistant shells of neighbouring pixels within a given local area resulting in a smaller set of coefficient values, and there are more axes of symmetry within the grid.

Local low pass filters are likely to be used before an edge detection operation to reduce the effect of noise which can lead to false edges being detected, or in image enhancement operations. Many authors of papers on advanced edge detectors argue that a Gaussian profile filter is optimum for removing noise while maintaining the positional and height accuracy of the subsequently detected edge. Local operator approximations to Gaussian filters exist. Two 3x3 square grid filters with different coefficient values have been reported and hexagonal grid equivalents developed. Results of tests on these filters are presented in Section 9.4.1 where it is concluded that the local area Gaussian filters can remove noise optimally for many image processing applications. An alternative local low pass filter, the median filter, may perform optimally in certain applications, especially if there is a predominance of edges of a single orientation in the image. If this is the case the local area can be limited to a "+" or "x" shape to help preserve the edge information while removing the image noise. High pass filters are likely only to be required for image enhancement operations.

The local area Gaussian and Median filters together with the high pass filters discussed in this

chapter should be implemented by a processor designed for industrial image processing.

Edge detectors. An edge detector must reliably detect edges between features within the image in the presence of image noise. The edge position must be accurately detected and ideally no gaps should exist in the detected edge that surrounds the particular feature. Certain mid-level processes such as the Hough transform also require the edge height (brightness differential across the edge) and the angle the edge makes with a reference axis to be accurately measured. Much research has been conducted into detectors that are classified as advanced edge detectors in this thesis. These are suitable for use in naturally light and noisy environments, and usually give the edge position accurate to within one pixel width. However, they require global areas, or large local areas to support their calculation. The smallest local area reported here for such a detector was 11x11 pixels, and all of the detectors surveyed required many computational steps in their calculation.

As reported in this chapter, in controlled lighting conditions, such as those that are often found in industrially captured images, edge detectors that require only small local areas for support and thus are more easily calculated are often sufficient. Their design has been studied further. The Sobel detector requires a 3x3 local area for support. This local area size implies that some low pass filtering is also being performed by the process. A differential detector with no smoothing would simply contain two complementary coefficient values. The Sobel detector has been found by previous researchers, and in the results presented in following chapters of this thesis, to perform adequately in the presence of typical image noise levels found in industrial images.

Other edge detectors requiring 3x3 local areas exist, but the Sobel detector has been shown by Davies [35] to produce the most accurate measurement of the edge angle and highly accurate measurement of the edge height. The Sobel detector was therefore considered further for use with the applications reported later in this thesis, and a hexagonal grid equivalent designed. Details of the design and results of performance comparisons with the square grid Sobel detector are presented

in the following chapter. The Sobel and hexagonal grid detector's mask coefficient values are all integer numbers, and this can lead to fast operator calculation. The implementation of the Sobel detector was a primary requirement of the image processor architecture reported later in the thesis.

Morphological Operators are another group of operators that are widely used for the processing of industrially derived images. Many such images are binary, and morphological techniques are often used to search for particular features within the image or to remove unwanted detail. For example a binary image of a printed circuit board can be obtained and tracks that are too close together searched for. The output of the edge detection process may be a binary edge map. With simple edge detectors the lines in the edge map may be several pixels thick. Binary morphological techniques can be used to thin these to single pixel thick line or to fill in small gaps in otherwise connected lines. Grey-level morphological operators provide similar functions for direct use on grey-level images.

The unambiguous connectivity definition has resulted in many researchers working with hexagonal grids in this area. Morphological operators should be able to be performed by a processor designed for industrial image processing.

Chapter 5

A Comparison Between Local Edge Detection Operators in Square and Hexagonal Data Structures.

5.1 System Implementation.

Comparisons are made here between the computational efficiency and accuracy of edge detectors operating within the two systems. Accuracy, which is discussed later in the chapter, is dependent on the pixel weightings of the operators [35]. These weighting values are calculated here. For template matching techniques the accuracy is dependent on the number of templates. Computation time is a function of the number of sampling points within the local area, the complexity of the operators, and the complexity of the subsequent processes. The hexagonal system is only at a disadvantage in the third of these cases, as indicated by the relative lengths of Equations 4.4 and 4.9. If a fast algorithm and a processor with a fast multiply instruction are used this is minimised. It will be shown that for some processors the overall processing time is less for edge detection with the hexagonal system.

Local operators requiring only a small area of support are not the most accurate for detecting edges in images as discussed in Section 4.4.2.1 on advanced edge detectors. However, they have been found to perform adequately for images captured in controlled lighting conditions where the signal to noise ratio is high. The small area of support enables them to be efficiently implemented on digital computers. Where a fast, economical solution is required, they may be advantageous. The tests on the square and hexagonal system edge detectors were performed on a personal computer with an Intel 80286 central processing unit. This had a fast integer multiply instruction.

Timing Routines. The edge detection task involves several processes, which are listed below, only the first three are dependent on the data structure and so only these were implemented.

- Address the pixel
- Apply the operators
- Calculate the edge response from operator information
- Threshold the response
- Identify the edge pixel

Operator Application. For both systems, optimum operators, as described earlier, were applied. These were for the hexagonal system, I_1 and I_2 from Figure 4.3, and for the square system the Sobel operators from Figure 4.2. Both sets of operators use only integer arithmetic.

Edge Response. Three edge magnitude measures were implemented for each system. Firstly, the true magnitude was found from Equations 5.1, and 5.2. These equations are developed from Equations 4.4, and 4.9.

$$M = I_h^2 + I_v^2 \quad (5.1)$$

$$M = t_1^2 + t_2^2 - t_1 t_2 \quad (5.2)$$

Secondly, an approximate method using absolute values, as outlined by Gonzalez and Wintz [65] was tried. As before t_3 is replaced by $t_1 - t_2$.

$$M = |t_3| + |t_4| \quad (5.3)$$

$$M = |t_1| + |t_2| + |t_1 - t_2| \quad (5.4)$$

And thirdly, template matching equations were developed from Equation 4.6.

$$M = \max(|t_1|, |t_2|, |t_3|) \quad (5.5)$$

$$M = \max(|t_1|, |t_2|, |t_3|, |t_4|) \quad (5.6)$$

5.1.1 Overall Computational Processing Time.

Table 5.1 shows the computation time per pixel for the above equations. The hexagonal system is faster than the square at every point of comparison, due to the faster local operator calculation. For both systems, with this processor, the approximate magnitude and template matching approaches give no computational advantage, and the true magnitude calculation gives better accuracy and computes in less time. Together, the quicker processing time per pixel (31.4%) and the reduced number of pixels required (13.4%), gives a 44.8% time saving for true magnitude calculations in the hexagonal system. If a maths co-processor had been available, all of the times displayed in Table 5.1 could have been reduced, and non-integer coefficients could probably have been used

efficiently.

Edge Vector Magnitude	Square (μs)	Hexagonal (μs)
Differential Operators		
True (Equations 5.1, 5.2)	41.31	28.33
Approximate (Equations 5.3, 5.4)	39.79	25.28
Template Matching Operators		
3 templates (Equation 5.5)		35.20
4 templates (Equation 5.6)	84.81	

Table 5.1: Edge Magnitude Computation Time for Each Pixel. Computer: Olivetti M28 with Intel 80286 Processor and no Maths Co-processor. Clock rate 8 MHz.

5.1.2 Calculation of the Edge Operator Mask Weights.

5.1.2.1 Square Grid System.

Davies [35] has developed the idea of a circular edge detection operator and used it to explain the high accuracy of the Sobel operator. An edge detector must be able to measure the edge height and orientation accurately. These parameters are required by mid-level image processes such as the Hough transform (Section 4.5.1). His method involves the concept of closed concentric bands of pixels that together form near circular local neighbourhoods. The radius of the circle can be increased so that the neighbourhoods encircling from 3×3 pixels upwards can be considered. Edge detection involves the fitting of a plane to the edge profile. For the worst case edge the edge profile is a step function. Davies has shown theoretically that a circular neighbourhood is optimum as in the analogue case there is only one way to fit a plane to a step edge within such an area, assuming both pass through the same central point. He continues to show that for the analogue case zero angular errors will result from fitting a plane to an edge of any profile within a circular neighbourhood. Angular errors are however present in the spatially discrete case. Davies attempts to minimise these in the way that the circular neighbourhood is fitted to the discrete lattice.

Davies considers the image to be tiled by square pixels and the image brightness to be equal throughout each pixel. Actual image data consists of a set of brightness values constrained at the sampling points, and it is the discontinuities at the pixel boundaries in Davies' model that lead to the errors. Figure 5.1 shows two possible choices of radii for a 3x3 circular neighbourhood. The mask weights are calculated from the area of each pixel that lies within the circle. In Figure 5.1 the ratio between the weights for the corner elements and the other non-central elements varies with the radius r .

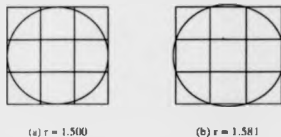


Figure 5.1: Circular Neighbourhoods of Radius 1.500 and 1.581 Imposed on 3x3 Square Pixel Areas.

Davies then calculates the values of radius r that give minimum angular errors for neighbourhoods from 3x3 to 7x7 pixels. He achieves this by applying the resulting mask for each value of r to a set of modelled edges at orientations from 0° to 90° . The edges are modelled as step edges in an image in which each pixel has been divided into 51x51 sub-pixels. Sub-pixels on one side of the edge are given one brightness value, and those on the other side, a different value. The sub-pixel values within each pixel are then averaged to give the pixel value. As r increases the angular errors are minimised at certain values. Appropriate mask weights for the various neighbourhood sizes have then been chosen. For a 3x3 operator Davies publishes the following coefficient values.

-0.464	0.000	0.464
-0.959	0.000	0.959
-0.464	0.000	0.464

These results have been stated as being achieved with radius $r = 1.500$. When they are rescaled, these values are very close to the Sobel integer weights of 1 and 2, and so the Sobel detector is considered as near optimal. Davies' results have been largely verified here. A comparison of results is given in Table 5.2 for $r = 1.500$. The weights in the column labelled "Analogue" were calculated by analysing the geometry of the arrangement, which is easy to do at this radius, and confirm the results in the previous columns.

	Davies (51x51 sub-pixels)	Staunton (65x65 sub-pixels)	Staunton (Analogue)
Weight (a)	0.464	0.543	0.543
Weight (b)	0.959	0.971	0.971

Table 5.2: Edge Detector Mask Weights for a 3x3 Operator, radius $r = 1.500$.

Staunton has used a 65x65 array of sub-pixels to minimise errors in the hexagonal system weight calculations reported below. The small discrepancy in the value of the weights calculated by Davies and Staunton is possibly not just a result of the difference in the number of sub-pixels used, but could result from a reporting mistake by Davies. The minimum errors occur at radius $r = 1.460$ and not at $r = 1.500$ as Davies reports. If Figure 3 of Davies' paper [35] is examined it can be seen that the local minimum error occurs when r is slightly less than 1.500. However, the differences in the calculated values of the weights are small and the important result is that the Sobel mask weights are close to optimum for edge angle measurement. A comparison of results is given in Table 5.3 for $r = 1.460$.

	Davies (51x51 sub-pixels)	Davies (scaled)	Staunton (65x65 sub-pixels)	Staunton (scaled)	Sobel
Weight (a)	0.464	1.000	0.492	1.000	1
Weight (b)	0.959	2.067	0.933	1.896	2

Table 5.3: Edge Detector Mask Weights for a 3x3 Operator, radius $r = 1.460$.

5.1.2.2 Hexagonal Grid System.

Davies' edge operator design principle was applied here to the design of a hexagonal system operator. The pixel shape chosen was rectangular and the size chosen to tile a rectangular hexagonal grid. As discussed in Section 3.2.3 this resulted in pixels with an aspect ratio of $2/\sqrt{3}$ to 1. As for the square grid system, the circular neighbourhood was centred at the centre of the central pixel and its radius r increased in small steps. At each step a new set of mask weights were calculated and the set that produced the minimum angular errors identified. Figure 5.2 shows the circular neighbourhood imposed on the seven member hexagonal pixel arrangement.

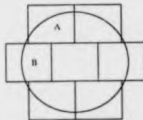


Figure 5.2: A Circular Neighbourhood Imposed on a Seven Element Hexagonal Grid Pixel Area.

The symmetry of the arrangement results in a calculation of only two mask coefficients (A and B of Figure 5.2) to give all those required. Similarly only two need be calculated for the 3×3 square operator. For the edge angle tests, each pixel was divided into 75×65 sub-pixels, as these integer numbers fit the aspect ratio almost exactly. This division results in errors in pixel area calculation of less than 0.1%. As for the square system tests the radius r was increased from near 0 and minima searched for in the angular errors produced as the edge was rotated from 0° to 90° at each r value.

Figure 5.3 shows minimum angular errors at $r = 1.35$ and when $r \geq 1.75$. The graph only covers a small range of r values compared to Davies' graph [35] for the square grid system as only a seven element operator is under consideration here. However, a similar pattern of peaks and troughs in the plot is emerging. Here the plot levels out for $r \geq 1.75$ as the circle now completely encloses all seven elements. For values of $r \geq 1.75$ the maximum angular error is 5.13° . To reduce

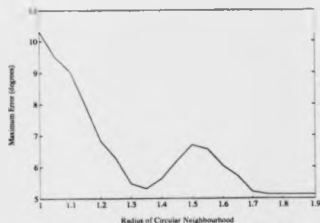


Figure 5.3: Variation in the Magnitude of the Maximum Angular Error as a Function of the Radius of the Circular Neighbourhood.

this further, elements surrounding the group of seven would have to be included and an appropriate value of r chosen. Table 5.4 lists the edge detector mask weights for a seven element operator at the two local minimum error points in Figure 5.3.

	65x65 sub-pixels $r = 1.35$	65x65 sub-pixels $r = 1.75$
Weight A	0.68	1.00
Weight B	0.67	1.00
Max. Angular Error	5.31 ⁰	5.13 ⁰

Table 5.4: Edge Detector Mask Weights for a Seven Element Operator as Indicated by the Circularity Principle.

Davies' design principle has suggested that the weights should be nearly equal. For this simple operator there is only one degree of freedom in the calculation of the ratio between the weights. If the circular neighbourhood criterion is relaxed so that the operator symmetry is kept and weight A is fixed at 1.00 while weight B is altered, then the angular errors can be further reduced as presented in Table 5.5.

This maximum angular error of 5.06⁰ is very close to the error of 5.13⁰ resulting when the

	65x65 sub-pixels
Weight A	1.000
Weight B	1.004
Max. Angular Error	5.06°

Table 5.5: Edge Detector Mask Weights for a Seven Element Operator as Indicated by a Numerical Method.

circularity principle weights are used. In both cases both the weights are very close to the integer value of 1, and so the use of integer 1 in the mask calculations would appear to be appropriate and introduce only very small errors.

5.1.2.3 Summary of Edge Detector Mask Weights.

Table 5.6 displays the optimum real and integer number mask weights that have been calculated here for the 3x3 square grid and the seven element hexagonal grid edge detection operators. The integer values have been used for all the following calculations reported in the thesis for the Sobel and equivalent hexagonal edge detectors.

	Square grid		Hexagonal grid	
	real no.	integer no.	real no.	integer no.
Weight A	1.000	1	1.000	1
Weight B	1.896	2	1.004	1

Table 5.6: Final Edge Detector Mask Weights for Both Systems.

5.1.3 Edge Accuracy.

Tests were performed to determine the magnitude and angular accuracy of the hexagonal edge detection operator employing integer only mask coefficients, and to compare these results with those obtained from the application of the near optimum integer coefficient Sobel operator on equivalent image data in the square system. Davis [40] identifies several possible edge models. The

one chosen for these tests was of a "sudden" step edge of height 10 units in a system with an intensity range of 256 units. In a real image, an edge is likely to have a slanted profile, or the operator may even be calculating from planar data. Theoretically, apart from quantisation error, neither operator will exhibit errors on planar intensity data. This was found to be true in practice. The "sudden" step edge is considered to constitute the worst case situation [35], and so is implemented here.

The model was implemented by a computer program. For the square sampling grid the pixel shape was square, and for the hexagonal grid, rectangular. The edge position was limited so that it could pass only through the centre of the central pixel. The area of this, and each surrounding pixel, situated on either side of the edge, was then calculated, together with the resulting intensity value. The appropriate operator was then applied to this data and the magnitude and angle results compared with the values used to generate the model data. The process was repeated for different orientations of the edge in one degree angular steps from 0 to 90 degrees. The gradient of the step edge, here with a height of 10 units, is measured by a difference procedure at two points separated by 2 spatial units. Hence theoretically, the gradient measured should be 5. The results are shown in Figures 5.4 and 5.5. These do not include intensity quantisation noise, which can add an additional 2 degrees of uncertainty to the angular results (256 quantisation levels).

Both the operators produce only positive magnitude errors, but both were designed to minimise angular errors. A different operator design could shift the mean error to zero. The hexagonal operator peak errors are larger than those of the square, but the errors for both systems will probably be low enough for many practical applications.

The mean value of the angular errors in both systems is zero. Again the errors in the hexagonal system are larger. The larger edge magnitude and angular errors in the hexagonal system possibly result from the reduced radius of the circular neighbourhood that can be completely enclosed within the seven rectangular pixel area in comparison to the nine square pixel area. Figure 5.1 shows that a circle of radius 1.500 can be completely enclosed within the nine element area, while the

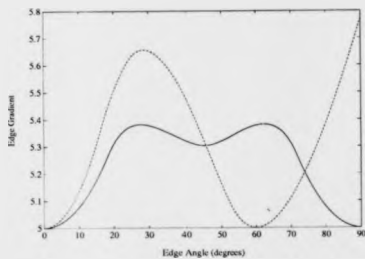


Figure 5.4: Operator Edge Gradient Response, for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$.
Key: — Square; - - - Hexagonal System.

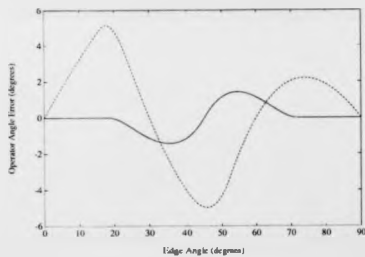


Figure 5.5: Operator Angular Error for Edges Oriented $0 \Rightarrow 90^\circ$.
Key: — Square; - - - Hexagonal System.

equivalent radius for the seven element area shown in Figure 5.2 is 1.258. If the radius of the circle is increased beyond this, as shown in Figure 5.2, then, in the first instance, four additional pixels at the corners of the present local area should be included in the edge detection masks. Including these extra pixels, whilst reducing errors, will increase the complexity of the edge detection calculations. The extra pixels may not in practice be necessary when a practical image sensor model is included in the edge modelling program as discussed in the next section.

5.1.4 Circular Band-Limiting of the Modelled Edge Data.

The previous results were obtained with an edge model that produced data that was not circularly band-limited. A step edge was constructed in a high resolution 230x230 square pixel image. To obtain the 3x3 pixel image data on which to test the Sobel detector, an appropriate square area of 65x65 high resolution pixels were averaged to give a value for each of the 9 pixels. Figure 5.6 shows such a step edge and 9 pixel area. The same high resolution data was averaged over appropriate 75x65 rectangular areas to give values for each of the 7 hexagonal system pixels. Referring back to Section 3.2 on sampling schemes, this corresponds to tiling the frequency plane with pixel shaped band regions as shown in Figure 3.4. The square grid system appears to gain an advantage from such a band region geometry, as the frequency response in each angular direction remains more constant than for the rectangular band-limit regions in the hexagonal system.

Image sensors have been discussed in Section 3.3, and it was noted that it is not always appropriate to model the sensor elements by a square or rectangular area in which each point within such an area contributes equally to the sensor's output value. Additionally, before spatial sampling, which is achieved by the sensors in systems containing 2-D arrays of sensors, band-limiting of the analogue signal is required to comply with the sampling theorem (Section 3.1). With circular band-limiting the frequency response of the system is equal for all spatial directions within the image. Circular band-limiting is important in, for example, industrial inspection applications where

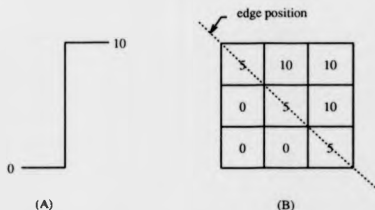


Figure 5.6: A Step Edge of Height 10 Units. (A) Profile. (B) The 2-D Image Data Resulting from Such an Edge, Oriented at 135° , Passing Through a 3×3 Pixel Area. The Pixel Values Have Been Calculated by Averaging Over a Pixel Size Square Area.

a small defect may be required to be detected independently of the object's orientation. The edge modelling program was modified so that the pixel values were calculated from circular spatial areas. Such circular areas in the spatial plane lead to circular tilings of the frequency plane and hence circular band-limiting of the image data. The diameter of these circular areas could be chosen from 0.01 to 2.00 of the square grid horizontal sampling point distance.

Figure 5.7 shows the square and hexagonal operator edge gradient responses when the diameter has been limited to 1.00 units. With this diameter the circular area just fits within one square grid pixel. The overall shape of the curves are the same as the original curves shown in Figure 5.4. The positive peak errors are larger, but the trough errors are the same as those in the original curves. At an edge angle of 45° , the square grid gradient magnitude error is 5.3, the same as with the original band-limiting scheme. If the square system measurements were in fact gaining an advantage from the increased band-width at 45° , then this error would have been greater for the circularly band limited case.

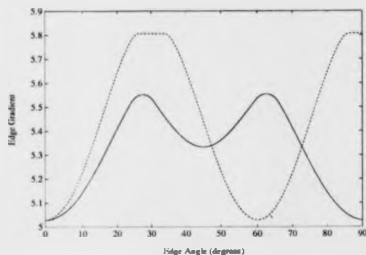


Figure 5.7: Circular Spatial Region, Diameter 1.00 Times Square Grid Horizontal Sampling Distance. Operator Edge Gradient Response, for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$. Key: — Square, - - - Hexagonal System.

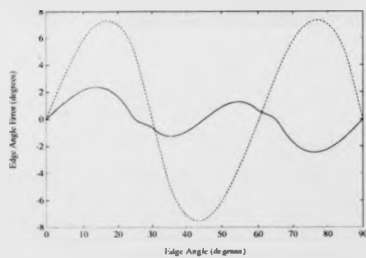


Figure 5.8: Circular Spatial Region, Diameter 1.00 Times Square Grid Horizontal Sampling Distance. Operator Angular Error for Edges Oriented $0 \Rightarrow 90^\circ$. Key: — Square, - - - Hexagonal System.

It is concluded that other features inherent in the edge detection operator design are responsible for the errors measured. Figure 5.8 shows the square and hexagonal operator edge angle errors. Generally these are approximately 25% larger than for the original band-limiting scheme, but for the square grid, large errors are exhibited for angles between 0° and 25° , and between 65° and 90° , where no errors could be detected with the original band-limiting scheme. This probably results from Davies' observation [35] that the Sobel detector is the optimal design for minimising angular errors with square spatial support regions.

As shown in Figure 5.9, when the diameter of the circular spatial area surrounding the sampling point is increased to 1.50 units, the peak magnitude errors found in the square and hexagonal systems reduce and become almost equal. It should be noted that the vertical scale on the graph has been changed. Figure 5.10 shows the corresponding angular errors. For the square case, the maximum errors are slightly larger than before, but for the hexagonal case the errors are much reduced, and are now less than those measured for the square case.

Figures 5.11 and 5.12 show the edge gradient magnitude and the angular error when the diameter is increased to 2.00 units. Now both the magnitude and the angular errors are less for the hexagonal than for the square system.

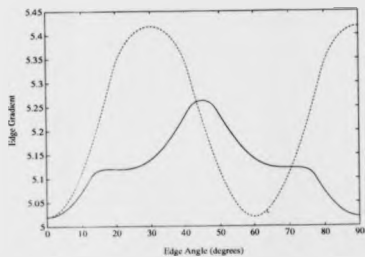


Figure 5.9: Circular Spatial Region, Diameter 1.50 Times Square Grid Horizontal Sampling Distance. Operator Edge Gradient Response, for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$. Key: — Square, --- Hexagonal System.

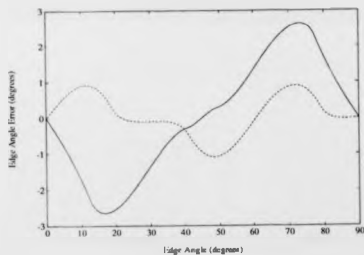


Figure 5.10: Circular Spatial Region, Diameter 1.50 Times Square Grid Horizontal Sampling Distance. Operator Angular Error for Edges Oriented $0 \Rightarrow 90^\circ$. Key: — Square, --- Hexagonal System.

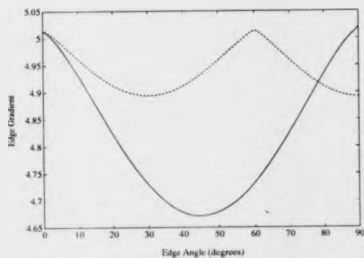


Figure 5.11: Circular Spatial Region, Diameter 2.00 Times Square Grid Horizontal Sampling Distance. Operator Edge Gradient Response, for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$. Key: — Square, - - - Hexagonal System.

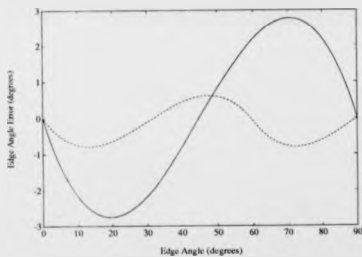


Figure 5.12: Circular Spatial Region, Diameter 2.00 Times Square Grid Horizontal Sampling Distance. Operator Angular Error for Edges Oriented $0 \Rightarrow 90^\circ$. Key: — Square, - - - Hexagonal System.

In summary, both the square and hexagonal edge detection operators were designed to produce minimum edge orientation errors for schemes where the sampling point value was calculated by averaging the brightness values in the area of the associated pixel. The operator weights were calculated exactly and then these values relaxed so that integer only arithmetic needed to be performed. This gave a processing advantage.

This technique assumes the sensor's active area to be pixel shaped, and its response to be an unbiased average of the light intensity distributed across its surface. However, for practical sensors, even if the sensor element is pixel shaped, it is unlikely to exhibit this ideal response. Some sensors such as a vacuum tube TV camera will produce good approximations to circularly band-limited signals, whereas others such as CCD arrays may not. Individual CCD elements may be square or rectangular in shape, but as noted in Section 3.3.1.2 researchers have concluded that modelling the signals as square or rectangular band-limited signals can be inaccurate. Additionally, in a practical image sampling device the signals are likely to be band-limited before sampling so that the Nyquist sampling criterion can be met. Circular band-limiting is preferable, but practical considerations, such as those inherent in CCD array design, may lead to non-optimal limiting. Hexagonal CCD grid arrays of rectangularly shaped elements have been produced for TV cameras, but for hexagonal image processing with the operators developed here, and to obtain the advantage of equal high frequency information at all orientations, a hexagonally shaped element will produce the best approximation to a circularly band-limited signal.

Different edge detection operators may perform optimally on images captured by practical sensor systems, but the hexagonal edge detector designed here has been shown to produce small errors when used on signals derived from circular spatial areas with diameters larger than the sampling point spacing. These areas produce simulated signals that are a reasonably good approximation to circularly band-limited signals. A more sophisticated model could weight contributions to the sampled value as a function of the inverse of the distance from the sampling point.

5.1.5 Edge Position Within a Pixel.

The previous results were all obtained for edges that passed through the centre of the central pixel of the local area. In this section families of edges with orientations from 0° to 90° are modelled and detected for both the square and hexagonal grid systems to see if any extra errors are introduced. This has been done for five points within the central pixel for each grid system. Points are situated on either side, and above and below the pixel's central point, and at various distances from it. Figure 5.13 shows the five curves for the measured edge gradients for the set of edges at each point in the square grid system. These are plotted together with the original central common point curve. The edge model employing sampling points with support from square spatial regions has been used. Interestingly, the curve for the edges through the original central point exhibits the maximum magnitude error of 0.4 units. The other curves diverge from this curve at various edge angles, but the error magnitudes are less. In contrast, the curves plotted in Figure 5.14 show minimum angular errors for the family of edges passing through the central point. This is as expected, as the Sobel edge detector has been designed to minimise angular errors.

Figures 5.15, 5.16, 5.17 and 5.18 show the corresponding families of curves for the hexagonal system. In Figures 5.15 and 5.16 the sampling point values have been calculated from the average over rectangularly shaped local areas by the edge image modelling program. Here the magnitude of the gradient errors can be greater for the non-central edges. In Figures 5.17 and 5.18 the sampling point values have been calculated by an averaging over a circularly shaped local area by the edge image modelling program. Now the peak magnitude of the gradient errors for the non-central edges are less than the magnitude of the peak errors for the central edges, and the peak magnitude of the angular errors for the edges at all the points are reduced.

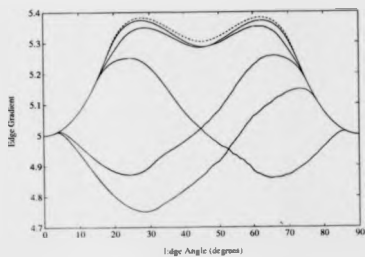


Figure 5.13: Sobel Operator Edge Gradient Responses, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Square Sampling Point Support Region. Key: - - - Central Common Edge Point, — Non Central Common Edge Point.

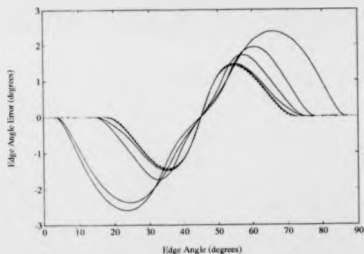


Figure 5.14: Sobel Operator Edge Orientation Errors, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Square Sampling Point Support Region. Key: - - - Central Common Edge Point, — Non Central Common Edge Point.

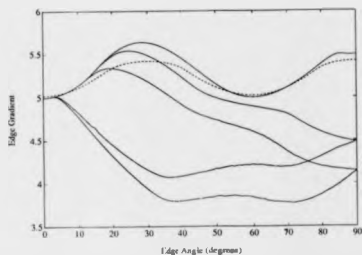


Figure 5.15: Hexagonal Operator Edge Gradient Responses, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Rectangular Sampling Point Support Region. Key: - - - Central Common Edge Point, — Non Central Common Edge Point.

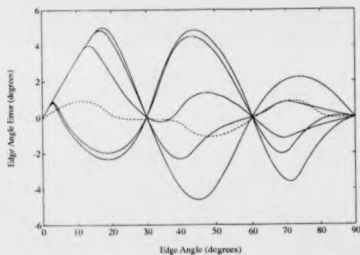


Figure 5.16: Hexagonal Operator Edge Orientation Errors, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$, Rectangular Sampling Point Support Region. Key: - - - Central Common Edge Point, — Non Central Common Edge Point.

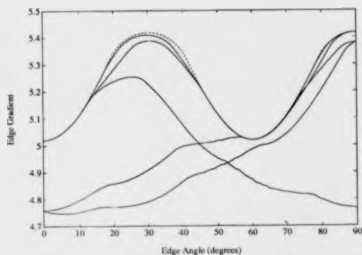


Figure 5.17: Hexagonal Operator Edge Gradient Responses, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90$. Circular Sampling Point Support Region, Diameter 1.5 units. Key: - - - Central Common Edge Point, — Non Central Common Edge Point.

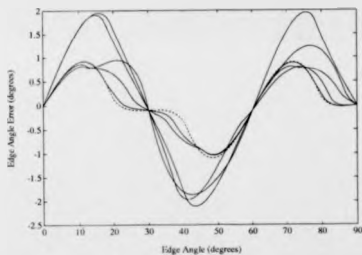


Figure 5.18: Hexagonal Operator Edge Orientation Errors, for Families of Edges, Through Various Common Points Within The Central Pixel. Gradient = 5.0, Oriented $0 \Rightarrow 90$. Circular Sampling Point Support Region, Diameter 1.5 units. Key: - - - Central Common Edge Point, — Non Central Common Edge Point.

In summary, generally, for a square grid, the edge angle errors increase for edges that do not pass through the central point of the central pixel of the local area. The magnitude of the edge gradient errors for non-central edges generally stay within the boundaries for the central edge errors. This is not so for edges modelled on a hexagonal grid where the sampling point values are calculated by averaging over a rectangular spatial support region. If a circular spatial support region is chosen, then the hexagonal system errors are comparable to those exhibited by the square system.

5.2 Conclusions.

The high symmetry of the hexagonal grid led to the design of simple, accurate, edge detection local operators, compared to those used in rectangular grid systems.

Comparable magnitude and angular accuracy can be achieved with well designed differential operators in both systems. However, whereas the square system Sobel operator produced low errors with image sampling points that were calculated by averaging square spatial support regions, the hexagonal operator produced low errors when the sampling point values were calculated by averaging circular spatial support regions. It was noted that circular band-limiting is advantageous for image processing as equal high frequency information is contained in the sampled image in all directions, and thus for example, if a small feature can be detected at one orientation, it will be detectable at all other orientations. The ease of producing circularly band-limited signals from sensors with circular active areas was noted. The simpler operator used with the hexagonal system led to faster processing, and together with a reduced number of sampling points required with the hexagonal grid, a processing time saving approaching 44% was demonstrated.

The computation processing time savings are very machine dependent, and the edge detector accuracy is dependent on how the signals are band-limited, but the case has been made that hexagonal sampling grids should be considered when machine vision systems are being designed.

Chapter 6

Case Studies: Comparison of Square and Hexagonal System Algorithms.

At an early stage in the design of the processor architecture, it was decided to identify which was the optimum sampling system, and which was a useful set of image processing operators for general industrial use. Many examples needed to be explored before these questions could be satisfactorily answered. Two different examples, one involving the detection of surface defects in sand cores used for automobile engine castings, and the other dealing with the recognition of printed characters, are presented below. The sand core example involves the grey-level processing of images with techniques such as edge detectors, while the printed character example explores the different area of binary image processing, and in particular the formation of skeletal representations of shapes.

The pipelined processor architecture, pursued in Chapter 8, can easily accommodate local operators, and so these are used where ever possible in the studies presented here. The pipeline simulation program, developed in Chapter 9, was extensively used for the image processing.

6.1 The Simulation of Hexagonally Sampled Images.

In the following examples, real world images, digitised and captured by a PIP-512 [117] frame grabber were processed. This board digitises images on a rectangular 512x512 matrix of sampling points, but with a 4:3 aspect ratio. The pixels were therefore rectangular. To make comparisons between the rectangular and hexagonal sampling systems, the resolution was reduced to 256x256 and neighbouring pixels averaged in such a way as to produce one image on a rectangular grid, and one on a hexagonal grid. Both the rectangular pixels and the hexagonal pixels were of equal size, and had a 4:3 aspect ratio. The hexagonal pixels were, however, offset by half the sampling distance on alternate lines.

It should be noted that these images were processed by operators designed to be optimum for square and regular hexagonal systems. True square and regular hexagonal images have been computer simulated where necessary to enable more exact comparisons to be made. An alternative would have been to resample the signals, or to obtain a new frame grabber. Both alternatives however, would have been expensive in terms of time or money.

6.2 Surface Defect Detection in Sand Cores used for Automobile Engine Castings.

Sand cores are used as part of metal casting processes. The core is constructed from sand and is bound together by a resin. Such a core, used in the casting of a multi-port internal combustion engine inlet manifold, is shown in Figure 6.1. Defects can be large, as with the missing "finger", or small. Figure 6.2 shows detail of the second finger pair from the right of the sand core image. Three small surface scratch defects exist in this pair and these can also be seen in the corresponding "hexagonal" image, Figure 6.3. The illumination employed divided the image of each defect into

a bright and a dark (shadow) segment. There is one large circular defect, a long thin defect, and a small defect with dimensions comparable with the pixel size. The core is 35cm long and 14cm high. The large circular defect has a diameter of 6mm, the long thin scratch has dimensions 30mm by 2mm, and the small circular defect has a diameter of 1mm.

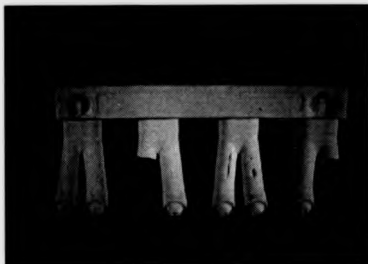


Figure 6.1: Rectangular Sampled Sand Core Image, 512x512 Resolution.

On comparing the square and hexagonal images in Figures 6.2 and 6.3, the defects can generally be seen more clearly in the hexagonal. The offsetting of pixels on alternate lines enables the eye to trace their outlines more readily at this resolution. The large circular defect appears more circular, and the light and dark segments are more easily discerned. The long thin defect is more easily discernible as a connected component. The object's edges, which in these examples are near vertical, are easier to localise in the hexagonal image. Long repeating brightness step sequences are observed in the rectangular image, whereas, a small castellated effect is observed in the hexagonal. In an attempt to segment the defects from the remainder of the image, the two images were then edge detected using the optimum Sobel and corresponding hexagonal operators introduced in Section 4.4.1.3. The threshold level was set manually so that the resulting edge images contained,



Figure 6.2: Rectangular Sampled Sand Core Image Zoomed, 64x64 Resolution.



Figure 6.3: Hexagonal Sampled Sand Core Image Zoomed, 64x64 Resolution.

where possible, connected edges around the defects, and so that the number of false detections was minimised.

Figure 6.4 shows the resulting square edge detected image, and Figure 6.5 the resulting hexagonal image. The large circular defect appears to be square in overall shape in the rectangular image and there is a small disconnection in the outline. In the hexagonal image, it appears more circular, and the structure, such as the central dividing line between the light and dark segments, is more easily discerned. There is also a small gap in the outline. The long thin defect has a break in its outline in the square image, whereas the outline is complete in the hexagonal. The equal width of this defect along its length is more discernible in the hexagonal image. The small defect's presence is indicated by a small group of edge pixels in each image. There are also more detected false edge points in the square image. These are seen as unconnected black pixels in various parts of the image.

Figure 6.6 and Figure 6.7 show the square and hexagonal edge detected images after thinning. The same points as above, concerning the defects, are still evident. The near vertical object edges appear as gradually increasing steps in the rectangular image, whereas in the hexagonal a castellated effect is visible.

Human Interpretation. Interpretation of the images depends on the individual observer, and the resolution of the image being viewed. At the low, 64x64 resolution of the above images, features are easily discerned in the hexagonal images, and their true shapes, whether circular or rectangular, can be more easily estimated. At the higher resolutions of 256x256 and 512x512, the aliasing effects at the object edges are less troubling to the eye and may be undiscernible at even higher resolutions. With the offsetting of pixels on alternate lines in the hexagonal system, the human eye may be able to estimate the boundaries between features more accurately as the pixel boundaries do not align to form long vertical features as in the square system. These vertical pixel boundaries structures may bias the eye into perceiving a near vertical feature to be perfectly vertical rather than allowing



Figure 6.4: Rectangular Sampled Sand Core Image Edge Detected, 64x64 Resolution.



Figure 6.5: Hexagonal Sampled Sand Core Image Edge Detected, 64x64 Resolution.



Figure 6.6: Rectangular Sampled Sand Core Image, Thinned, 64x64 Resolution.



Figure 6.7: Hexagonal Sampled Sand Core Image, Thinned, 64x64 Resolution.

a more realistic estimation of the feature's angle. The effects are discussed further in Section 3.4.3.

Small Defect Detection. In the above study, the small circular defect (diameter, 1mm) was the smallest significant defect. For defect detection, as opposed to measurement, the most efficient system will be one that detects a sufficient percentage of defects, while covering the objects by the smallest number of images. Once a defect has been detected, subsequent, higher resolution systems can be used for measurement. In the following sections, the problem of detecting small defects is investigated.

6.2.1 Defect Modeling.

In the section above, the rectangular and hexagonal grid images, simulated from the original 4:3 aspect ratio image, were not optimum for use with the chosen edge detectors. The results from the sand core images appeared to show some advantages for hexagonal processing, but these advantages must be shown on true square and regular hexagonally sampled images as the edge detectors and other operators have been optimally designed for these. Similar defects have been simulated, from mathematical models, on these grid systems as described below.

The Model. The model was implemented on a computer. Its aim was to produce two output images, one on a square 16x16 grid, and the other on a 13x16 hexagonal grid, that were representations of a precisely positioned and sized shape, the dimensions of which had been input at the start of the program. At present, two defect shapes have been implemented, a circle and a rectangle. These can be of arbitrary size and position in the output images to within an accuracy of 0.03 of the length of a square pixel side. These shapes were chosen because they are representative of a broad range of image features. For example they closely represent the defects found in the sand core image, a scratch being a long thin rectangle. Other shapes can be easily implemented as required.

Within the computer model the outline of the requested shape was plotted on a 512x512 square

pixel grid and then each pixel was modified by other parameters such as background and defect brightness. When completed, this defect image was averaged to produce the output images. A local area of 32×32 pixels was averaged to produce one square output pixel and an area of 37×32 pixels to produce one hexagonal output pixel. With images of this resolution, averaging of an integral number of pixels produced no distortion to the square and only very little to the hexagonal grid. This method produced a pair of almost exactly comparable images with which to test the square and hexagonal systems. The total area covered by the hexagonal image was slightly less than that covered by the square image.

A simple model will contain the specified shape with the background brightness set at one value, and the defect brightness set to another. Where a pixel straddles the background-defect border it will take an intermediate brightness value. The pixel's brightness value is determined by a simple averaging of the values of the sub-pixels within the area of the pixel. As discussed in Section 5.1.4 such a calculation will generally favour the square system edge detection process, but is more appropriate than averaging over a circular area, which favours the hexagonal process, for the comparison tests described here. A 3-D plot of such a defect is shown in Figure 6.8. A close examination of actual defects revealed the defects' edges to be spread over several pixels in the transition from defect to background brightness. This effect was recreated in the model by smoothing the defect data in the high resolution image, before the low resolution output images were produced. Real images are noisy, and so noise of a predetermined variance can be added to the model data. A 3-D plot of a defect, as in Figure 6.8, but which has had its edges smoothed and noise added, is shown in Figure 6.9. A further sophistication can be added by dividing the defect into two segments, one which is brighter than the background and one darker. This models oblique illumination from one side of the object.

Applying edge detectors to the various modelled defects revealed results in close agreement with those obtained from the sand core images in Section 6.2. However, many variables in the detection

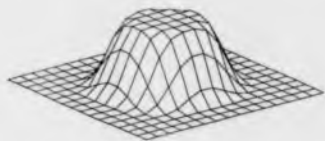


Figure 6.8: A Simple Circular Defect Model.

process were identified and the overall question remains: Will a hexagonally sampled system enable quicker and more exact detection for a large class of defects? In a noise free environment, even a sub-pixel size defect casting a shadow may have a detectable effect on the value of a pixel, but in a noisy environment false accept-reject decisions will occur. Slightly larger defects may produce effects in a group of neighbouring pixels, this group may still be unconnected, but their proximity can be used to decrease the number of incorrect accept-reject decisions. Still larger defects may be detected as bounded regions and measurements of area and shape made to aid the accept-reject decisions and provide diagnostic information for the industrial process. Theoretically the hexagonal system should show advantages with small defect detection. Firstly, as the local operators used are more compact, they should follow tightly curved edges more easily, and secondly, the connectivity definition should enable easier outlining. The definition of defect detection used is task dependent, and additionally many other variables are introduced by the processes used to achieve the detection. These variables have to be considered before a comparison can be made between defect detection on square and hexagonal grid images. For a detection scheme that involves defect outlining, variables include :- (1) The edge detector type, (2) the defect position and size, (3) the connectivity definition,

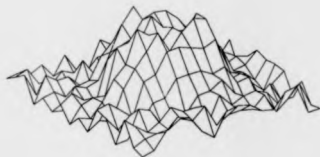


Figure 6.9: A Simple Circular Defect Model with Noise Added and Edge Smoothing.

(4) the environmental and system noise, (5) the detection criterion, and (6) the threshold method and value. These points are expanded below:-

1. **The edge detector.** The square and hexagonal detectors used in the preliminary studies are equivalent for use in both systems, but they were designed to be optimal for straight step edges. With defects, tightly curved non-step edges are likely to predominate, and this may lead to different detectors being designed. Step edges are defined by Haralick [71].
2. **The defect position and size.** If the unsmoothed defect models are used, the position and size of small defects with respect to the digitisation grid makes a large difference to the edge detector output. A defect at a particular position may line up more optimally with, say, the square grid than the hexagonal grid of sampling points. However, the signals representing real defects are band-limited, and therefore the edges are slowly changing relative to the pixel spacing.
3. **The connectivity definition.** Several definitions are available for the square system as discussed in Section 4.4.

4. **The environmental and system noise.** Noise will add uncertainty to the detection processes. System noise includes the noise introduced by analogue electronic circuit components, optical components, and quantisation noise. In some cases, image filtering stages will need to be employed.
5. **The detection criterion.** This is defined here as the percentage of complete outline necessary for detection. It may be possible to detect a defect and estimate its area and position without complete connectivity having been achieved.
6. **The threshold method and value.** The process may not be able to use optimum values.

To consider the effects of these variables, a "Defect Detection Figure of Merit" algorithm (DDFOM) could be designed. A figure of merit method has been developed by van der Heyden [181] for edge detectors, which involves applying detectors to randomly generated images of straight edges and curves. It may be possible to derive a DDFOM from the detection of a series of randomly generated defect images and to link this with computation cost figures.

6.2.1.1 Circular Defects.

Examining the large circular defect in detail, Figure 6.2 indicates the diameter to be about 6 pixels. Figure 6.10 is a 3-D plot of the area of the defect, the dark segment is easily distinguished as a slot in the centre of the plot, whereas the bright segment stands only slightly proud of the surface. Examining this plot, and the actual pixel values, a circular defect with a diameter of 6 pixels was modelled. Three models of the defect were used to generate defect images. Each subsequent model included a further sophistication that resulted in images that more closely resembled the actual defect. For the first model the circle's edges were not smoothed and no noise was added, for the second the edges were smoothed by the extent indicated by the data in the real image, and finally, in the third the edges were smoothed as in the second model and random noise was added to produce

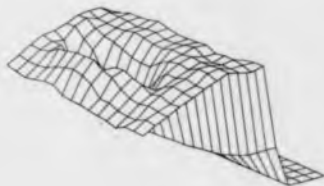


Figure 6.10: A 3D Plot of the Sand Core Large Circular Defect Image.

a modelled image that resembled the real image in most respects.

Sobel and equivalent hexagonal edge detectors were then applied to the modelled defect images, and a maximum threshold level identified so that a completely connected line of highlight pixels encircled the defect. Eight-connectedness was allowed for the square grid images. For edge detection there is an advantage in using the highest threshold value that still results in a connected edge, as the thickness of the edge, and the number of false edge points due to noise are then both minimised.

Simple Circular Defect Model. Figures 6.11 and 6.12 show the results for the set of modelled defects for which the edges were not smoothed and no noise had been added. Figure 6.11 shows the results for a circular defect with a diameter of 6.0 square sampling point spacing units. The circle's centre has been moved in 0.1 unit steps along a line parallel to the X axis through both the 16x16 square sampled image and the corresponding hexagonal image.

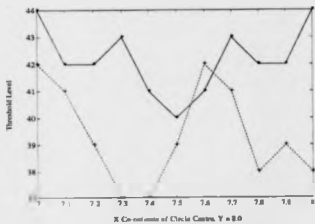


Figure 6.11: The Threshold Values for Circles, Diameter 6.0 Units, Centred at Various Positions.
Key: — Square, - - - Hexagonal System.

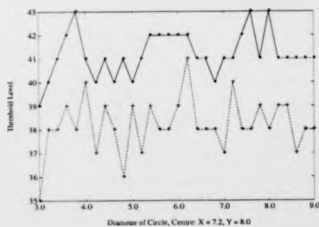


Figure 6.12: The Threshold Values for Circles of Various Diameter, Centred at 7.2,8.0.
Key: — Square, - - - Hexagonal System.

The resulting threshold values vary as the circle is moved within the image. A high threshold results if the circle locks on to the grid points within the image, as can be seen on the square system curves when the centre $X = 7.0$ and 8.0 units. The circle in the square grid image locks on to the grid more frequently than the circle in the hexagonal grid image as the circle's diameter is an exact multiple of the sampling point spacing, and the centre points that the circle can take have also been synchronised with the square grid. Even so, for one centre point the hexagonal system detection results in a higher threshold value than the square. The main conclusion here is that with these circular defects that have not been modelled with smoothed edges, the threshold value is dependent on the position of the defect with respect to the sampling grid.

In Figure 6.12 the centre of the circle has been fixed and the diameter of the circle allowed to change from 3.0 to 9.0 units in 0.1 unit steps. Again the threshold value changes as the position of the circle's edge changes with respect to the sampling grid.

Circular Defect Model with Smoothed Edges, but Without Added Noise. Figures 6.13 and 6.14 show the results for the set of modelled circles for which the edges had been smoothed, but no noise added. The resulting threshold values are lower than for the unsmoothed images as the edges are not modelled as step functions, but are more gradually rising. The spread of the threshold values for these circles of diameter 6.0 units is 3 threshold units and is reduced in comparison to the spread of 7 threshold units for the unsmoothed circles. This reduced spread results as there is a reduced locking effect between the grids and these smoothed circles.

If a suitable threshold value were to be chosen for circular feature detection (diameter 6.0 units) then the minimum values, from the above tests, of 31 for the square and 30 for the hexagonal grid system would be appropriate. It is concluded that for these modelled, smoothed circular features the two systems perform equally.

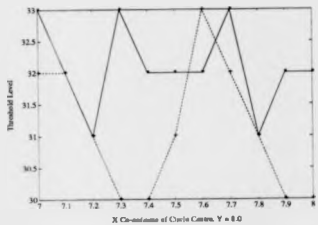


Figure 6.13: The Threshold Values for Smoothed Circles, Diameter 6.0 Units, Centred at Various Positions. Key: — Square, - - - Hexagonal System.

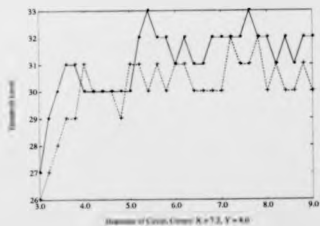


Figure 6.14: The Threshold Values for Smoothed Circles of Various Diameter, Centred at 7.2, 8.0. Key: — Square, - - - Hexagonal System.

Circular Defect Model with Smoothed Edges and Added Noise. Table 6.1 shows the results for the third set of modelled circular features. In this set the edges have been smoothed and random noise has been added to the image data. The centre of the circles have been randomly positioned in the initial high resolution image so that they will lie between 7 and 8 square system spatial units (the pixel width) from the left hand and top edges of the resulting 16x16 square system image. The diameters of the circles have been randomly chosen to lie within the range 4 to 8 units. Equivalent square and hexagonal images have then been calculated and random noise with zero mean, a uniform probability distribution, and a variance of 400, superimposed on the circle images. By inspection, addition of noise at this level produced simulated defects that closely resembled the circular defects in the original sand core images.

	Minimum	Maximum	Mean
Square System	17	34	25.9
Hexagonal System	17	39	25.1

Table 6.1: Threshold Value Statistics for 50 Random Circular Defects.

Fifty such random circular defects were produced in each system, and the threshold value that resulted in a completely connected outline for each circle recorded. The mean, maximum, and minimum thresholds for the two groups of circles are presented in Table 6.1. The mean threshold value was 25.9 for the square and 25.1 for the hexagonal system. For processing systems required to produce a completely connected outline for the majority of defects the minimum threshold value from the set of tests should be chosen. This was a value of 17 for each system. It is concluded that for these modelled, smoothed, noisy circular defects the two systems perform equally.

6.2.1.2 Rectangular Defects.

Several rectangular defects of various size and orientation were edge detected and similar comparisons as above made between the two systems. Three models of each were produced. For the first set of models the edges were not smoothed and noise was not added. For the second set the edges were smoothed by the extent indicated by the data in the real sand core image. Finally, in the third set the edges were smoothed as in the second set and random noise was added to produce modelled images that resembled the real image in most respects. Sobel and equivalent hexagonal edge detectors were then applied to the modelled defect images, and a maximum threshold level identified so that a completely connected line of highlight pixels encircled the defect. Eight-connectedness was allowed for the square grid images, i.e. diagonal elements were included.

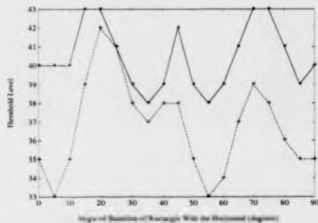


Figure 6.15: The Threshold Values for 5x5 Unsmoothed Rectangles with no Added Noise, Centred at 8.0, 8.0 on the Rectangular and a Corresponding Point on the Hexagonal Grid, and with Their Base Lines at Angles Between 0 and 90 Degrees to the Horizontal. Key: — Square, - - - Hexagonal System.

Simple Rectangular Defect Model. Figure 6.15 shows the results for a set of unsmoothed rectangular defects with no added noise. The rectangles were each of size 5x5 square system pixel units and centred at 8.0, 8.0 on the 16x16 square system grid. The base line of each successive

rectangle was oriented in 5° steps from 0° to 90° . The rectangles in the hexagonal grid system were each equivalent to one of the square system rectangles in size and position. In the square grid system the geometric centre of the rectangles are located at one of the grid points and the graph is symmetric about a 45° base line angle. This is with the exception of a one threshold unit difference between the values at 5° and 85° that probably results from a rounding error in either the modelling or detection program. In contrast the rectangles on the hexagonal grid are not centred at a sampling point and no symmetry about any point is observed in the graph. The threshold values for the rectangles in the hexagonal system are generally lower than those in the square system. This results from the ease with which the square system rectangles, with their dimensions equal to an integral number of square system pixels and their grid locked centres, lock on more readily to the grid points throughout the square system image. However by chance, the hexagonal system rectangle at 20° does lock well with the hexagonal grid and results in a threshold value of 42 units. It is concluded that for these basic unsmoothed models the locking of the feature with the grid is a large factor in the value obtained by the thresholding process.

Rectangular Defect Model with Smoothed Edges, but without Added Noise. Figure 6.16 shows the results for a similar set of rectangles to the above, but that have been smoothed. Noise has not been added to these modelled images. The degree of smoothing was as indicated by the smoothing found in the real sand core defects. As for the corresponding set of circular defect models the resulting threshold values were lower than for the unsmoothed images as the edges are no longer modelled as step functions, but are more gradually rising. The spread of the threshold values is more narrow than for the unsmoothed rectangles although some peak values due to the locking of the feature with the grid can still be observed. The mean value for the square grid results is 30.5 threshold units, and the mean value for the hexagonal grid 29.0 units. This small difference in the threshold values is attributed to the grid locking effect which has not been fully filtered out.

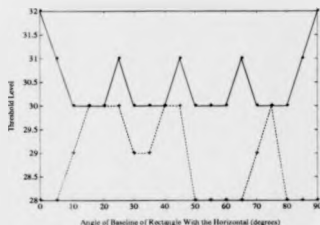


Figure 6.16: The Threshold Values for 5x5 Smoothed Rectangles with no Added Noise, Centred at 8.0, 8.0 on the Rectangular and a Corresponding Point on the Hexagonal Grid, and with Their Base Lines at Angles Between 0 and 90 Degrees to the Horizontal. Key: — Square, --- Hexagonal System.

Rectangular Defect Model with Smoothed Edges and Added Noise. Table 6.2 shows the results for the third set of modelled rectangular features. In this set the edges have been smoothed and random noise has been added to the image data. The centres of the rectangles have been randomly positioned in the initial high resolution image so that they will lie between 7 and 8 square system spatial units (the pixel width) from the left hand and top edges of the resulting 16x16 square system image. The length of the major and minor axes have been randomly chosen to lie within the range 4 to 6 units, and the angle that the base line makes with the horizontal has been randomly chosen between 0° and 90°. Equivalent square and hexagonal images have then been calculated and random noise with zero mean, a uniform probability distribution, and a variance of 400, superimposed on the images of the rectangles. By inspection, addition of noise at this level produced simulated defects that closely resembled the rectangular defects in the original sand core images.

Fifty such random rectangular defects were produced in each system, and the threshold value that resulted in a completely connected outline for each rectangle recorded. The mean, maximum, and minimum thresholds for the two groups of rectangles are presented in Table 6.2. The mean threshold

	Minimum	Maximum	Mean
Square System	16	38	25.4
Hexagonal System	16	35	25.7

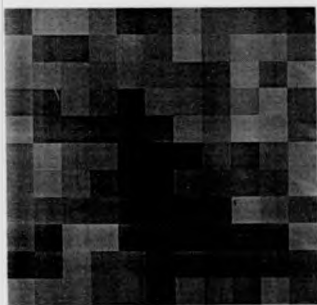
Table 6.2: Threshold Value Statistics for 50 Random Rectangular Defects.

value was 25.4 for the square and 25.7 for the hexagonal system. For processing systems required to produce a completely connected outline for the majority of defects the minimum threshold value from the set of tests should be chosen. This was a value of 16 for each system. The random choice of each rectangle's size and position has removed the grid locking effects that were observed in the previous sets of data. It is concluded that for these modelled, smoothed, noisy rectangular defects the two systems perform equally.

6.2.1.3 Long Thin Defects.

Long thin defects have been modelled as rectangles with high major to minor axis ratios. The long thin defect in the original sand core image (Figure 6.2) has been modelled as a rectangle with a major axis of length 8 and a minor axis of length 2 square pixel units. The major axis was inclined at 60° to the horizontal. Initially the background pixel values were set to 120 and the defect pixel values to 30 brightness units (0 to 255 unit system range). These values were then smoothed by a filter in the high resolution image before a pair of equivalent low resolution square and hexagonal images were formed. Random noise with zero mean and variance 400 was then superimposed on each low resolution image. Figures 6.17 and 6.18 show a pair of square and hexagonally sampled images that were produced by the rectangle modelling program from this set of starting conditions. The edge detected images are also shown.

Twenty five such defects were modelled, each with a randomly chosen geometric centre with co-ordinates in the range 7.0 to 8.0 square system spatial units (the pixel width) from the left hand

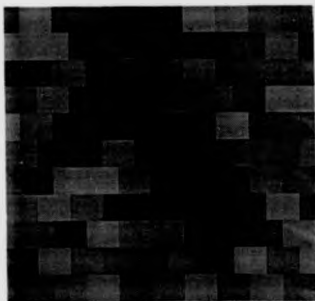


(a)



(b)

Figure 6.17: Modelled Rectangular Defect, Square Sampled. Size: 8×2 Pixels, Major Axis at 60 Degrees to the Horizontal, and Noise Added with a Variance of 400. (a) Grey-level Image. (b) Edge Detected Image.



(a)



(b)

Figure 6.18: Modelled Rectangular Defect, Hexagonal Sampled. Size: Equal to the Corresponding Square Sampled Defect, and Noise Added with a Variance of 400. (a) Grey-level Image. (b) Edge Detected Image.

	Minimum	Maximum	Mean
Square System	13	25	18.0
Hexagonal System	16	27	20.4

Table 6.3: Threshold Value Statistics for 25 Random Long Thin Defects.

and top edges of the resulting 16x16 square system image. A different set of random noise values were added to each modelled image. The modelled images were found to resemble numerically and visually the original sand core defect image. Table 6.3 presents the mean, maximum and minimum threshold values for the two groups of defects. The threshold value for each image was chosen so that a completely connected defect outline was formed. Care was taken to ensure the outline was not shortened in the direction of the major axis by the choice of too high a threshold value.

The mean threshold value was 18.0 for the square and 20.4 for the hexagonal system. For processing systems required to produce a completely connected outline for the majority of defects the minimum threshold value from the set of tests should be chosen. This was a value of 13 for the square and 16 for the hexagonal system. For this particular defect which models an actual defect in the sand core, a quality advantage exists for processing in the hexagonal system as a higher threshold value can be set. A higher threshold value will result in fewer noise induced edge pixels being detected.

6.2.2 Alternative Methods.

In some applications, it may be possible simply to threshold images to identify defects. This was not possible in the sand core example as the surfaces are curved, resulting in large illumination gradients across the image.

In this example, the defect was considered detected if a group of connected edge pixels were found. This procedure could be formalised by employing distance functions as described in Section 4.4.1.6, or a Hough transform method, as described in Section 4.5.1. A Hough space could



Figure 6.19: Isolation of Object Edges using a Hough Transform Technique.

be chosen that would accept data from either the square or hexagonal grid, however, the Hough method will only respond to a known shape, and will require a large defect perimeter to be effective.

A Hough transform method was used to produce the image shown in Figure 6.19, in which the straight line boundary pixels have been identified by darker lines than the other edges. If these object boundary regions were now relaxed a little, the defect and noise pixels could be isolated. An alternative defect isolation technique would be to correlate the image outline with a perfect model outline.

6.2.3 Sand Core Study Conclusions.

Although a subjective judgement, the visual appearance as judged by a human observer of the hexagonally sampled sand core images made the task of defining object and defect boundaries easier in the grey level images. Once edge detected, the outline of the defects gave a more geometrically correct interpretation of the perimeter in the hexagonal system. This can be important for defect measurement applications.

Detection reliability was investigated further by modelling the same defects on both the true

square and hexagonal grids. It was initially thought that the hexagonal operators would produce more robust small defect outlines as the operators are more compact and therefore capable of following tight curves more easily. In practice the extra sampling points in the square system cancelled the advantage out, and equal detection reliability was observed.

Simple models of features that have edges modelled as step functions were found to lock on to the sampling grid when they were of a certain size and position. For these simple models the effects on the choice of edge detection threshold value of these grid locking effects was found to be greater than the effects attributable to the different square and hexagonal edge detection processes used.

A more sophisticated defect model was developed in which the step edges in the original model were smoothed. The degree of smoothing applied to the edge was variable and in practice chosen to give a brightness gradient across the edge equivalent to that found in a real defect image. The effects on the threshold value attributable to the locking of the image features with the sampling point grid were greatly reduced with these models, and it was possible to conclude that the threshold values required for a range of equivalent defects in the square and hexagonal systems were almost equal. The threshold value was considered as a measure of quality with which to compare the square and hexagonal edge detection processes. To produce a high quality edge map image the threshold value must be chosen high enough so that thin edge lines result and so that the number of edge points attributable to image noise is reduced, while being low enough so that a continuous edge line results around a feature. Using the threshold value as a measure of quality the processes operating in square and hexagonal systems have been shown to produce equivalent results.

A third defect model was developed in which the feature's edges were smoothed as before, and then random noise was added to each image point. The threshold levels required were lower than for those in the previous model, but again were found to be almost equal for the square and hexagonal processing systems.

Finally the long thin defect in the original sand core image was modelled. The threshold value

was adjusted for each test image until a completely connected edge outline covering the entire area of the defect resulted. The hexagonal system produced better results for this particular defect.

The production of a general "defect detection figure of merit" for a particular sampling system or edge detection process was found to be difficult to achieve as each problem exhibited a different set of defects, lighting conditions and surface reflectance properties etc. However the technique of examining a small defect and then modelling it had some practical use as the modelled defect can then be scaled, moved, or oriented differently with respect to the sampling grid, and a more confident figure for detection reliability obtained for any particular edge detection operator.

6.3 Printed Character Recognition.

This topic has been reviewed in Section 4.4.1.4. Here a comparison is made between similar thinning algorithms applied to equivalent data in square and hexagonal grid systems. 2-D images of sections of printed text were captured by a grey-level frame grabber, averaged to produce rectangular and hexagonal images, and then thresholded to produce binary images such as those shown in Figure 6.20 and Figure 6.21. The threshold value was set at a level indicated by the valley between the dark and bright peaks of a histogram of the pixel values of each image. Such a bimodal histogram is likely to result from black printing on white paper. The thinning operators used below operate equally on 4:3 aspect ratio images, and square or regular hexagonally sampled images.

The Rectangular Thinner. An iterative thinner, as developed by Chin et al [27], was used on the rectangular image. This algorithm was chosen as the operators were always applied to the image matrix in the same pixel order. Usually this means starting at the top left corner and scanning along each image line, left to right, until the last line was reached. This arrangement suited the pipeline architecture presented in the following chapters of this thesis, in that the operators were local, and no intermediate static frame buffering was required.



PROG

Figure 6.20: Thresholded Rectangular System Text.



PROG

Figure 6.21: Thresholded Hexagonal System Text.

The algorithm involves the application of a set of 8 thinning templates, and 2 restoring templates, as shown in Figure 6.22. The thinning templates switch the centre pixel to 0 if all the 0, 1 conditions in that template are true. The restoring templates ensure that the skeleton is not broken. The templates are applied in parallel to each image pixel in turn, and repeatedly to the image until no further changes occur. The final result is the skeleton. Figure 6.23 shows the result of applying this algorithm to the text in Figure 6.20. The characters are thinned, but several "spurs" are evident, especially on the apex of the P and on the G. The spurs are caused, in most cases, by single dark pixels protruding from the edge of the character. This can be verified by referring back to Figure 6.20. However, the lower right spur on the G is produced by the foot of the letter. All but the last mentioned spur could be removed by a morphological filter.

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ x & 1 & x \end{bmatrix} \begin{bmatrix} 0 & 1 & x \\ 0 & 1 & 1 \\ 0 & 1 & x \end{bmatrix} \begin{bmatrix} x & 1 & x \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x & 1 & 0 \\ 1 & 1 & 0 \\ x & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} x & 0 & 0 \\ 1 & 1 & 0 \\ x & 1 & x \end{bmatrix} \begin{bmatrix} 0 & 0 & x \\ 0 & 1 & 1 \\ x & 1 & x \end{bmatrix} \begin{bmatrix} x & 1 & x \\ 0 & 1 & 1 \\ 0 & 0 & x \end{bmatrix} \begin{bmatrix} x & 1 & x \\ 1 & 1 & 0 \\ x & 0 & 0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} x & x & x & x \\ 0 & 1 & 1 & 0 \\ x & x & x & x \end{bmatrix} \begin{bmatrix} x & 0 & x \\ x & 1 & x \\ x & 1 & x \\ x & 0 & x \end{bmatrix}$$

(b)

Figure 6.22: (a) Chin's Thinning Templates. (b) Chin's Restoring Templates. $x = \text{don't care}$.



Figure 6.23: Rectangular System Text Thinned Without Morphological Smoothing.

The Morphological Edge Smoothing Filter. The edge pixels that caused the unwanted spurs in the thinned character images can be removed by morphologically opening the image with a square 3×3 structuring element. Morphologic openings are discussed in Section 4.4.1.5. The structuring element appears as in Figure 6.24, but note that the image was inverted before this 1's element was applied. An alternative would have been to close the non-inverted image with this structuring element.

This opening removed one and two pixel wide spikes as shown in Figure 6.25. When the thinning routine was then applied, the resulting image was spur free, as shown in Figure 6.26. The



Figure 6.24: The Square 3x3 Structuring Element.

hexagonal image can be smoothed by a hexagonal structuring element as shown in Figure 6.27, but this was found to be unnecessary.

PROG

Figure 6.25: Rectangular System Text, Morphologically Smoothed.

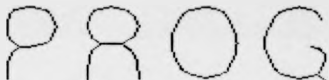


Figure 6.26: Rectangular System Text, Morphologically Smoothed and Thinned.

The Hexagonal Thinner. A similar iterative thinner to Chin's was developed for use on the hexagonal image. Here 6 thinning templates and 3 restoring templates are required as shown in Figure 6.28. As with the square thinner, the thinning templates switch the centre pixel to "0" if all the "0" and "1" conditions in that template are true, and the restoring templates ensure that the skeleton is not broken. The thinning templates test for pixels on the edges of features. Every pass



Figure 6.27: The Hexagonal Structuring Element.

of the thinning templates through the image removes a layer of value "1" pixels from each edge of the feature.

The templates presented in Figure 6.28 are logical operation templates. With some processors it may be advantageous to replace them with convolution templates, where, logic 1 becomes +1, logic 0 becomes -1, and logic x becomes 0. With the convolution set of templates, $T_1 = -T_3$, $T_2 = -T_4$, $T_3 = -T_6$, and so only the thinning templates $\{T_1, T_2, T_3\}$ need to be applied to each image point, if the magnitude of the template operator is compared with the threshold value of 3. However, as described in Section 3.4.2 on the addressing of memory in the rectangular and hexagonal systems, if orthogonal axes are used to address pixels, then two sets of templates will need to be stored for use on alternate lines. As observed in Figure 6.29, these templates are able to thin the text correctly without the need for pre-filtering. The spur, caused by the foot of the "G" in the rectangular image, does not occur. This could be advantageous, as the feature is probably type-face dependent. Chin et al [27] observe that the skeleton produced by their algorithm departs from the ideal skeleton at sharp corners, in that the skeleton is biased towards the inside of the corner. This effect is less pronounced with the hexagonal algorithm, as can be seen in the "G" of the comparable images.

Restoring template T_7 prevents the erasure of vertical and near vertical lines. It is the simplest of the restoring templates and is applied to the image pixel under test so that the pixel locates within the left-most "1" of the template. If the pattern 0110 exists the leftmost "1" will be preserved and the thinning process will move on to the right-most "1" which T_7 will not be able to preserve as the pattern will now be 110 ψ where ψ is either "0" or "1". In this way single pixel thick vertical skeletal


$$\begin{array}{c}
 \begin{bmatrix} 1 & x & \\ 1 & 1 & 0 \\ x & 0 & \\ & & T_1 \end{bmatrix} \begin{bmatrix} x & 0 & \\ 1 & 1 & 0 \\ & 1 & x \\ & & T_2 \end{bmatrix} \begin{bmatrix} 0 & x & \\ 0 & 1 & 1 \\ x & 1 & \\ & & T_3 \end{bmatrix} \\
 \\
 \begin{bmatrix} x & 1 & \\ 0 & 1 & 1 \\ & 0 & x \\ & & T_4 \end{bmatrix} \begin{bmatrix} 0 & 0 & \\ x & 1 & x \\ & 1 & 1 \\ & & T_5 \end{bmatrix} \begin{bmatrix} 1 & 1 & \\ x & 1 & x \\ 0 & 0 & \\ & & T_6 \end{bmatrix} \\
 \\
 \begin{bmatrix} x & x & x \\ 0 & 1 & 1 & 0 \\ x & x & x \\ & & T_7 \end{bmatrix} \begin{bmatrix} x & x & x \\ 0 & x & \\ x & 1 & 1 \\ 1 & 1 & \\ x & x & 0 \\ & & T_8 \end{bmatrix} \begin{bmatrix} x & x & 0 \\ 1 & 1 & \\ x & 1 & 1 \\ 0 & x & \\ x & x & x \\ & & T_9 \end{bmatrix}
 \end{array}$$


Figure 6.28: Hexagonal Thinning and Restoring Templates. $x = \text{don't care}$.

PROG

Figure 6.29: Hexagonal System Text Thinned Without Morphological Smoothing.

features are preserved.

T_8 and T_9 are more complicated than T_7 . They operate at $+60^\circ$ and -60° to the horizontal as shown in the vector diagram inserted in Figure 6.28. The 0110 decision as with T_7 still exists at the centre of the templates but the extra constraints of the two "1s" on either side of the 0110 structure have been added to limit the effectiveness of these restoring templates. In T_8 and T_9 the central element is placed over the pixel about which the decision is to be made. A pixel value of "1" is retained if all the "0,1" conditions are met. The T_8 and T_9 values were chosen after some experimentation. If the simple 0110 decisions had been retained, double pixel thickness lines can occur at some positions within the skeleton as shown in Figure 6.30. The restoring templates shown in Figure 6.28 were found to be the simplest that ensured connected, single pixel width skeletons. It was not necessary to increase the complexity of T_7 .



Figure 6.30: Hexagonal System Text Thinned Without Morphological Smoothing, but Utilising Simplified Restoring templates.

A Comparison of the Hexagonal and Square System Text Thinning Techniques. High resolution images of printed text were captured and then processed to produce equivalent 256×256 resolution images of square and hexagonally sampled text. Some of these texts contained the complete alphabet for a particular typeface, and others samples of English words. The smallest characters contained strokes with an average width of two pixels, whereas the largest contained strokes that were up to thirty pixels thick.

	No. broken before processing	No. broken by processing	No. incorrectly thinned	No. obviously better than in the other system	No. of spikes	No. correctly thinned
Hexagonal	4	0	1	5	0	149
Square	3	0	7	1	3	143

Table 6.4: A Comparison of 150 Small Characters.

Table 6.4 provides statistics from a comparison of 150 of the smallest characters thinned. Several characters contained breaks attributable to the digitisation process. It is unlikely that characters smaller than these would need to be processed by a practical text reading system. The number processed correctly was high in both cases. For the hexagonal system the letter "t" was incorrectly processed to a skeleton that resembled an "l", while in the square system, three capital letters contained unwanted spikes that could not be removed by morphological smoothing in a preprocessing stage as this process removes lines with a pixel width of less than three pixels. Additionally, in the square system, four lower case letters contained errors that could not be attributable to any cause. It was concluded that for both systems thinning can be accomplished correctly for the majority of such small characters, but that for the square system the unwanted spikes resulting from single pixel deviations in the character perimeters cannot be removed by the previously discussed morphological smoothing technique. These spikes can occur randomly in any letter and introduce additional junctions in the skeleton that will inhibit the correct operation of character recognition processes that are based on the analysis of the relative position and type of such junctions.

Table 6.5 provides statistics for a comparison of 150 large characters. These contained strokes with widths of at least four pixels. With these characters no gaps occurred within the digitised characters and no errors were detected in any of the digitised skeletons apart from the square system

	No. broken before processing	No. broken by processing	No. incorrectly thinned	No. obviously better than in the other system	No. of spikes	No. correctly thinned
Hexagonal	0	0	0	0	0	150
Square	0	0	0	0	204	150

Table 6.5: A Comparison of 150 Large Characters.

"spikes" which occur more frequently in large characters. After these spikes had been removed by a morphological smoothing filter the square system characters were all processed to their correct skeletons as noted in the rightmost column of Table 6.5. It was concluded that for both systems thinning can be accomplished correctly for characters, but that for the square system a morphological preprocessing stage must be included.

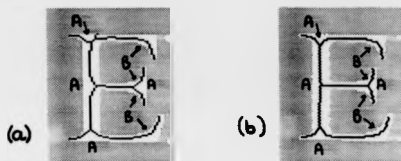


Figure 6.31: The Letter "E" with Superimposed Skeletons. (a) Square Sampled. (b) Hexagonally Sampled.

In a further test the quality of the skeletons resulting from the square and hexagonal processing were evaluated. The tests were performed on images of a Times Roman character set of 62 upper case, lower case and numerical characters. The character set contained features such as vertical, horizontal and angled strokes, tails, and a variety of curves that can be found in most character sets. Figure 6.31 shows an unprocessed letter "E" with a processed skeleton superimposed on it for both systems. The resulting skeletal features displayed in this figure are typical of those found in

the other characters in the set. The points of comparison included: (a) The position of the major skeletal lines with respect to the medial axis of the strokes in which they lay. (b) The correct position of skeletal junctions. (c) The biasing of the skeleton towards the inside edge of curves.

(a) The skeletal lines processed from the long strokes that go to make up the individual characters were checked for correctness of position with respect to the observable medial axis of each stroke. These skeletal lines were found to be positioned to within one pixel of the medial axis in both systems for all but one stroke in the hexagonal and four strokes in the square system. Table 6.6 details the results. These deviations from the perfect skeleton were few in number and not considered to be large enough to affect the subsequent character recognition processes in either system.

System	No. of characters	No. of strokes	No. better than in other system
Square	62	147	1
Hexagonal	62	147	3

Table 6.6: Analysis of the Skeletal Lines of Character Strokes.

(b) The positions of the skeletal junctions, which are formed where the skeletal lines relating to the character strokes meet, were evaluated for each grid system. Such junctions are marked "A" in Figure 6.31. The type and relative positions of junctions form the basis of many character recognition systems. The junction points were found to be placed closer to the geometric centre of the section of the original character in their locality with the hexagonal system.

System	No. of characters	No. of junctions	No. better than in other system
Square	62	150	1
Hexagonal	62	150	78

Table 6.7: Analysis of Skeletal Junctions.

Table 6.7 details the results. More than 50% of the junctions were more optimally placed in the hexagonal system than their counterparts in the square system. Only one junction was found to be

more optimally placed in the square system than its counterpart in the hexagonal system.

(c) As discussed previously in this section, the square system thinner was found by Chin [27] to produce skeletons that are biased towards the inside of tight curves and corners. Such points have been marked "B" in Figure 6.31. A similar biasing was evident in the skeletons produced by the hexagonal system, but was less pronounced than in the square system. Table 6.8 details the results. More than 50% of the skeletons were less biased towards tight curves in the hexagonal system characters than their counterparts in the square system. Only one instance where the square system performed more optimally could be found.

System	No. of characters	No. of tight curves	No. better than in other system
Square	62	136	1
Hexagonal	62	136	73

Table 6.8: Analysis of Tight Skeletal Curves.

These results have shown that the hexagonal character thinner produces skeletons which maintain a better representation of the elements of the character's shape than the square system thinner.

Implications for the Processor Architecture. Chin et al [27] suggest a pipeline processor element (PE) to implement the 10 templates in their scheme. The pipeline comprises a PE for each iterative step, and so sufficient PEs must be provided for a given thickness of line to be thinned. Pipeline processors are discussed in Chapter 8. Chin's PE contains 3 video line delays to assemble the 4x4 local area required by the templates. The templates are applied, in parallel, by a number of combinational logic gates, and the new value for the central pixel of the local area is put onto the output stream. The hexagonal scheme presented here requires only 9 templates, but 4 video line delays are required to assemble the 5x5 local area. Some address switching will also be required between alternate video lines.

6.3.1 Printed Character Study Conclusions.

The hexagonal thinning algorithm was found to maintain the essence of the characters' shapes better than the rectangular algorithm, with less bias of the skeleton to the inside of sharp corners, and a more exact positioning of the skeletal junctions. The hexagonal grid results in 13.4% fewer pixels and no morphological pre-filtering was required. The morphological "opening" pre-filter, used with the square system to remove unwanted spikes, can cause breaks in the character if the number of pixels in a line element is less than three. This can happen if the local area enclosing the character is allowed to become too small. Hexagonally processed characters do not exhibit these spikes and so the pre-filter is not required and these small characters can be successfully processed without breaks being introduced in their skeletons. There are 9 templates to be tested in the hexagonal system as opposed to 12 for the square system. Overall, fewer bits need to be tested for the hexagonal images, but an extra video line delay is required. Most commercial document readers incorporate a line scan camera as the image sensor. A dual-line scan camera, or an electro-optic deflector would need to be developed if hexagonal character processing were to be incorporated in such a reader.

6.4 Final Conclusions.

Detailed conclusions for the sand core case study are included in Section 6.2.3, and for the printed character processing case study in Section 6.3.1. Several of the hexagonally sampled images were found to be easier to interpret by a human observer than their square system counterparts.

For many of the square and hexagonal algorithm comparisons reported in this chapter the two systems produced equivalently accurate results. The hexagonal system processed long thin defects and printed character skeletons to a higher quality, but the main advantage of the hexagonal system is that a comparable processed image quality can be achieved with 13.4% fewer sampling points per unit image area, and with simpler processing templates.

Chapter 7

Computer Architectures for Machine Vision.

7.1 Introduction.

This chapter contains a survey of the computer architectures that are commonly used for computer vision. The advantages of pipeline architectures that perform low level local operations in controlled lighting conditions are given. Such conditions often exist in an industrial environment.

Processors are classified, and the advantages of each type listed. Two and multidimensional arrays are explored further with an explanation of fine and coarse grain processor arrays. Fine grain arrays and processors within a pyramid are sometimes interconnected hexagonally to their nearest neighbours, if hexagonal operations are to be performed. Several practical machines are investigated.

One dimensional vector arrays and pipeline processors are discussed and examples of their use in image processing given. Pipeline processors are subdivided into real-time processors that can operate at the data rate of the sensors, and recirculating systems that store partial results in image frame stores and process the image by passing the data several times through the same short pipeline.

7.2 The Parallel Processing of Images.

Images can be one or multidimensional, but the sensing process when an image is quantised, is often two dimensional or less. The multidimensional image data is built up over a period of time, and often stored in a memory device of equal or lower dimensional organisation.

The majority of computers at the present time employ processors that can compute a single instruction on a single data at any one period of time. Image processing on this type of architecture involves organising a 1-D stream of data and instructions from memory, through the processor, and back to memory again in such a way that efficient processing results. The image processing rate is limited by the switching speed of the devices contained within the computer processor and the memory.

Most image processes require several computer instructions to complete for each image pixel, and may need support from several additional pixels. An example of such processes are the local convolution filters presented in Section 4.4.1.2. Equation 4.2, reproduced here, is used to calculate the value of the output pixel P_S .

$$P_S = a i_{1,p} + b \sum_{p=1}^4 i_{2,p} + c \sum_{p=1}^4 i_{3,p} + d \sum_{p=1}^4 i_{4,p} + e \sum_{p=1}^4 i_{5,p} + f \sum_{p=1}^4 i_{6,p} \quad (7.1)$$

Where $a, b, c, d, e,$ and f are filter weights associated with the six shells, $i_{m,p}$ are the local area pixel values, and p is an integer number from 1 to 4. Six multiplications and twenty five additions are required for the computation of each output pixel. If we consider a computational instruction - data space, the dimensionality of this space will be a function of the dimensionality of the data, any local area specified within the data, and the complexity of the calculation.

Parallel processing involves the use of two or more processors operating within this space. The image processing rate is now a function of the number of processors. This function is not linear as communication overheads between a processor and memory, or another processor, increase with

the number of processors. Parallel processing can offer a faster processing of images, or a more cost effective solution at the same rate if a less expensive processor fabrication technology is used.

7.3 The Classification of Processors.

Flynn [54] classified processors into four groups:-

- Single instruction stream – single data stream (SISD).
- Multiple instruction stream – single data stream (MISD).
- Single instruction stream – multiple data stream (SIMD).
- Multiple instruction stream – multiple data stream (MIMD).

The term stream refers to the flow of data or instructions through the processor during program execution.

SISD. This type of processor is used in the classical architecture in wide use today. The processor communicates with the memory and input - output devices via data and address busses. Instructions and data are sequenced into a single stream through the processor by the computer program. The program is relatively easy to write as all events are separated in time.

MISD. Here there is a single stream of data from the memory, but the stream passes through several processor elements (PEs), each of which performs a different operation on the data before the data is sent to its destination. This arrangement of PEs can be thought of as a pipeline, data enters one end of the pipe, is processed by each pipe segment as it flows through, and then emerges processed, at the far end of the pipe. This is shown diagrammatically in Figure 7.1. At a particular time PE3 is operating on data0, PE2 is operating on data1, and PE1 is operating on data2.

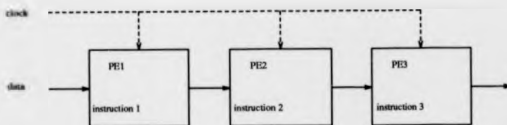


Figure 7.1: A MISD Pipeline.

The pipeline is a group of PEs operating in the data – instruction space, and so can be termed a parallel processor. Programming flexibility can be compromised by such an arrangement. Events are separated in time, as with the SISD processor, but the sequence of instructions performed in a single pipe does not allow branching or easy rescheduling of instruction order. Efficient computation is achieved by applications where the same set of instructions need to be applied to large sets of data.

SIMD. These processors again contain several PEs, but here each PE has direct access to at least part of the memory. In some cases the PE may incorporate a memory element or group of elements. Each PE in the processor will be performing the same instruction at the same time. The PEs are processing the data in parallel.

Being so closely associated with memory the PEs tend to be formed into arrays of the same dimension as the memory organisation. The dimensionality is no longer simply a matter of the way the memory is addressed, but also indicates which PEs are near neighbours and may need to communicate with each other. In an alternative architecture the PEs are formed into a lower dimensional array through which the data is streamed in parallel. In both cases each PE in the array is a SISD processor, but in many applications the arrays tend to be large and the PEs simple so that many can be integrated onto a single VLSI device. The next instruction for all the PEs to operate on is broadcast to each PE simultaneously from a system controller. The controller is programmed so that the PE array will process the data. Efficient computation is achieved by applications requiring

the same instruction to be applied to every data point simultaneously.

MIMD. As with SIMD processors each PE has direct access to at least part of the memory, but each PE can be executing a different instruction at any one time. There is a tendency for the PE to be more complex than the SIMD PE, and for it to access a larger local area of memory. A wide variety of applications can be run on such a processor. The programmer must divide the task between PEs so that the process will run efficiently.

Bandwidth and Latency. Flynn [54] defines computational or execution bandwidth as the number of instructions processed per second, and the storage bandwidth as the retrieval rate of data from memory. He defines latency as the total time associated with a process from excitation to response for a particular data. In practical terms, the latency is the number of processor clock cycles that elapse between the input of a datum and the output of the processed result.

Processor Architectures. These characterisations and definitions are used in the following descriptions of image processing architectures. The architectures are grouped according to the dimensionality of the arrays in which the PEs reside.

7.4 Two and Multi-Dimensional Processor Arrays.

7.4.1 Introduction.

An array of PEs is a group of elements that operate in parallel to process a set of data. Consider initially a simple image intensity transform where each member of a data set $b_{i,j}$ is multiplied by a scaling constant K . Each transformed pixel $a_{i,j} = Kb_{i,j}$. An array of PEs of size $i \times j$, could transform the entire array in one operation time period. However, in many image processing problems, $a_{i,j}$ would be a function of pixels within a local or global area. To facilitate these operations,

interconnection is provided between PEs. The topology of the interconnections determines the dimensionality of the PE array. Figure 7.2 shows two examples of 2-D interconnection topology.

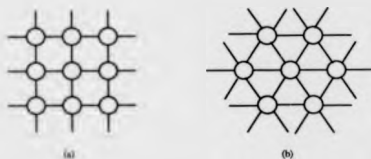


Figure 7.2: 2-D Processor Array Interconnection Topology. (a) Rectangular. (b) Hexagonal.

3-D interconnection involves the vertical stacking of such 2-D planes, or the formation of toruses etc. as with Li and Maresca's [103] polymorphic torus. Multidimensional interconnection topologies are also realisable. The PEs in an array can be SIMD or MIMD. MIMD implies a high level of processor autonomy, but some autonomy is possible for PEs within the SIMD definition [112], for example, some PEs may be switched off during a processing stage, or several array controllers may exist enabling separate tasks to be performed in different areas of the array. Autonomy in other forms is realisable, for example connection autonomy and addressing autonomy. Fountain [55] suggests that Flynn's [54] original taxonomy is unable to deal with the various PE autonomy issues. Maresca et al. [112] have published a tree structured taxonomy with Flynn's MIMD, MISD, and SIMD definitions as roots, the various autonomy and topological connection definitions as branches, and implemented examples as leaves. A taxonomy generalised for parallel computers in all data processing fields is provided by Skillcorn [162].

It is through a study of the history of implemented architectures that an understanding of processor arrays can be obtained. Fountain [55] provides an in depth study of such systems up to 1986. A special issue of the Proceedings of the IEEE on computer vision, 1988, edited by Li and Kender [102], provides survey papers on architecture by Cantoni and Levaldi [22], and by Maresca

et al. [112]. More recently, 1989, there has appeared a special section in the IEEE Transactions on Pattern Analysis and Machine Intelligence, on computer architecture, edited by Dyer [51]. Some of the main points are examined in the following sections.

7.4.2 Fine Grain Arrays.

Fine grain arrays are more likely to be used for low level image processing where there is an advantage in associating the array structure with the original data structure and applying connections between elements in local areas. Hexagonal interconnections where each processor connects to its six nearest neighbours are often realised, especially in systems designed for morphological image processing. Simple SIMD PEs are often used because arrays are large, many PEs can be integrated on each VLSI device, and the single instruction processing reduces communication overheads. Figure 7.3 shows a fine grain array together with some of the other units required to realise a system.

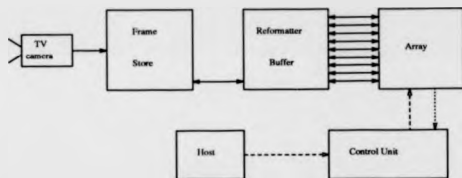


Figure 7.3: A Fine Grain Array System.

The control unit broadcasts image processing instructions to the PEs in the array, and receives back busy - finished signals from each. Many array processors are used with a raster scanned input device such as a TV camera. The TV picture is captured by a frame grabber and then reformatted so that efficient array loading can be achieved. Loading such images requires the hardware overhead of the reformatting logic, and a loading time overhead. Loading is often achieved by transferring a

complete column of data from the reformatting logic to the array, and rippling this and subsequent columns across the array until all columns are filled. In the Clip4 array [55] the data and control paths are separated so that image loading can be performed concurrently with image processing. This requires an additional hardware overhead. Direct light input to the array has been investigated in the UCPR1 [49] system. This had a 20x20 array of PEs each with a photodiode input. The Hughes wafer stack [130] has a front end wafer comprising an array of sensor elements.

The University College Clip4. The Clip4 system [55] is a fine grain SIMD processor that embodies the features of Figure 7.3. The Clip4 chip, which is used to assemble the array, was designed in 1974, and was limited by the fabrication technology available. Limitations included the number of transistors per device (5000), the packaging (40pin), and the clock speed (5MHz). The resulting device contained 8 PEs and has been used to build arrays of from 32x32 to 128x128 elements. The arrays can be connected in square or hexagonal 2-D meshes. The processor data width is 1-bit, and as a processing speed example, an 8-bit addition can be performed in 80 μ S. The processor can perform Boolean operations, a 32-bit RAM is provided in each PE, and input gating is used on the near neighbour input connections to facilitate efficient morphological operator implementation. Individual PEs can be switched off by certain processes, and a global propagation function allows data to be passed through the array 50 times more efficiently than if propagation is limited to near neighbour only communication. Clip4 arrays have been used for many applications [48]. A process involving the measurement of the rate of growth of biological cell cultures was possible, for a large number of samples, as computation could be performed in less time than it took physically to change the sample [55].

The Goodyear MPP. Improved specification SIMD PEs have followed the Clip4, for example the Goodyear MPP [12]. The MPP system employs an array of 128x128 PEs. Each PE has separate

logic and arithmetic processing units, enabling much faster arithmetic processing than is possible with Clip4. For example, an 8 bit multiplication can be performed in $10\mu S$ as opposed to $2mS$ for Clip4. 8 PEs are integrated on each chip, the clock rate is 10MHz, the pin out is 52, and each PE can be associated with external RAM. MPP systems have been used to process Landsat images. Here, the program is based on a sophisticated clustering algorithm. The final cluster of like valued pixels represents a feature.

The GEC GRID. A big step in SIMD technology has been the provision of autonomy in interconnection topology. The GRID [151] array system has a nearest neighbour switching facility, normally PEs are connected in a square mesh to their 4 nearest neighbours, but switching to the 4 corner neighbours is possible within one instruction cycle. For enhanced global calculations, such as histogramming, the "E" register in each PE is connected to the "E" register in the other PEs to form a large, single, shift register. Other registers enable the efficient rank ordering of values in a local area.

The MIT Connection Machine. The connection machine, reported by Hillis [77], provides greater PE connection autonomy. The first system comprised of 4096 VLSI devices, each containing 16 PEs, and a sophisticated message router. Two interconnection networks were incorporated in the array, a 4-way nearest neighbour connection, and using the router, and message packet switching, a 12-D Boolean n-cube was superimposed on the nearest neighbour mesh. However, there was only one router for each group of 16 PEs, and so only groups were connected by the n-cube network. Much of the PE function was involved with interconnection. The interconnection flexibility should improve the efficiency of global calculations, and the efficiency of high level processes that do not map so readily to the fine grain array - input data architecture.

7.4.3 Coarse Grain Arrays.

With coarse grain arrays, one PE will be associated with many data, or large local areas of pixels. In some systems memory may be shared among PEs. The PEs are likely to be sophisticated micro-computers, and considerable processing and communication autonomy will be devolved to them. The array is likely to be a MIMD processor. Communication overheads limit the number of processors that can be inserted in an array to obtain faster processing. For some processes, such as low level image processes, it may be advantageous to divide the image space and associate one PE to each local area. For higher level processes, the computer programmer may perceive an advantage in redistributing the processing in a different way across the array. This is easier with a shared memory system. The following are examples of coarse grain systems.

The Transputer. The transputer PE [87] is a sophisticated micro-processor and communication device integrated on a chip. It has 4 high hand-width serial ports that enable it to be connected to its neighbours to form a 2-D array. Other interconnection topologies are possible [5]. Various devices are available within the family offering options such as floating point processing and large local on chip RAM. Even additional externally connected RAM is considered as being local to the PE. Its price is too high for it to be a fine grain array element. It is used in various coarse grain arrays, and as a high level processor array element in systems incorporating a low level pre-processor. For example, Valkenburg et al. [180] are developing a system with a pipeline low level processor front end that feeds a transputer array, and Nudd et al. [131] are developing a 4 level pyramid machine with an 8x8 transputer array at the second highest level.

The Hypercube. A computer with n^2 PEs, each with a direct connection to $\log(n^2)$ others. The connection machine, discussed in Section 7.4.2 is such an architecture. MIMD cubes also exist such as the Intel iPSC [88].

The PRAM. The parallel random access machine is where there is a global stored memory access by all of the PEs. Variations exist with machines that allow PEs either to read from or write to the same memory locations simultaneously. PRAMs are discussed by Stout [167].

7.4.4 Pyramid Processors.

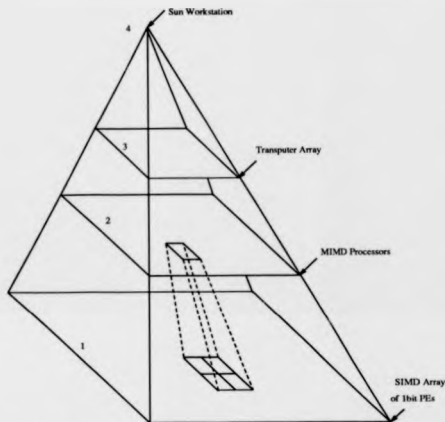


Figure 7.4: A Pyramid Architecture.

A pyramid architecture is shown in Figure 7.4. At the base of the pyramid, level 1, is the input image. This is connected upwards to level 2 so that four level 1 pixels connect to one level 2 pixel. This is known as a quadrature pyramid, binary, hexagonal, 16-way, and other connection systems have also been realised. In its simplest form the structure may be a pyramid of memory elements so that reduced resolution images are stored at each level, as with the pipelined parallel machine [19].

Pyramids of PEs are also realisable, with architectural variations in the types of PE at each level, and in the autonomy of control. PEs communicate between neighbours within their level, and also pass data upwards to their associated PEs at the higher level. Some processes require that data passes both up and down the pyramid, as with Watson and Ahumada's [184] data compression pyramid. PEs can work autonomously within the pyramid, or control can be passed down layer to layer from the apex.

In one type of pyramid, the PEs are of the same type in each level, the arrays can be coarse grained [70], or more usually fine grained [171]. In another type, different PEs will be incorporated at the different levels, as with Nudd et al [131], where level 1 is populated by a 256x256 array of SIMD PEs, level 2 by a 16x16 MIMD array, level 3 by a 8x8 transputer array, and level 4 by a host Sun workstation. Pyramid processors are efficient architectures for image understanding systems. The input is any general image at level 1 and the output would be a description of the scene in the form of say a list of objects at the highest level. This is the goal of computer vision and is currently beyond the scope of existing machines. It may be possible to map Marr's three stage representational framework (Section 2.2.3) onto such a pyramid, or perform 2-D scene analysis with low level processes performed in level 1, and mid level processes in level 2 etc.

Hexagonal Pyramids. Hartman and Tanimoto [74] investigated a hexagonal pyramid data structure for image processing. Level 1 (taking 1 as the lowest level) was tiled with hexagons, but each hexagon was subdivided into 6 equilateral triangular pixels. 4 triangles were then combined to give a single equilateral triangle at the next level, as shown in Figure 7.5, where level 1 triangles: [a,b,c,d] combine to give a level 2 triangle: [A]. PEs could also be arranged in such a scheme, but the basic triangular pixel scheme is difficult to sample directly with a raster scanned device as the image line spacings would be uneven. Resampling hexagonally sampled data to the triangular scheme could be achieved relatively easily.

Watson and Ahumada [184] present a hexagonal pyramid structure that models the processing structure in the human eye. Their technique is described in Section 2.2.3.1.

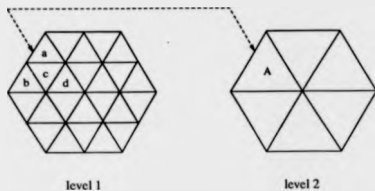


Figure 7.5: Hartman and Tanimoto's Pyramid Structure.

7.5 One Dimensional Arrays.

As described in Section 7.4.2, 2-D arrays of PEs map efficiently to 2-D images, but often the image sensor is a device producing a 1-D data stream. A 1-D array of PEs, operating on the data stream produced by the sensor may be more efficient for some processes as no reformatter buffer or array loading is required. Alternatively, a 1-D vector array of PEs can be swept through the 2-D image data. Only n PEs are required, as opposed to n^2 for a 2-D array.

A 1-D systolic array is shown in Figure 7.6. Here, each PE is operating the same instruction. There are two streams of input data, X and Y , and one output stream, Z . Stream X could represent image data, and Y , filter coefficients. Stream Z would be the filtered image output.

A pipeline processor can be similar to a 1-D systolic array in that the PEs can be performing the same instruction, but often each PE performs a different instruction. The input and output data streams are always pipelined. Data transfers between PEs are highly synchronised, but often in a pipeline of autonomous PEs, the longest process will determine the data rate through the pipeline.

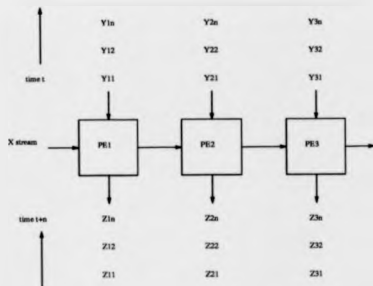


Figure 7.6: A Typical 1-D Systolic Array.

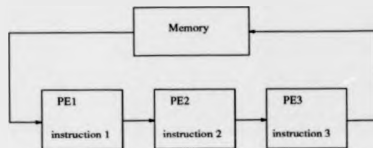


Figure 7.7: A Typical Pipeline Processor.

Often a pipeline has only one input data stream, the filter coefficients etc. are pre-programmed.

7.5.1 Vector Arrays.

Clip7A. This system is described by Fountain [55], it is a systolic linear array of 256 pairs of Clip7 PEs. The Clip7 was developed from the Clip4, but contains 8-bit input/output registers, 16-bit internal registers and ALU, and an external RAM interface on a single chip. Each PE can operate autonomously within the array, but the data streams, which are conceptually similar to those in Figure 7.6, enable the system to keep within the classification of a systolic array. Data is loaded serially into the array of PEs and stored in local memory associated with each PE. The array can

emulate a 2-D array of PEs, and at a higher processing level can be more appropriate for processing symbolic data sets. Clip7A has been used to simulate a 256x256 PE SIMD array and the concurrent processing of several smaller arrays. Locally directed database processing can be performed by the array and this is a pre-requisite to object identification. Figure 7.8 shows an application in which a 256 PE array is scanned through an image space containing two separate images. One string of PEs is processing image A and another string image B.

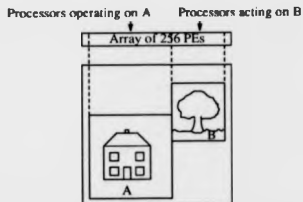


Figure 7.8: A Clip7A Linear Array Operating on Two Sub-images, A and B.

WARP. This 1-D array processor is classified by Maresca et al. [112] as a MIMD machine, with a linear network topology, and a floating point calculation capability. The system is a programmable systolic array processor. One of its design specifications was that it should support computer vision research. Warp was designed at Carnegie-Mellon University, USA, and the basic system is described by Gross et al. [69]. The array can be of 10 or more PEs, with each PE constructed from commercially available devices on a separate circuit board. It includes Weitek floating point multiplier and ALU devices. The PE internal architecture is explained further by Kung [97]. A VLSI version of the PE is planned.

The system was designed to process most low level vision algorithms. A library of 200 low level

algorithms was consulted to ensure that this was possible. Local operations such as convolution, and global operations such as the FFT and the Hough transform can be performed. Kung and Webb [99] describe the mapping of image processing algorithms onto the array.

The 10 PEs connect to a host computer that controls how the PEs will operate, and how they will be interconnected for a particular task. All 10 PEs can be connected in a pipeline with each PE performing the same or different operations, or the PEs can be connected to form several short pipelines. The PEs contain local storage for partial results which may be required by later stages of the algorithm execution. The interconnection topology and PE internal architecture are very flexible to allow for the wide variety of algorithms that can be processed. This flexibility also requires that address information must be transmitted along with the data through the array. The Warp system is a very fast processor, but some processes will give a larger performance increase over a SISD computer than others, depending on how efficiently the process can be mapped onto the hardware. Warp's architecture and its implementation are discussed further by Annaratone et al. [4]. The use of the Warp array together with a host computer for the control of an autonomous land vehicle is described by Crisman and Webb [32].

7.5.2 Pipelined Processors.

A typical industrial vision system might use a raster-scan TV camera and a converter to produce a 512 x 512 pixel picture, 50 or 60 times a second. Ideally, the image processing system should be able to process one frame of a picture before the next frame is available giving maximum system throughput. Such a system is referred to here as a 'real-time' system. Such a bandwidth may not be required for a particular application. Very fast processing can be achieved using SIMD and MIMD array processors. However, the cost and complexity of 2D arrays of PEs for industrial systems can be high. Many PEs are required for real-time processing, and large random access memories may be needed at each PE to store image and program data. Additionally, these systems must

employ a frame store and data reformatting logic which introduces a delay of one video frame time, unless extra hardware facilities that allow array loading during processing of the previous image are provided (as with Clip4 [55]), or the problems of optical image input and output to each PE can be solved [55].

As discussed in Section 4.4.2.1, with industrially derived images, or images captured under controlled lighting conditions, local image processing operations can be sufficient. Local image processing operations do not require a knowledge of the complete frame of an image, but only of a group of adjacent pixels. If these operations are performed on a pipeline processor there is no need to store a complete image. Such a processor may operate directly on the serial data stream from the digitised output of a raster-scanned device. The PEs must operate in real-time, but since the operations are very simple only a few lines of the image need be stored in line length shift registers within each PE. Figure 7.9 shows a pipeline processing element which stores two lines of the video image.

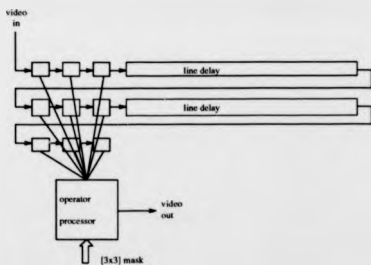


Figure 7.9: A Pipeline Processor Element.

In this example, three bytes of each of the two line storage registers, together with three bytes from the previous line, form a 3x3 local image area on which processing is performed. Real-time

processing operations are performed on this array of elements and a new value for the centre pixel is calculated and output to form the output video stream. Storing more lines of an image enables larger local areas to be used. For example a 5x5 pixel area could be used by storing four video lines. Processes of increased complexity can be achieved by cascading a number of PEs in a string.

Many pipelined image processing machines containing PE architectures based on that of Figure 7.9 have been reported in the literature. Few are limited to contain just this simple PE design but also have general purpose ALU and lookup table elements. If the Warp system, described in Section 7.5, is considered as a pipeline, then each PE can also contain local memory to aid multi-pass algorithm calculation. However, this feature causes Warp to be classified as a 1-D MIMD array. Some of these existing systems are now reviewed under the headings of Real-Time Pipelines and Re-Circulating Pipelines.

7.5.2.1 Real-Time Pipelines.

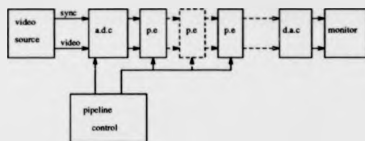


Figure 7.10: A Real-Time Pipelined Image Processor.

Figure 7.10 shows a typical real-time pipeline processor arrangement. A video source provides analogue video and TV synchronisation signals which are digitised by an analogue to digital converter (a.d.c.). These are fed into a pipeline of PEs. Each PE performs a separate operation on the data, and then at the end of the pipeline, the signal is converted back to an analogue signal by the digital to analogue converter (d.a.c.). In this case the TV synchronisation signals have also been passed down the pipeline and delayed equally with the data in each PE. The processed image can

be viewed on a TV monitor. The bandwidth of the system is a function of the number of pixels per TV line. This could typically be 512. The output rate of processed images is the same as the TV frame rate, and so such a system is classified as being a video rate, real-time system. There will, however, be some image latency in the system, depending on how many TV scan lines are stored in each PE and the number of PEs. This latency may limit the rate at which an object can be moved in an application where an image is enhanced for a human operator. In an alternative mode of use the pipeline could form a pre-processor for a host computer. The interface would probably contain a frame store buffer.

Early pipelined image processors are reported by Preston et al. [145] in a review paper published in 1979. The basic PE architecture of Figure 7.9 is evident in these systems, but at that time only the Cytocomputer was capable of real-time processing.

The Cytocomputer architecture is described by Loughheed and McCubbrey [106]. The PE design conforms closely to Figure 7.9, but with a programmable operator function. The PEs in the pipeline are hardware identical, but programmed for differing operations. The operator processor is limited to morphological and logical operations on a 3x3 local area. Scan line lengths up to 2048 pixels can be accommodated, and PEs are constructed on individual circuit boards from LSI and VLSI components.

The PIPE system developed by Luck [107, 108] is usually configured with from three to eight PEs. Each PE contains three look-up table operators, image combining units, a 3x3 arithmetic or Boolean local operator, and output crossbar switching logic. Images with resolutions from 256x256 to 1024x1024 can be processed. The pipeline forms part of a low level processing system, with an interface to a SISD computer which is used for high level processes. Other system components are four frame buffers, a global histogram processor, a video interface, and a system controller. The crossbar switching enables data to process normally along the pipeline, for data to be switched in reverse direction along the pipe, or for the PEs to operate independently. Morphological and

filtering operations are possible.

A University of California machine is reported by Ruetz and Brodsen [156]. This system provides a custom chip set for the designer. Each PE function is realised by a different VLSI chip. Advantages include real-time operation and potential cost reduction through the use of VLSI, but the non programmability of PEs has led to a dynamic inflexibility and a requirement to design a different chip for each PE function. The PE chips produced include a 3x3 convolver, a 3x3 sorting filter, an edge detector, a morphological operator and a contour tracer which locates closed contours after the edge detection stage.

A pipeline system designed at the University of Strathclyde is reported by McCafferty et al. [119]. The system uses LSI components, operates in real-time and employs sophisticated image processing algorithms. Unfortunately the system is limited to edge detection.

Mellroy et al. [123] have designed a real-time PE using LSI logic devices that performs the Roberts edge detection algorithm.

Lenoir et al. [100] have designed a PE, implemented on a gate array, capable of several binary and grey level morphological operations. The local operator size is 4x3 pixels.

Goldstein and Nagler [64] have designed a pipeline processor system for detecting surface defects in metal parts. Defects include scratches, splits, scale, dents, dirt, corrosion, and perforations in ammunition cases. Some defects are more severe than others for equal size, and so the system must identify the type. Each PE is a single board SIMD computer. The first PE is a morphological processor, the second a convolver, and the third an ALU. High level processes are performed by a host computer.

Various programmable single chip filters based on the architecture also exist, such as the Inmos A110 [87, 14]. This is a programmable device, with programmable line delays of up to 1120 stages, and a data throughput rate of 20MHz. It is capable of a range of operations. Template matching, 2-D convolution (3x7 local area), and image translations can be performed within a single PE.

However, it is not able, as a single chip, to perform such operations as the Sobel edge detector and the application of templates in parallel. The device contains 21 multiply and accumulate stages. These are arranged to implement convolution operations efficiently, but are unsuitable for operations such as those in grey-level morphology where each member of the local image area is multiplied by a separate coefficient value before a rank ordered selection is made.

7.5.2.2 Re-Circulating Pipelines.

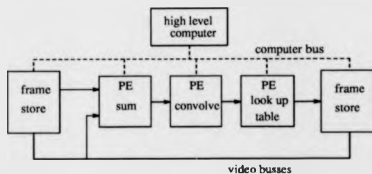


Figure 7.11: A Re-Circulating Pipeline System.

The systems described here are characterised by having only short pipelines of PEs, and by individual PEs in the line being of a different hardware construction. Frame stores are used to enable data to be re-circulated through the pipeline as shown in Figure 7.11, and many video and system buses are employed for data communication and system control. Systems are comprised of many separate function boards. These systems are however capable of performing a wide variety of complicated image processes, some of which can be classified as being in the mid level vision range. For example the convex hull process has been realised by Bowman [17] on a pipeline processor. First data is scanned horizontally out of one frame store, processed, and re-stored in the second frame store. The data is then scanned vertically out of the second frame store, processed, and re-stored in the original frame store.

The Cyto-HSS is a system, developed by McCubbrey and Loughed [105], that is based on a

pipeline of one or more Cytocomputer PEs. Other system components include a host computer, a video digitiser, a multiple frame store, an image combiner and auxiliary processors. The pipeline is capable of performing Boolean and morphological operations. McCubbrey and Loughheed [122] have reported its use for morphological image analysis, as have Crimmins and Brown [31].

Batchelor and Bowman et al. [9] review developments in image processing for industrial inspection, and consider the suitability of arrays, pipeline and SISR processors. They conclude that pipeline systems are likely to be the most cost effective. Bowman and Batchelor [18] have developed a system called Kiwivision. The standard system has a pipeline of three PEs, each performing a different set of operations. The first PE is a 16-bit ALU, the second a general purpose local filter, operating on a 3x3 local area, and the third a look-up table processor. Input/output modules and two 256x256 frame stores are employed. Memory addresses are passed through the pipeline and delayed equally with the data. The modules are linked by five sixteen bit buses together with a VME bus which also communicates with a 68000 based computer. This computer provides system control and high level processing.

Valkenburg and Bowman [179] describe a new system, Kiwivision II, which consists of a pipeline section for low level operations and an array architecture section for high level operations. The array is comprised of Inmos transputers, and the pipeline section of commercially available pipeline module boards. They intend to interface the PEs to the standard transputer communication links to enable high level control.

Datacube [34] have produced a series of single board processing modules that can be configured as a recirculating pipeline. The low level operation pipeline of Kiwivision II is constructed from these boards. The system is expensive, but its flexibility makes it an ideal tool for pipeline research.

In the PREP system developed by Wehner [185], several parallel recirculating pipelines are used to speed processing, by operating on distinct areas of the image. The system is used for morphological and image algebraic operations. The PEs are simple, only 16-bit integer arithmetic

is performed, but they have been realised in gate array technology.

7.6 The Choice of Architecture for an Industrial Vision System.

7.6.1 Introduction.

The reasons for choosing a pipeline pre-processor architecture for further development are explored.

As stated in Section 2.4.1 an industrial vision system must be cost effective, fast and reliable.

A robot will only be installed on a production line if it can be calculated that over its lifetime it will be more cost effective than an alternative method. A robot with vision is more versatile than a sightless robot, but to be cost effective the vision system must not add too much to the cost of the robot. The cost of a vision system can be divided into two parts, the basic hardware and software costs, and the application development costs. At a research stage, for systems biased towards industrial applications, the eventual minimisation of these costs and the ease of technology transfer need to be considered. As discussed in Section 7.6.3 it is expected with the proposed pipeline system to map one easily defined image processing operation on to each PE in the line. This will make the system easy to understand and lead to the fast development of applications. Assuming that a versatile system would be widely used in industry, and even used in the home, the use of VLSI components would be appropriate. Costs may be further reduced if the variety of VLSI components is minimised by designing re-configurable devices.

The VLSI components will eventually be assembled on circuit boards to realise complete systems or subsystems. Such circuit boards should be versatile enough to be programmed for a wide range of applications. The applications engineer needs to be presented with a development station which provides an interface between the high-order processes that are required, and the low-order programming of the hardware. It would be useful if the computation time for a process could be easily discernible by the application engineer at the original design stage rather than at the

simulation or testing stages.

For a simple vision application involving a low part presentation rate, a conventional SISD architecture computer may be sufficient. For applications requiring more complicated processing or a higher part rate, a parallel architecture may be a more economic or the only solution. Real-time processing is discussed in more detail in Section 7.6.2, but the economic considerations are discussed here.

A 2-D array of SIMD processors requires one PE for each data element. Simple PEs have been assembled in arrays of up to 256x256 elements (date 1987) [55], whereas, pipeline processors have been realised that can process image sizes of over 1000x1000 elements. For a 2-D array an increase in resolution of two times in each dimension requires a four fold increase in the number of PEs. The number of SIMD PEs that can be integrated on a single device has been limited in the past by the fabrication and pin-out technology [55]. Even with the large pin-out capability of modern devices, the number of integrated PEs will be small compared to the overall array size. Local PE interconnections are provided with various topological patterns enabling efficient local operator implementation, however global communications such as image loading are less easily performed. The Connection Machine [77], a composite of SIMD and MIMD processors, solves many of the communication problems, but as noted by Bowman [18] the cost precludes its use in industrial machine vision applications. Each of the processes identified in Chapters 4 and 6 could be implemented on a SIMD array, but as the PEs tend to be simple to reduce chip area, the time taken to calculate the more complicated operations such as the Sobel edge detector or a rank order selection will be long.

2-D arrays of MIMD processors can be used to compute low level operations in less time than an equivalent SISD processor. The programmer simply divides up the task between PEs. However as arrays become large, communications overheads become prohibitive. MIMD arrays are likely to be effective for high level processes that require global image decisions, data base

searches, or symbolic data processing. Valkenburg et al. [180] are researching a system which comprises a pipeline processor to provide low level image processing, and an array of transputers (MIMD devices) to provide high level processing. The pyramid processor is another system that comprises separate hardware for low and high level processes. The low level processes are often accomplished on a SIMD array at the lowest level of the pyramid, and the higher level processes by MIMD arrays at higher pyramid levels. The top level is often a SISD computer through which results are reported. Such a pyramid has been researched by Nudd et al. [131]. The large number of PEs required makes the cost of pyramids high and the problems of loading the image into the lowest level are the same as those for loading 2-D arrays. However, pyramids are very flexible machines and are capable of performing a wide range of image processes. For the processing of images captured in controlled lighting conditions the range of image processes required is reduced. A system comprising a pipeline processor for low level operations and a MIMD processor array for high level operations should be sufficient.

Of the 1-D arrays reviewed in Section 7.5 each looked promising for use in industrial inspection image processing. The systolic array, Clip7A, requires two PEs for every sampling point along the configuration dimension of the array. Each PE requires up to 64k bytes of local memory. One PE at each array position is used to access memory while the other processes the image. A minimum system for a 256x256 resolution image requires 512 PE and 256 local memory devices, with each PE integrated as an individual device. The device count is high, but the system can perform both high and low level operations.

The WARP system, reviewed in Section 7.5.1, was developed to be used in a research environment and is capable of a wide range of low level local and global operations. The cost of the system is high but may reduce if a VLSI PE version is produced in large quantities. Many of the image processing operations that WARP is capable of, such as the FFT are unlikely to be required to process images captured in controlled lighting conditions.

Pipeline processors are ideal for low level local processes, but have been inflexible in the past as a different PE design has been required at each stage in the line. Recirculating pipelines have overcome some of the inflexibility, but have reduced the processing rate of systems as the PEs must be repeatedly used to process the current image before the next is entered. If a single programmable PE could be identified and realised as a VLSI device, then long reconfigurable pipelines of PEs working at real-time rates on the direct video data stream could be economically realised. The length of the pipeline would be a function of the complexity of the process, and thus a more economic solution may be possible than with 1-D arrays such as the Clip7A. However it may not be possible to design a single PE capable of a wide enough range of processes. Such a PE design is considered in more detail in Chapter 8. System design can be simplified as ideally each processing operation selected by an applications engineer would be run on a single PE in the pipeline, the system would thus be clearly understood, and the total processing time estimated using straightforward techniques.

7.6.2 Real-Time Processing.

A real-time system can be defined as one that processes at the image input rate, or for an industrial system one that processes at the object presentation rate. Images acquired by a TV camera are usually updated 25 times a second, the frame rate of the most commonly used European system. A processor would need to complete operations on one image within 40ms. On a production line object rates are usually less than this video rate.

As discussed in Section 7.5.2.1 it is possible to realise a real-time pipeline processor that operates on the raster scanned data stream from a TV camera once it has been digitised. As many operations as there are PEs in the line can be accomplished in real-time, however there is a latency period before the processed image exits from the pipeline which depends on the number of TV lines stored in each PE, and the number of PEs in the pipeline. Long latency periods are often acceptable in industrial production line environments. For example an actuator used to remove a faulty object

may be located at a position further down the line from the visual inspection station.

With 2-D arrays, 1-D systolic arrays and recirculating pipelines, the object visual inspection rate is a function of the complexity of the inspection algorithm. Higher object rates can sometimes be achieved by adding more PEs, using faster PEs, or quite simply by reducing the complexity of the algorithm by changing the lighting conditions for example. Processing rate is increased in the Clip7A array by using two PEs per pixel, one to access memory and the other to process the data. Increasing the length of a recirculating pipeline may increase its speed. The ability to load a new image while the previous one is being processed can increase the image processing rate of a 2-D array. The most expensive option to increase the processing speed is to redesign the PE in a new technology.

Video rate real-time examples include an aid for glass blowers developed by Nixon [129]. He has implemented a shape comparison algorithm on a pipeline processor. The outline of the glass being blown is superimposed on a perfect outline and displayed on a TV monitor.

If processing time permits, a wider variety of algorithms can be realised on 2-D arrays, 1-D systolic, and recirculating pipelines than on a real-time pipeline. For industrial use, where the latency period introduced by a pipeline may not be significant, a pipeline PE that can be incorporated in a video rate real-time pipeline, or a recirculating pipeline appears attractive to research.

7.6.3 The Reliability of the Image Processes.

In a totally reliable system all measurements will be made to the required tolerance, and all objects or defects will be detected. As wide a range of processing algorithms should be implementable on the hardware as possible to enable this ideal to be approached. Considering the two classes of algorithms, local and global, most local algorithms have been implemented at some time on 2-D arrays, 1-D arrays, and pipeline processors, but global algorithms are more suitable for computation on 2-D MIMD arrays or 1-D arrays such as Clip7A or WARP. However as discussed in Section 7.6

the intercommunications overheads incurred with large 2-D MIMD arrays make them less suitable for use with low level processes.

The WARP machine has been designed with an architecture and large locally addressable memory (up to one mega-word per PE) so that global operations can be performed. Kung and Webb [98] report on the implementation of histogram, FFT, sorting, Hough transform and component labeling algorithms. The host makes global decisions, controls the process and prepares data streams for WARP to process. No processing times for these algorithms are given.

As yet there are few published global algorithms that work with Clip7A. Fountain [57] states that software will be upwardly compatible with Clip4 and outlines an algorithm for correcting spatial image distortions. Clip7A's major advance is that it can process some high level algorithms [56].

Pipeline processors, as presented in Section 7.5.2, have been built that are capable of performing many low level local operations. In an industrial inspection environment it should be possible to minimise the use of global low level operators. The FFT is a useful tool, but as stated by Micheli et al. [41] only simple filters and processes are required in situations where the lighting is controlled. There is unlikely to be any advantage in transforming processes to the Fourier plane. Brightness histogram evaluation can enable automatic threshold level calculation. This allows grey level images to be converted to binary images optimally. In applications where light levels change rapidly, such as in naturally light scenes, real-time histogram evaluation may be necessary, but for an industrial inspection, under controlled lighting conditions, the threshold level will need evaluating only infrequently. This task can be left to the host computer as a service routine.

Hough plane techniques can be aided by a pipeline in that edge points can be found and addresses for the Hough plane calculated, but the Hough plane will need to be managed and evaluated by the host computer. A powerful host comprising a MIMD array as suggested by Valkenburg and Bowman [179] would seem appropriate.

Distortion corrections and translation or rotation transformations can be performed on the host

computer. Translation and rotation transforms could be performed on a recirculating pipeline that allowed both vertical and horizontal scanning of a frame store. If necessary a specialist co-processor can be attached to the system to provide these functions. Suitable VLSI devices can be found in Yencharis' survey [189]. Hough transform and FFT transform co-processors also exist. Fisher and Highnam [53] present a linear scan line array architecture of SIMD PEs for a line finding Hough transform device. The device will be capable of line finding on a 512x512 image within 2 to 3 frame times. A wafer scale Hough transform processor is presented by Rhodes et al. [148]. Toshiba's IP9506 device [189] is capable of a 1k x 1k FFT with 32bit precision within 2ms.

7.7 Conclusions.

Many of the image processing systems surveyed in this chapter are expensive machines. They were conceived and built for the task of advancing the frontiers of computer vision research. For images captured in controlled lighting conditions, such as those often found in industrial environments, the range of low level image processing operators required can be limited to those that operate on small local image areas. The use of pipelines of PEs for the cost efficient processing of local area operators has been discussed. Apart from the reduced cost of the pipelined system other advantages of pipeline processors include:-

- The direct input of digitised data from a raster scanned sensor device such as a TV camera. There is no requirement for reformatting logic, additional circuits and extra communication channels to enable the loading of images as with 2-D arrays and some 1-D arrays.
- For a video rate, real-time pipeline processor the direct video input removes the need for the TV frame buffers that are required with many other architectures. This reduces the cost of the system further and can also increase the processing rate as the frame buffering process can introduce a frame time delay. The cost of a frame buffer is not high, but all cost reductions

that do not reduce performance will make a system more attractive for use by industry.

- The number of PE's in a non-recirculating pipeline is a function of the complexity of the problem, whereas with 2-D SIMD arrays and some 1-D arrays the number of PEs is a function of the spatial resolution of the images. For the pipeline the picture processing rate is independent of the complexity of the image processes, this is not the case with the other architectures discussed in this chapter.
- Pipeline technology can be easily transferred to the industrial applications development sector as the mapping of image processing operators to the PEs in the pipeline can be achieved on a one to one basis. This will enable systems to be easily understood. In the application development cycle the time taken for a result to emerge from the pipeline can be calculated as soon as the required operators have been identified. This time is constant, and is equal to the number of PEs times the individual PE latency. This may be calculated early in the development cycle.

The pipeline processors surveyed are only suitable for low level image processing operations. A system comprising a pipeline for low level operations and a MIMD array of, say transputers, for high level processing was identified as being suitable for the complete processing of images captured in an industrial environment.

Real-time and re-circulating pipelines were discussed. Real-time pipelines have a higher picture processing rate, but re-circulating pipelines are capable of a wider range of operations, as images can be scanned in various orientations before being processed by the pipeline. It is proposed that a PE that can be connected in either of these pipeline architectures should be designed.

The next chapter presents the design of a re-configurable pipeline processor element that maintains a maximum of operation flexibility and operates at a high data rate, while being inexpensive to produce.

Chapter 8

A Pipeline Processor Element.

8.1 Introduction.

A PE has been designed at a functional level that can be assembled into pipeline configurations to provide low level image processing. The images processed by such a pipeline would be able to be viewed directly on a TV monitor or transferred to a computer for high level image processing. The PE has been designed to operate at the video rate for a 512x512 sampled image.

Figure 8.1 shows a simple pipeline comprising an analogue to digital converter, two PEs, a digital to analogue converter and a control unit. PE (1) is performing edge detection and PE (2) binary line thinning. A novel feature of the PE design is that an image processing operation such as convolution, edge detection, median filtering, grey-level morphological, or a binary operation can be completely performed with a single PE in one pass of the image data. It is expected that the diverse range of image processing operations listed above will be able to be performed by a single re-configurable PE circuit design that can be fabricated as a VLSI device.

In the remaining sections of this chapter the details of a basic PE design are given, followed by suggested enhancements to allow the processing of hexagonally sampled images and additional image processing operations. Finally some new pipeline architectures that it will be possible to

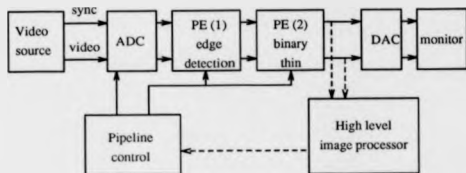


Figure 8.1: A Simple Pipeline Processor Consisting of two PEs.

construct with this PE design are discussed.

8.2 A Basic PE Design.

A basic PE which incorporates modules for processing many kinds of industrially derived square sampled images, but excludes modules that may be infrequently used, is now described. Possible enhancements to the basic design are discussed in Section 8.3.

8.2.1 The PE External Interconnections.

Figure 8.2 shows the PE input and output signals. The clock is at the pixel rate. There are two 8 bit image data input channels to each PE. In the figure, one channel is connected to the output of the previous PE in the pipeline, and the other to a second source which could possibly be the output from a second pipeline. The two channels are arithmetically combined inside the PE. Within the PE the image datum are clocked at the pixel rate through the various processing stages. A pair of unprocessed data are clocked in, and a processed datum clocked out from the PE with every clock pulse. The bandwidth of the PE is equal to the video rate, but the pipelining of the processes within

the PE introduces a latency equal to an integer number of pixel clock periods.

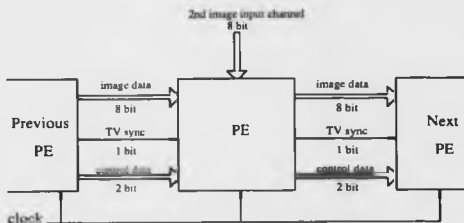


Figure 8.2: PE Input and Output Signals.

Signals derived from the TV line and frame synchronisation are piped along with the image data through the system. The PE delays these signals equally with the image data so that in a video rate pipeline the processed images can be viewed directly on a TV monitor. For images emanating from a host computer a frame synchronisation signal is used to indicate the start of a new image.

The control data is sent serially to the PE from a central pipeline control unit. A high bandwidth communication link allows the PEs to be re-programmed during the image inter-frame time. The network used for the control communications is a ring in which the control unit connects to the first PE. This then re-transmits the signal from a separate output to the next PE and so on along the pipeline until the last PE re-transmits back to the control unit. Each PE has a separate address, which is set by a power on - reset initialisation routine. During this routine, the first PE receives an address from the control unit, stores it, increments it, and then passes it on to the next PE in the pipeline. Eventually the control unit can calculate the number of PEs in the line. In normal operation the control unit would circulate the addressed packets of control information around the

system. The addressed PE temporarily stores the control information until a frame synchronisation pulse is detected. The new control information can then safely re-program the PE.

The input and output video ports are 8 bit wide as this word width would appear to have become a standard amongst manufacturers of peripheral equipment such as frame stores. Computer memory is also usually organised in 8 bit bytes. Studies of the number of grey levels a trained human observer can distinguish in a monochrome image indicate this to be $2^6 = 64$. The $2^8 = 256$ levels in an 8 bit system are therefore adequate. Many of the PE's internal connection paths have been allowed to have greater widths to maintain accuracy during numerical calculations.

8.2.2 Processor Features.

8.2.2.1 Real-Time Processing.

A real-time system can be defined as one that processes at the image input rate, or for an industrial system one that processes at the object presentation rate. Images acquired by a TV camera are usually updated 25 times a second, the frame rate of the most commonly used European system. A processor would need to complete operations on one image within 40ms. On a production line object rates are usually less than this video rate.

The image resolution has nominally been limited to 512x512 for a square sampled image with an overall square shape. The system is designed for compatibility with timing signals and data rates produced by cameras employing the non-interlaced 625 line, European TV monitor standard [118]. The system control unit provides sampling signals to the analogue to digital converter at a rate so as to maintain the square or rectangular spatial grid of points. However the PE will process data captured from cameras employing similar TV standards so long as the data rate is not too high. For 512x512 resolution video rate processing, it is simplest to allow the vertical pixel spacing to be one TV line, and so the vertical resolution nominally extends to 625 for this TV standard. The TV

aspect ratio is 4:3, and so the number of pixels on one TV line becomes $625 \cdot 4/3 = 833$. Each TV line lasts for $64 \mu s$ and so traverses each pixel in $64/833 = 76.8 ns$. The PE input data rate is therefore $13.0 \text{ MBytes} \cdot s^{-1}$. For a re-circulating pipeline it is advantageous to process the data as fast as possible, but for this initial device the specification of the data rate is limited to $13.0 \text{ MBytes} \cdot s^{-1}$. The brightness of each input pixel is binary coded as an 8 bit byte giving a brightness range from 0 to 255. In summary:-

- Image resolution: Up to 833×625 .
- PE data rate: $13.0 \text{ MBytes} \cdot s^{-1}$.
- Brightness range: 0 to 255.

8.2.2.2 Image Processes to be Implemented.

For this basic design the local image area operated on by the image processes is limited to a 3×3 square area. With such an area the range of image processes identified in Chapters 4 and 6 as being useful in a controlled lighting environment can be investigated without too much repetition of cellular hardware blocks such as TV line delays and multipliers. This enables ideas for circuit modules to be tested at a lower cost. Design problems associated with expanding the device's capabilities at a later stage are considered in Section 8.3. The image processes to be implemented have been grouped together below. All the members of a particular group require similar hardware to execute the process.

Convolution operators. Convolution operators have been introduced in Section 4.4.1.2. They require the 3×3 local image area to be convolved with a 3×3 array of coefficient values. Low and high pass filters, and point and edge enhancement operators are included in this category.

Edge detection operators. Edge detection operators have been described in Section 4.4.1.3. They require the 3x3 local image area to be convolved with two 3x3 arrays of coefficient values. Each convolution produces an edge response vector that is orthogonal to the other. The magnitude of the edge vector can then be calculated, for example by Equation 4.4, and the edge angle by Equation 4.5. The edge magnitude can be output to implement an orientation non-specific edge enhancement filter, or for edge detection, it can be compared to a threshold value to determine if a significant edge exists. The edge detector has a binary output, where 255 indicates the presence of an edge, and 0 no edge. The Sobel detector, described in Section 4.4.1.3, has been implemented as it requires integer only coefficients, and has been shown in Chapter 5 to produce accurate results. Only the edge magnitude can be calculated by this basic PE. The edge angle is required by some mid-level processes, and it is envisaged that this information could be calculated and output as a separate channel in a later PE design. The basic hardware will also permit the use of other edge detectors employing differently valued convolution arrays.

Sort and select operators. Here, sort and select operators include median filters, introduced in Section 4.4.1.2, and grey level morphological operators, introduced in Section 4.4.1.5. With median filters the local area is first masked to identify a sub-set of local area pixels. Often the median of the entire 3x3 local image area is required, but, as noted by Bovik et al. [16], sometimes a "+" or a "x" shaped local area is required if edges of a particular orientation are to be preferentially preserved. The sub-set of the local image area can be selected by multiplying the local area with an array of appropriate 0 and 1 valued coefficients. The median of the values in the sub-set of the local image area is then selected.

With grey-level morphological operators a 3-D structuring element is applied to the local area and then the required rank ordered value is selected as the output value. The application of the structuring element involves array multiplying the local image area with an array of coefficient

values.

Parallel binary morphological operators. Binary morphological operators have been described in Section 4.4.1.5. Some of these operators require several masks to be applied to the local area in parallel, as for example in line thinning where vertical and horizontal thinning masks need to be applied, or with the optical character thinner devised by Chin et al. [27] and discussed in Section 6.3 where ten masks must be applied. It should be noted that Chin's masks require a 4x4 local image area to be assembled. Modifications to the basic PE design to achieve this are discussed in Section 8.2.3.4. The masks contain 1, 0 and don't care values, and the local image area contains binary values 0 and 255. The masks are applied in three stages, firstly the local image area is processed by a sub-mask to switch the don't care conditions to 0 and then logical operations are performed to determine which masks fit the image data patterns. Finally further logical operations combine the mask outputs to produce a single binary output value.

In summary,

- Local area size: 3x3.
- Convolution operators have been implemented.
- Edge detection operators have been implemented.
- Sort and select operators have been implemented.
- Parallel binary operators have been implemented.

8.2.3 A Functional Description of the PE Hardware.

8.2.3.1 System Overview.

The PE hardware is based on that shown in Figure 7.9 of a simple pipeline processing element. A block diagram of the basic PE designed here is shown in Figure 8.3. The video image enters the PE as a raster scanned 1-D stream and the 3x3 local image area is assembled by employing two TV line length, 8 bit wide, digital shift registers. The image adder allows two separate images to be combined at the PE input.

The information in the 3x3 local image array consists of 9x8 bit values, these are input to multiplier array (A) and, for edge detection operators, to multiplier array (B). In the multiplier arrays the individual pixel values in the local area are multiplied by the coefficient values in an array that is pre-loaded into the PE via the control information communication channel.

Up to this point the hardware modules are identical for the processing of each of the 4 groups of operator types listed in Section 8.2.2.2 except for multiplier array (B) which only needs to operate during edge detection processes. The next block in Figure 8.3 is labelled "Selectable operator". Here a different hardware sub-module is switched in to the circuit depending on which of the 4 processing groups the required image processing operator is a member of.

The final module in the video processing path of the PE before the data is output contains the image post processes. These are selectable and common to all the image processes. The post processes implemented are a thresholder, an image inverter, and an image lateral shifter.

The TV synchronisation delay module is a 1 bit wide shift register that introduces a delay to the synchronisation signals equal to the delay of the video data as it passes through the PE. The PE control module interfaces to the control communications ring. If a packet is intended for the PE it is loaded into a holding register before being transferred to a control register during the image frame synchronisation period. Signals from the control register control the other modules within the PE.

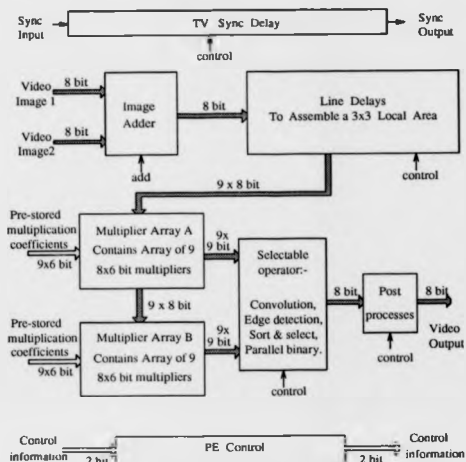


Figure 8.3: A Block Diagram of the Pipeline Processor Element.

8.2.3.2 The Local Area, Line Delays, and Associated Modules.

The Image Input Section. The image input section includes the input image combining ALU and the TV line delays required to assemble the 3x3 pixel local image area as shown in Figure 8.4. Two 8 bit wide line delays, each of 833 pixels each are required. Programmable taps are inserted in the line delays to enable lower resolution video real-time images to be processed, and for more efficient processing of pre-stored images in re-circulating pipeline applications.

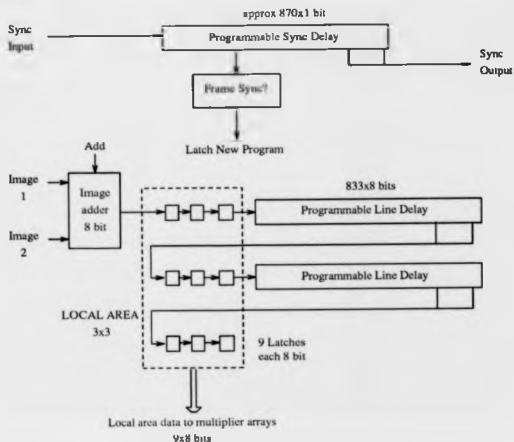


Figure 8.4: The PE Input Section, Line Delays and Local Area.

The PE can be used to process 833×625 images, the extra resolution has been included so that square real-time video images can be processed. Outside the required 512×512 image data-set, pixel values can be zeroed, but the PE will process these pixels exactly as if they were normal image pixels. A datum entering the PE must wait a latency period of more than that introduced by one scan line delay before it reaches the centre of the local area and can be modified. The modification processes are also highly pipelined, before an 833×625 data set completely exits from the PE the next set will already be being processed. If images were presented to the PE without padding data between images and at the end of scan lines, then the PE would operate on image edge

pixels producing modified pixels that were a function of the pixel's values in the current image and a neighbouring image, or a function of pixels on the opposite side of the image. Line and frame synchronisation signals enter each PE and could have been used to switch to a no operation state in the region of the image edges, but this would be complicated in such a PE containing many pipelined sub-modules.

With this PE design, processing 512×512 images, padding pixels can be added in up to 113 lines vertically and up to 321 pixels horizontally. In real-time video applications these would all be processed by the PE, and generated as a separate stream or accepted directly from the analogue to digital converter. In re-circulating systems the frame processing rate can be increased by minimising the number of padding pixels. Various processing possibilities exist:- Padding pixels can be set to 0, 255 or the edge pixel value can be extended outwards. In a long pipeline of, say, Gaussian filtering stages the effects due to an edge discontinuity may propagate significantly into the processed image area, whereas, with other processes such as median filtering, this will not occur. With edge detection operators a discontinuity at the edge will lead to a border around the image. The image resolution, the choice of the extent of the padding, and the padding values can be left to the application engineer as program selectable multiple taps are provided in the line delays. For this basic PE taps are inserted at 833, 562 (ie $512 + 50$ padding pixels), and 70 (ie $64 + 6$ padding pixels).

The Input ALU. The input ALU is used to combine two input images. A wide choice of combining operators could be provided, for example addition, subtraction, logical, or scale and then combine operators are possible, but for PE simplicity only image addition followed by division by two to scale the result back to 8 bits is provided. Other functions can be obtained by scaling or inverting the images in earlier PEs.

The Synchronisation Signals. The synchronisation delay line is one bit wide and conveys both frame and line synchronisation pulses. These pulses are obtained from the video source, or from the pipeline system control unit in a re-circulating system. A simple combinational logic circuit is required to recognise the frame synchronisation pulse, which is relatively wide compared to the line synchronisation pulse, and latch new control configuration information into the PE during this period. At this time false output images due to re-configuration during the passage of a frame of data will not occur. Selectable taps are provided in conjunction with the line delay taps to enable various image resolutions etc. The exact length of this delay line cannot be determined until the entire PE is fully designed for implementation in a particular technology and the depth of the pipelined processing stages known.

8.2.3.3 The Multiplier Arrays.

There are two 3×3 multiplier arrays labelled Array (A) and Array (B) in Figure 8.3. Multiplier array (A) is used by all of the image processing operators implemented within the PE, whereas multiplier array (B) is required only by the edge detection operators. Within each multiplier array each of the nine, eight bit values assembled in the local image area are multiplied by each of nine signed six bit fractional coefficients that have previously been loaded into the PE via the communications channel. The use of fractional numbers for the coefficients enables the scaling of the multiplier outputs to be achieved simply by rounding and truncating them to nine bit signed numbers. These output numbers can also be considered as being fractional as the image data can be considered as fractional. The precision of these values have been shown by the high level simulation presented in Chapter 9 to be adequate for the image processes involved.

Figure 8.5 shows a nine element multiplier array together with a definition of the input and output signals. The inputs come from latches in the delay lines or from storage latches in the PE control circuit. A latch clocked at the pixel rate is included at the output of each multiplier to

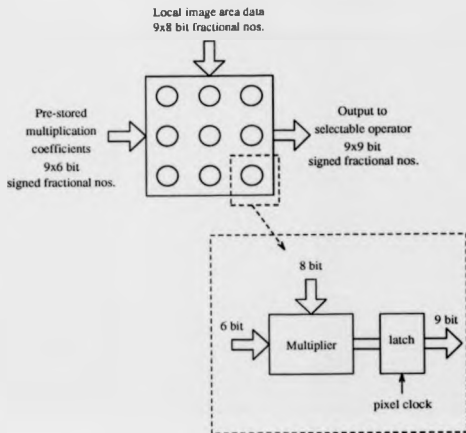


Figure 8.5: The Multiplier Arrays.

temporarily store the result. This latching arrangement sets the operation time of the multiplier to be equal to one pixel clock period. The pixel clock period is at least 70nS.

8.2.3.4 The Selectable Operator.

As can be seen in Figure 8.3 the selectable operators have as inputs the two groups of 9x9 bit signed fractional numbers from the multiplier arrays. The four operator modules are shown in Figure 8.6. Except for edge detection, only one of the four is used by each of the image processing operators that can be processed by the PE. The types of image process performed by each module are discussed

in Section 8.2.2.2. Each module produces a single 8 bit unsigned output word for every pixel rate clock pulse applied to it.

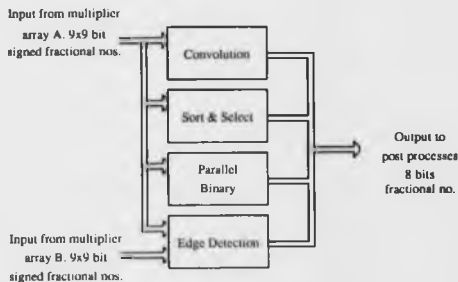


Figure 8.6: The Selectable Operators.

The Convolution Operator Module. Within the PE, 2-D convolution involves the array multiplication of the 3×3 local image area with a 3×3 array of convolution coefficients in multiplier array (A), followed by the addition of the nine multiplier array outputs and a scaling operation both performed within this module. Figure 8.7 is a block diagram of the module.

The module inputs are labelled $A_{r,s}$ and are nine bit fractional numbers from multiplier array (A). The adder tree produces the 13 bit signed result $X = A_{11} + A_{12} + A_{13} + A_{21} + A_{22} + A_{23} + A_{31} + A_{32} + A_{33}$ within one pixel clock period. This result is then latched. Next a 13x6 bit multiplier scales the result. The signed 6 bit scaling coefficient is stored within the PE after being transmitted to the PE via the communications channel. The 9 bit result from the multiplier can now take one of two routes, it can be sent to the edge detection module where it is further processed as detailed

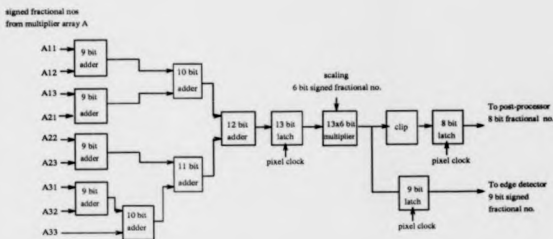


Figure 8.7: The Convolution Operator Module.

below, or it can be clipped to an 8 bit unsigned value in the range 0 to 255 and latched before being output to the post processing module.

The scaling introduced by the multiplier and the selection of the most significant bits to produce a signed 9 bit output have been shown by the simulation results reported in Chapter 9 to produce sufficiently accurate results for the majority of the required image processes. During the simulation it was found to be useful to report instances when the clipper module had to limit values to 0 or 255. Such information is sent to the PE control unit for communication to the overall pipeline control unit.

The Edge Detection Module. Edge detection operators have been described in Section 4.4.1.3. Three stages are involved in their calculation:-

- Convolution with two masks to provide, for a square sampled image, two orthogonal edge response vectors. Masks developed by Sobel, and described in Section 4.4.1.3, were shown in Chapter 5 to produce accurate results.

- **Magnitude calculation.** For a square sampled image, the edge magnitude can be calculated from the two orthogonal edge response vectors using Equation 4.4. This is repeated here, where t_h, t_v are template responses:-

$$m = (t_h^2 + t_v^2)^{1/2}$$

- **Magnitude thresholding.** The edge magnitude is compared to a threshold value that has been stored in the PE to determine if a significant edge exists.

Within the PE these three stages are calculated in different modules. One convolution is performed in multiplier array (A) followed by the selectable convolution operator module, and the second by multiplier array (B) followed by an additional convolution operator module. The magnitude calculation is performed by a sub-module, which together with the additional convolution operator, and an output latch clocked at the pixel rate, form the selectable edge detection module. The thresholding is performed by a module in the post-process unit, as described in Section 8.2.3.5. Figure 8.8 shows a block diagram of the edge detection modules.

To implement Equation 4.4 directly two multiplication, an addition, and a square root function are required. The multiplication and addition functions can be easily achieved, but the square root function is more difficult. As noted in Section 9.4.2.3, for relative magnitude comparison $m^{1/2}$ is not required, but for edge enhancement filters it is. An efficient complex number to magnitude approximation, based on the four region approximation algorithm, is given by Filip [52]. The magnitude is given by :-

$$m = \max \left\{ t_h, \frac{2}{3}t_h + \frac{1}{2}t_v, \frac{1}{2}t_h + \frac{2}{3}t_v, t_v \right\}$$

Where t_h and t_v are positive. This can be written:-

$$m = \max \left\{ G, \frac{2}{3}G + \frac{1}{2}L \right\}$$

where

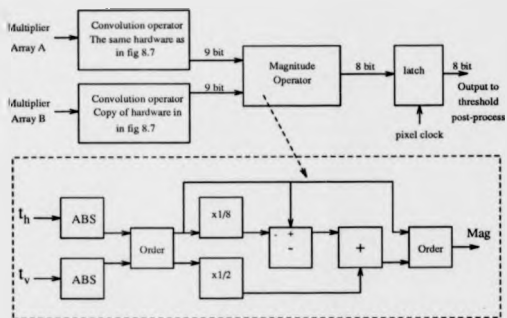


Figure 8.8: The Edge Detection Selectable Operator.

$$G = \max \{t_h, t_v\}$$

$$L = \min \{t_h, t_v\}$$

A simulation of 500 pairs of t_h and t_v values randomly chosen in the range 0 to 255 produced a maximum error of 3.2%. A higher accuracy could be obtained by including more regions in the approximation. A signal flow diagram of the algorithm is given by Denyer and Renshaw [42]. This has been reproduced in Figure 8.8. The first blocks produce the absolute values of inputs t_h and t_v . The leftmost "order" block selects G and L . The value of $\{G, \frac{7}{8}G + \frac{1}{2}L\}$ is then calculated, and finally the rightmost "order" block selects the value to be output as m . It is expected that this signal flow graph will be able to be efficiently implemented as a circuit.

The Sort and Select Module. This is used by median and grey-level morphological filters which are described in Section 4.4.1.2. The pixels of the local image area must be sorted in order of

magnitude so that the mean, maximum, minimum, or other ordered value can be found.

An efficient sort and select algorithm has been developed by Danielsson [33] based on an examination of the bit content of variables and successive approximation techniques. An example is used to explain its operation. The following variables are involved:-

- T = The rank order number of the wanted value.
- $\{A_1, A_2, \dots, A_n\}$ = The set of input arguments.
- $\{B_1, B_2, \dots, B_k\}$ = The column bit vectors of the A_i 's.
- B_0 = The initial column vector.
- N = The accumulated sum of arguments considered to be converging towards T .
- $\{S_1, S_2, \dots, S_k\}$ = The bit vector indicating the discarded arguments in different steps of the algorithm.
- $\{n_1, n_2, \dots, n_k\}$ = A vector indicating bit differences between B_j and B_{j-1} in input arguments that are not indicated as discarded by S_j .
- $Count$ = The number of bits valued true or "1" in n_j .

For this example let $T = 5$ and the arguments be seven four bit numbers {3, 13, 12, 6, 8, 6, 14}. The relationship between these numbers, A_i , and B_j is given in Table 8.1. If the vector B_1 of the most significant bits is inspected, the number of arguments starting with "0" is seen to be 3, which is less than $T=5$. This allows the set of arguments $\{A_1, A_4, A_6\}$ to be removed from further consideration. This is the basis of how the algorithm works, the $\{B_1, B_2, \dots, B_k\}$ vectors are successively examined and k arguments removed if certain rules are satisfied. After the inspection of each B_j only one argument, the rank ordered value, remains.

Table 8.2 details the process for this example. The four B_j columns of Table 8.1 have been expanded by the addition of n_j and S_j vectors and a left hand column has been added, labelled B_0S_0 , containing vectors with each bit initialised to logic "1". The variables $Count$ and N together with the results of the test $N < T$ are included at the foot of the table for each column. Each bit in n_j is calculated by the Boolean equation:-

	B_1	B_2	B_3	B_4	Decimal
A_1	0	0	1	1	3
A_2	1	1	0	1	13
A_3	1	1	0	0	12
A_4	0	1	1	0	6
A_5	1	0	0	0	8
A_6	0	1	1	0	6
A_7	1	1	1	0	14

Table 8.1: The Arguments to be Sorted, A_i , and the Corresponding Bit Vectors, B_j .

Decimal	B0 S0	B1 n1 S1	B2 n2 S2	B3 n3 S3	B4 n4 S4
3	1 1	0 1 0	0 0	0 0	0 0
13	1 1	1 0 1	1 0 1	0 1 1	1 0 0
12	1 1	1 0 1	1 0 1	0 1 1	0 1 1
6	1 1	0 1 0	0 0	0 0	0 0
8	1 1	1 0 1	0 1 0	0 0	0 0
6	1 1	0 1 0	0 0	0 0	0 0
14	1 1	1 0 1	1 0 1	1 0 0	0 0
Count		3	1	2	1
N	0	3	4	6	5
N<T		1	1	0	0

Table 8.2: The Rank Ordering Process.

$$n_j = (B_{j-1} \div B_j) \cdot S_{j-1} \quad (8.1)$$

Initially n_1 is calculated from the values in vectors B_0, S_0 , and B_1 . For this column if a "0" existed in B_1 then a "1" will now exist in n_1 . The number of "1's" in n_1 is then counted and displayed in the table as variable *Count*. Here *Count* = 3. *Count* is then added to the accumulated count in variable *N*. $N = 0 + \text{Count} = 3$. *N* is then compared with the required rank order number, $T = 5$. Here $N < T$ and so the most significant bit of the rank ordered result is a "1". If $N \geq T$ then it would have been a "0". The set of arguments $\{A_1, A_4, A_6\}$ have been removed from further

consideration as they each start with a "0", and the set of possible arguments that will provide the solution is $\{A_2, A_3, A_4, A_7\}$. The vector S_1 can now be calculated:-

$$S_j = (\overline{N < T} \oplus B_j) \cdot S_{j-1} \quad (8.2)$$

A "0" in S_j indicates that the A_i argument contributing to the corresponding B_j has been removed from further consideration. The accumulation of N in the remaining stages is dependent on the result of the comparison $N < T$ in the previous stage.

$$\text{If } (N < T) \text{ Then } N = N + \text{Count} \quad \text{Else } N = N - \text{Count} \quad (8.3)$$

Moving to the next column, n_2 , Count , N , $N < T$, and S_2 can now be calculated. In Table 8.2 B_2 bits that are indicated by S_1 as removed from consideration have not been printed. The result $N = 4$ is less than $T = 5$ and so the second most significant bit of the sorted value is a "1". The calculation of S_2 indicates that the set of possible values for the solution is now $\{A_2, A_3, A_7\}$.

For the next column, $N = 6$ and $N \geq T$ so the third most significant bit of the sorted value is "0". The calculation of S_3 indicates that the set of possible values for the solution is now $\{A_2, A_3\}$.

For the final column $\text{Count} = 1$ but equation $N = N - \text{Count}$ applies, as in the previous column $N \geq T$. Now $N = T$ so the least significant bit of the sorted value is "0". The final sorted value is "1100" or decimal 12. This was the third value in the original set of A_i arguments. S_4 now contains a single "1". The position of the "1" can be used to point to the ordered value as an alternative to assembling it from the $N < T$ decision values at each stage.

Danielsson [33] does not provide a formal proof of the method or develop Equations 8.1, 8.2, or 8.3. Within the PE the number of eight bit arguments in the local area from which the rank

ordered value can be selected is nine. A computer simulation of the algorithm with processing steps based of the circuit blocks presented in Figure 8.12, and configured for the rank ordering of the nine, eight bit arguments was run. Ten thousand randomly selected sets of nine arguments were sorted in rank order (ie the program was run nine times with $T = 1 - 9$ on each of the 10,000 sets), and the results compared with those produced by a computer library function sort routine. No differences were found between the orderings produced by each method, and it was concluded that Danielsson's algorithm performs correctly with typical pixel values to be found within the PE.

Danielsson presented three circuits that implement the algorithm, each employing progressively more parallelism. The higher the parallelism, the quicker the operation, but the number of circuit components required is larger. If n is the number of input arguments and k is the length of an argument in bits, then for the first scheme two k -bit shift registers, an n -bit zeros counter, a one bit subtracter, and a few simple combinational gates are required. The calculation time is proportional to $n.k$. The second scheme requires n , k -bit shift registers to sequence the arguments in parallel through the sort and select hardware which comprises two n -bit registers, an n -bit adder-subtractor, and an n -bit counter. The calculation time is proportional to k . In the third scheme the n , k -bit shift registers are retained as a means of loading the arguments, but their outputs are now available in parallel and connected to n copies of the remainder of the basic scheme 2 circuit. The calculation time is now independent of both n and k . For this PE scheme 3 is likely to be necessary to calculate the rank order within the inter-pixel input time.

Danielsson's block diagram for the scheme 3 hardware is shown in Figure 8.9 with the addition of word widths relevant to the PE. A diagram of his logic cell, nine of which are included in the Figure 8.9 block marked "logic", is shown in Figure 8.10. There are errors in both these circuits that prevent them from implementing the algorithm. In Figure 8.9, $N < T$ is calculated in each stage and then output to the "logic" block in the following stage to be used in the calculation of S_{j+1} . To operate the algorithm correctly, $N < T$ must be connected to the "logic" block in the current stage.

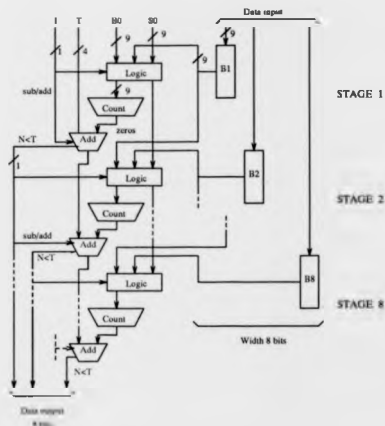


Figure 8.9: Danielsson's Circuit to Perform the Rank Ordering Algorithm.

and used to calculate S_j .

Several errors exist in the logic cell diagram of Figure 8.10. The B_j and B_{j-1} inputs have been swapped, S_j is used in the calculation of n_j instead of S_{j-1} , and errors exist in the calculation of S_j . A corrected logic cell is presented in Figure 8.11. The operation of this cell conforms to Equations 8.1 and 8.2 for the calculation of n_j and S_j .

Figure 8.12 shows a corrected processing stage for level j . The corrected logic cell has been split into n_j and S_j subcells and combined for nine bit operation in "Logic Blocks 1 and 2". As described previously in this section, the functional operation of the circuit has been proved to be correct by the use of a computer programmed high level simulation.

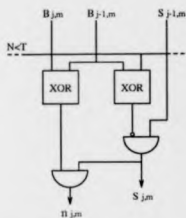


Figure 8.10: Danielsson's Logic Cell.

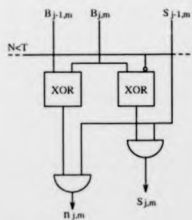


Figure 8.11: The Corrected Logic Cell.

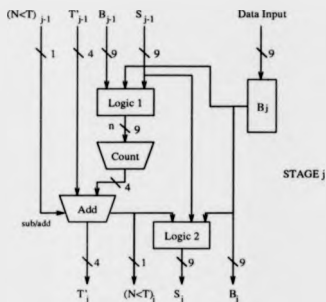


Figure 8.12: A Corrected Stage of the Circuit to Perform the Rank Order Algorithm.

The logic depth of the circuit is high and it will need to be pipelined in order to operate at the pixel rate. Latches, clocked at the pixel rate can be placed in the output lines from each of the eight stages. This will require between 23 and 85 latches per stage depending on the stage level. If the fabrication technology requires it pipelining can also be introduced within each stage.

The Parallel Binary Operator Module. This is used by binary morphological filters, described in Section 4.4.1.5, and by operators such as optical character thinners, described in Section 6.3, that require several templates to be applied to the local area before a decision on the value the output pixel will take can be made. Some of these operators require a local area of up to 5×5 pixels to be assembled, whereas in this PE the local area is limited to 3×3 pixels. In Section 8.3.3.2 a method of assembling a 5×5 binary pixel local area which requires little extra hardware is described. The binary image data must enter the PE in the least significant bit of the pixel value. Up to ten parallel templates can be applied in parallel, and these can be divided into two groups. If one of the action

group of templates is true then an action such as thinning is initiated, but if one of the **restore** group is true then the action will be overruled.

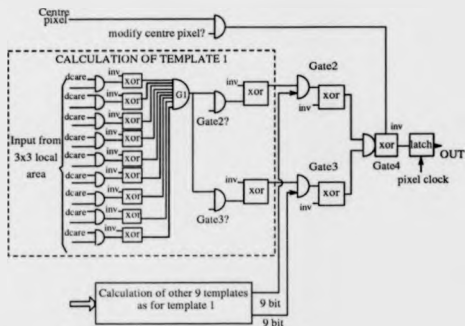


Figure 8.13: The Parallel Binary Operator.

Figure 8.13 is a block diagram of the programmable logic that applies the ten parallel templates. The templates contain "1", "0", and "don't care (dcare)" values. The following sequence of events are performed:

- Each template is split into one template containing don't care conditions, and one containing "1's" and "0's". The template pairs are then applied in ten identical logic circuits. The circuit for template 1 is shown expanded in Figure 8.13.
- The outputs of the template application circuits can be switched to either of two gates, "Gate2" or "Gate3". "Gate2" detects the outputs from the **action** templates and "Gate3" the outputs from the **restoring** templates.

- Finally the value of the output pixel is determined by Gate4.

At each of the stages listed above the operation of the circuit is programmable. Control information received from the communications channel is used to configure the logic. Examining the Template 1 calculation circuit in Figure 8.13, the leftmost column of nine AND gates are used to apply the "don't care" condition sub-template. If *dcare* = 1, then the pixel value is reset to "0". The next column of nine exclusive OR gates are used to switch expected "0" value pixels to "1's". If a template fits the pixel data then the output of each of these gates will be "1" as will the output of the nine input AND gate labelled G1. The output of G1 can be connected to the **action** gate, Gate2, or to the **restore** gate, Gate3. It is also possible to invert this signal under program control in the XOR gates in the input lines to Gate2 and Gate3.

The Template 1 signals are ANDed with the other nine template signals in Gate2 and Gate3. The gate output signals can be inverted under program control before they are ANDed in Gate4. The ANDed signal can be output or used to toggle the original centre pixel value.

8.2.3.5 The Output Post Processes.

The post processes are selectable and common to all of the processing groups. The post processes are located between the selectable operator modules and the PE output. A thresholder, an image inverter, and an image lateral shifter are to be implemented. Figure 8.14 is a block diagram of the module. Each sub-module takes one pixel as an input and produces a single pixel output for every pixel rate clock pulse.

The Thresholder. The thresholder is one of the sub-modules shown in Figure 8.14. The input datum is compared with an 8 bit value (range 0 to 255) that has been pre-programmed via the control unit. If the value is zero then the output datum is not modified, otherwise if the input datum is greater than or equal to the threshold the output datum is set to 255, and if it is below the threshold

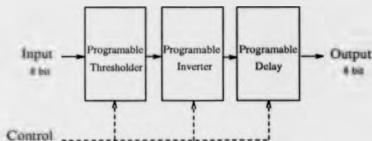


Figure 8.14: The Output Sub-Processors.

the output is set to 0.

The Image Inverter. If inversion is requested via the 1 bit control input then the 8 bit datum is logically inverted.

The Programmable Delay. Under the control of a signed 5 bit number the image datum can be delayed so as to shift the image horizontally to the right (positive control number) or to the left (negative control number). The shift is limited to 16 steps in either direction and for this initial design the host computer is designated to deal with data shifted outside the frame boundaries.

8.2.3.6 The Control Unit.

Figure 8.15 shows a diagram of the unit annotated with the input and output signals. The unit is programmed by a serial stream of data applied to the "program in" input.

Programming information is sent in fixed length blocks headed by a unique PE address that is recognised by the individual PE control unit. Communications are synchronised by a "communications clock", and a "start of program frame" signal indicates that the current received word is a PE address. Data is coded in pure binary. More robust codes such as "Non Return to Zero" [30] are not thought necessary in the first PE design as systems are likely to be constructed on compact, single circuit boards, and the data rate can be kept low. The "program" and "start of program frame"

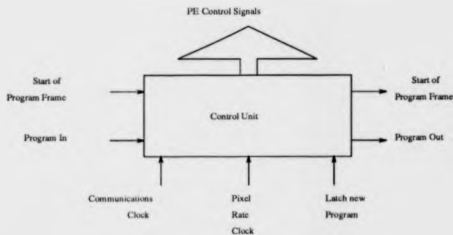


Figure 8.15: The Control Unit.

signals are retransmitted from each unit to avoid line loading problems, to enable automatic PE address generation as described in Section 8.4.1, and to enable communication from the PEs to the system control unit in future systems. A "pixel rate clock" is also received by the unit. The "latch new program" signal is derived from a tap in the PEs synchronisation delay line. Initially new program data is stored in holding registers until the current image frame has been completely processed. The PE control signals that form a frame of communication data are listed Table 8.3.

The PE Control Data Frame. This frame of data is sent from the host computer to the PE via the serial communication link. Extra information for a dual square-hexagonal PE is included to allow for future expansion. Table 8.3 shows a frame of control data to program, fully, one 3x3 local area PE in the pipeline. The first of the 23 entries is the PE address. The number of bits required by the control parameter is given followed by comments on the nature of the parameter. The word width for the address is 8 bits giving an address range from 0 to 255. The next group of control words, items 2 to 4, describe the images that are to be processed. Item 2 sets the length of the delay lines to suit the image resolution, item 3 sets square or hexagonal processing, and item 4 sets up the ALU

to combine two images.

Item 5 contains the four arrays of convolution coefficients. The coefficients are in the range ± 32 so a signed 6 bit number is required to specify them. This example is for a 3×3 local area and so in total 4x9 six bit words are required to specify the arrays. Two coefficient arrays are required for square system processing and four for hexagonal system processing.

Item 6 is decoded to connect the hardware modules for the operation selected. The choice is from convolution, edge detection, sort and select, or the parallel logical operator. Item 7 selects which of the pixel values graded during a sort and select operation is to be put on to the output stream. The next set of items, 8 to 16, set up the parallel logical processor operations. The don't care and gate G1 input invert masks operate on each member of the ten parallel local areas, and so ten 9 bit words are required. Items 10 to 13 control the gate G1 outputs and the gate G2 and G3 inputs. Ten one bit control words are required. Items 14 to 16 control the gate G2, G3 and G4 outputs, each item is a one bit word.

Item 17 conveys the scaling factor that is used in convolution and edge detection operators. A positive number indicates a division and a negative number a multiplication. Item 18 contains a threshold level that can be set in the range 0 to 255. Item 19 requests inversion of the brightness values, and this results in a negative image. Item 20 requests a delay of up to 16 pixels in the output stream, and is used to translate the image to the left or right.

The final item, number 21, contains the values for an output look up table. It contains 256 eight bit words. In total the frame length is 2538 bits if the look up table module is included, and 490 bits if it is rejected. The look up table will be used in few applications and is therefore rejected in this initial PE model.

No.	Name	Word Width	Comment
1	PE address	8 bit	Address range 0 to 255
2	Image resolution eg 833x625	8 bit	To set up line delay taps
3	Square - hexagonal	1 bit	0-square, 1-hexagonal
4	ALU	1 bit	0-no addition, 1-addition
5	Four arrays of convolution coefficients	4 x 9 x 6 bit	9 signed 6 bit numbers per array. 2 arrays for square and 4 for hexagonal, but 4 arrays always transmitted
6	Select Operator 1-2-3 type	2 bit	00 - convolution, 01 - edge detection, 10 - sort and select, 11 - parallel logical operator
7	Select max, min, mean etc after sort	4 bit	Select output after sorting operation. Decimal 1 selects min and decimal 9 max.
8	Set up the parallel logical processing operations Don't care condition masks	10 x 9 bit	10 parallel operators, 0-no operation, 1-don't care
9	Invert gate G1 inputs	10 x 9 bit	Binary pattern, 0-non invert, 1-invert
10	Gate G1 outputs connected to gate G2	10 x 1 bit	0-unconnected 1-connected
11	Gate G1 outputs connected to gate G3	10 x 1 bit	0-unconnected 1-connected
12	Invert gate G2 inputs	10 x 1 bit	Binary pattern, 0-non invert, 1-invert
13	Invert gate G3 inputs	10 x 1 bit	Binary pattern, 0-non invert, 1-invert
14	Invert gate G2 output	1 bit	0-non invert, 1-invert
15	Invert gate G3 output	1 bit	0-non invert, 1-invert
16	XOR gate G4 output with centre pixel	1 bit	0-centre pixel not used, 1-XOR
17	Scaling	6 bit	Range -32 to +31
18	Threshold level	8 bit	Decimal range 0 to 255
19	Image invert	1 bit	0-non invert, 1-invert
20	Programmable delay	5 bit	Signed number, positive - shift right, negative - shift left
21	Look up table	256x8 bit	Not implemented

Table 8.3: Control Signals for a PE Capable of Processing Square and Hexagonally Sampled Images. The Word Widths Given are for a 3x3 Pixel Local Area.

8.3 Possible Options and Modifications That Enhance the Basic PE.

8.3.1 The Processing of Hexagonally Sampled Images.

8.3.1.1 Pipeline System Modification.

If the digitised signal from a non-interlaced, analogue TV camera is to be the input device to the pipeline, a few system modifications are required to enable hexagonal digitisation of the signal. The signal must be circularly band limited at the same frequency as for an equal resolution square system, and as with square sampling, the vertical sampling point spacing is set to an integral number of scan lines, say 625. The horizontal sampling spacing is increased by a factor of $2/\sqrt{3}$, and the first sampling point on alternate lines delayed by half a point spacing. By definition only even numbered scan lines are delayed and the first image line is numbered one. The number of points per scan line is reduced by the $2/\sqrt{3}$ factor, giving typical image sizes of 721×625 or 443×512 pixels in comparison with the equal resolution square sampled image sizes of 833×625 and 512×512 . With hexagonal sampling the data rate is reduced by the $2/\sqrt{3}$ factor from $13.0 \text{ MBytes.s}^{-1}$ to $11.3 \text{ MBytes.s}^{-1}$.

The pipeline architecture is a synchronous system, data input must be synchronised with the clock that pulses the line of processors which operate on the stream of data as it passes through. Fewer modifications to the PE result if the input data is arranged to synchronise with the PE as designed for square sampling. This is achieved if an A to D converter that converts in half the hexagonal sampling point spacing time is used. The converter samples at times so as to effect hexagonal sampling, but data is read at times to effect a rectangular array. This is illustrated in Figure 8.16. Simple circuitry is used to re-synchronise the pipeline output data for display on a TV monitor. For re-circulating pipelines the only system modifications are to the initial image frame grabbing module. Data can be processed by the pipeline at the designed $13.0 \text{ MBytes.s}^{-1}$ rate and so hexagonally sampled images can be processed in 13.4% less time than equivalent square images.

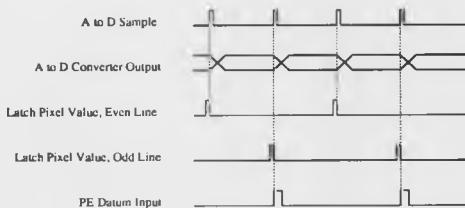


Figure 8.16: A to D Converter Sample and Data Read Timing.

8.3.1.2 PE Modification.

Changes to the PE architecture are minimal, extra taps are added to the line delays to reflect the reduced number of pixels per line. This will add another bit to the associated control word and increase the control unit demultiplexing circuit. Importantly, the length of the line delays does not need increasing. The operator processor must also be modified. For square sampled data, some operations performed by the processor require the convolution of the nine image pixels comprising the local area with one or more 3×3 arrays of constants stored within the processor. The equivalent for hexagonally sampled data requires convolution with a seven element array, i.e. a central element together with six surrounding elements. With the above system modifications for hexagonal data, the position of the central pixel with respect to the six neighbours changes within the grid of nine input pixels on alternate scan lines. This is illustrated in Figure 8.17.

It is necessary to store an extra set of convolution coefficient arrays within the PE's operator processor and to toggle between sets on alternate lines. This will require the line synchronisation signals to be detected and the extra control signal to be processed by the control unit. The two unwanted elements in each array are set to zero. The convolution coefficient magnitude range will

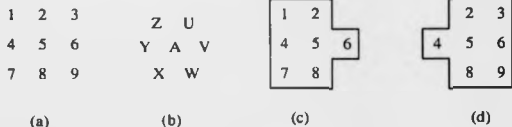


Figure 8.17: The Relationship Between the 9 Input Pixels and the 6 Hexagonal Nearest Neighbours on Alternate Scan Lines. (a) The 9 Input Pixels, (b) Logical Central Pixel A and 6 Nearest Neighbours, (c) Physical Set of 7 Pixels, Scan Line 1, (d) Physical Set of 7 Pixels, Scan Line 2.

be identical to the square range as will the scaling capability provided. In practice the amount of scaling will be less as fewer coefficients are employed.

For the processing of the hexagonal edge detector developed in Section 4.4.1.3 changes will need to be made to the square system edge detection hardware module presented in Section 8.2.3.4. The magnitude equations developed in Section 4.4.1.3 are, for the square system Equation 4.4, and for the hexagonal system Equation 4.9. These equations are reproduced here.

The square system edge magnitude, m_S is:-

$$m_S = (t_v^2 + t_h^2)^{1/2} \quad (8.4)$$

The hexagonal system edge magnitude, m_H is:-

$$m_H = [3(t_1^2 + t_2^2 - t_1 t_2)]^{1/2} \quad (8.5)$$

Where t_v , t_h , t_1 , and t_2 are template responses. Only two template responses are required for the magnitude calculation in each system. For the square system the templates detect orthogonal vectors, whereas for the hexagonal system the vectors are at an angle of 60° to each other, and the resulting magnitude equation is more complicated. A new hexagonal system magnitude approximation

circuit will need to be designed based on the four region approximation algorithm presented in Section 8.2.3.4, but which operates on vectors at 60° instead of 90° to each other. This hexagonal system circuit will be connected in parallel with the square system circuit shown in Figure 8.8.

For the grey-level morphological and median filter operator the two unwanted data bytes, from the square group of nine, will need to be removed from the routine that sorts the group of 7 neighbours. These two bytes change position on alternate lines but are reset to zero in the multiplier array module. If the rank order value is increased by two before transmission to the PE then the presence of these two zero values will be ignored.

No modification to the parallel logical operator is required as the "don't care" masks can be used to reduce the set of input variables to 7 per operator. However, some switching will be required on alternate lines to ensure the correct set of 7.

The post-processing sub-modules all operate on only one image byte and therefore require no modification for hexagonally sampled image processing.

8.3.1.3 Comment.

With a pipeline processor, the processing time for a real-time video image will be unaltered for a particular operation whether square or hexagonal digitisation is employed. One image is processed in one frame time, although a latency delay period is introduced by the string of PEs in the pipeline. Even so there are still advantages for hexagonal digitisation in pipelined systems despite the requirement of some extra control information. The line delays can be reduced in length by 13.4% and the PE master clock can be reduced by the same factor. The shorter line delays reduce process latency and the size of the circuit. In a re-circulating pipeline system hexagonally sampled images will be able to be processed in 13.4% less time than square sampled images. As the local image area contains only seven elements for a hexagonally sampled image many of the processing modules would be simpler than for a square system PE. For example only seven multipliers would

be required as opposed to nine in each multiplier array module. The edge magnitude calculation module is the only module in the hexagonal system PE that is likely to be more complicated than the corresponding module in the square system PE. The small advantages with a hexagonal system PE will not greatly affect the final cost of a VLSI device, and the argument has to be that, as square system processing is so well established, a square only or a dual standard device looks more attractive to fabricate than a hexagonal only device. The dual standard device will enable easy comparison of the quality of image processes implemented with the two sampling systems.

8.3.2 Alternative Circuits to Reduce the Number of Multipliers in the Multiplier Arrays.

8.3.2.1 Separable Operators.

In the proposed basic PE circuit there are a total of eighteen 8x6 bit multipliers in multiplier arrays (A) and (B). These arrays and the line delays are likely to be amongst the modules requiring the largest area of silicon in any VLSI implementation. The following scheme based on the separability of some of the convolution and edge detection operators used with square sampled images can reduce the number of multipliers to six of size 8x6 bit and six of size 11x6 bit. The addition tree of Figure 8.7 is also simplified. The reduction in the number of multipliers is at the expense of the word width of the line delays which must be increased from 8 to 11 bits to maintain accuracy.

The alternative hardware is only suitable for image processing operations requiring a convolution stage. It is unable to perform the array multiplications required by many of the sort and select operations in which each member of the local image area is multiplied by a separate coefficient value before the selection process. A possible compromise would be to construct multiplier array (B) with this new circuit, as it is only used in convolution calculations, and to keep the original, more general purpose circuit, for multiplier array (A).

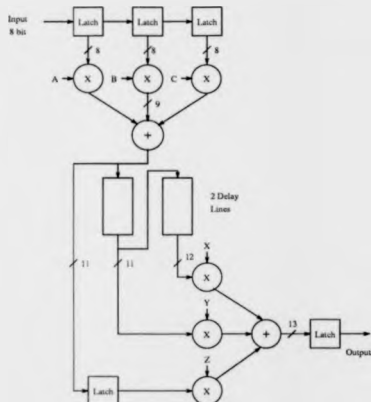


Figure 8.18: The Alternative Convolution Module Circuit. The Latches are Clocked at the Pixel Rate. Horizontal coefficient array {A B C}, vertical coefficient array {X Y Z}.

Separable operators are described and illustrated in Section 4.3. For the PE, instead of convolving the image with a 3×3 array of coefficients, it is first convolved with a 3×1 horizontal array, and then by a 1×3 vertical array. If the operator is separable, and the coefficient values correctly chosen, then the overall result will be the same as a single pass of the original 3×3 array. Figure 8.18 shows a suitable circuit for the module. The input data stream is clocked through a series of three 1x8 bit latches. The values in the latches are multiplied by the horizontal "row" coefficient values and added together. The result is then passed to a pair of TV line length delays with an 11 bit word width. These delays enable a "column" of partially processed results to be assembled with which

the 1x3 vertical array can be multiplied. The outputs of the "vertical" group of 11x6 bit multipliers are then added and the result latched before output.

The circuit will only work with separable operators. Not all square system, and no hexagonal system operators are separable. This is a severe limitation. It is unlikely that the trade off between a reduced chip area for the multipliers, and an increased area for the wider delay lines would give a worthwhile advantage.

8.3.2.2 Simplified Multiplier Array (B).

Multiplier array (B) is only used by edge detection operators. If only the Sobel detector, the equivalent hexagonal system detector, and the Robert's detector are required, then the complete set of convolution coefficients used by this multiplier array is {2, 1, 0, -1, -2}. These can be applied to the local image area by adder/subtractor circuits rather than by the 8x6 bit multipliers in the original circuit of Figure 8.6.

8.3.3 5x5 Pixel Local Image Areas.

8.3.3.1 In General.

The size of the local image area assembled within the PE can be increased by providing more line delays. Even sized arrays can be realised but it is not obvious which processed pixel position should form the output stream. For example in the following 4x4 array either F, G, J, or K could be chosen for output.

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

With an odd sized array there is one central pixel. Even sized operator arrays can easily be embedded within the next largest odd sized array. The number of line delays required is one less than the vertical array size, however, the multiplier arrays and much of the selectable operator modules will require an increase in component numbers proportional to the increase in local area size.

8.3.3.2 A 5x5 Local Image Area for Parallel Binary Operations Within a Normal 3x3 PE.

With the proposed circuit of Figure 8.13 for parallel binary operations, multiplier array (A) is not necessarily required. It simply provides a global method of masking bits in the local area. This masking could be performed in each of the current parallel operators. The image information contains only binary values, whereas the line delays are 8 bits wide. It would be possible to assemble a 5x5 binary local area within a 3x3 PE by circulating the image data through different bit levels in the current line delays. This could be achieved by simple switching. Apart from this switching, the circuits in Figure 8.13 for applying the templates would have to be increased to accommodate 25 as opposed to 9 binary inputs. The TV synchronisation signals would also need to be further delayed in unused bit levels of the TV line delays.

8.4 Image Processing Pipeline Systems Using the Basic PE.

8.4.1 A Flexible Programmable Video Rate Pipeline.

Figure 8.19 shows a typical video rate real-time programmable pipeline. The high level processor communicates with the pipeline control which instructs the PEs to perform the required processing functions. In a controlled environment, such as in many industrial inspection applications, the PEs will only need programming when the item type is changed and on initial switch on. On the other hand in a naturally light, constantly changing scene, some re-programming may be required every

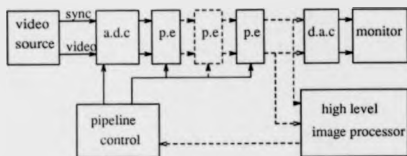


Figure 8.19: A Programmable Video Rate Pipeline.

video frame.

It is envisaged that individual PEs will be encapsulated in small packages, and that many such devices could be installed on single circuit boards together with the ADC, DAC and signal conditioning circuits.

8.4.2 Topological Interconnection Possibilities.

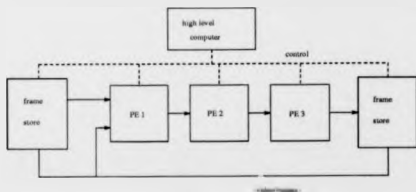


Figure 8.20: A Programmable Re-circulating Pipeline.

The PE as described can be used in re-circulating pipeline architectures as shown in Figure 8.20.

The control unit is not shown separately from the host here. PE1 can be programmed for any function and can combine images from the two frame stores. The other PEs are hardware identical

to PE1 and are also programmable.

If the PE delay is kept equal, independent of the programmed function, then several interconnection alternatives exist. Small delays introduced to shift images spatially would be acceptable.

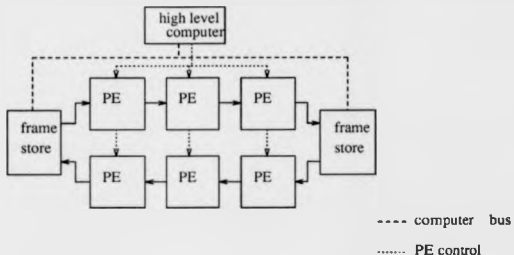


Figure 8.21: A Multi-pipeline Re-circulating System.

Re-circulating pipeline systems employing many frame stores, each interconnected by pipelines of PEs, could be constructed, as shown in Figure 8.21. Links could be made between pipelines to the second PE input ports.

Video rate real-time systems could be constructed with several parallel pipelines employing many cross interconnections from PE outputs in one line, to the second video input of PEs in other lines. A simple example is shown in Figure 8.22. Here the top pipeline produces an edge detected and thinned line image which is then superimposed on the original grey level image that has been equally delayed by the lower pipeline of PEs.

8.4.3 Towards Adaptive Control.

With an adaptive control mechanism various parameters produced by a system are measured and new coefficients or operator configurations calculated to optimise the system performance. No

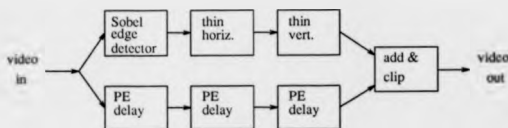


Figure 8.22: A Multi-pipeline Arrangement to Produce an Edge Overlay on an Image.

research has been performed to identify adaptive control algorithms for any specific applications of this pipeline architecture, but the features of the architecture that may allow adaptive control are outlined below.

In an industrial inspection environment adaptive control algorithms may be simple. For example the type of object on a conveyor may change, the new shape could be recognised by the host computer, and the pipeline re-programmed to perform optimum tests on the new object. For a naturally light, out-of-doors scene, many parameters may need to be measured at points along the pipeline, in co-processors such as histogram sub-modules, or by the host computer. Separate pipelines may be constructed in parallel with the main pipeline to facilitate control parameter measurements. Individual PEs could also report results via the control communications ring, but these would need to be identified at the PE design stage. The control algorithm would be able to change coefficient values or the order or type of operation being performed in each PE. The host computer would control the point in time at which re-programming would occur, together with the detection of the frame synchronisation pulse in each PE.

Such out of doors, general scenes may require a different set of processing operations to be applied for different spatial regions of the image. For instance part of a scene may be in shadow, but detail within this region may be required. The best solution, as realised by a camera reported by Ginosar et al. [61], would be to control locally the sensitivity of the sensor elements covering this region. Another solution would be for the adaptive control algorithm of a multi-pipeline

architecture to program different algorithms in each pipeline, that would each process the entire image, but optimally for the different regions. The host computer together with the frame store address counters would then select particular regions from each pipe to assemble the output image.

8.5 Conclusions.

The design of a basic PE that will process operations from four different image processing groups has been discussed. A basic PE has been proposed as it is easier to understand the various processing modules involved, and many system and circuit problems can be identified before the design is scaled up to provide more features and a larger local image area.

The basic PE will process square sampled images with spatial resolutions of up to 833x625 pixels. A 3x3 local area is assembled within the PE. The four image processing groups implemented are a convolution group, an edge detection and enhancement group, a sort and select group, and a parallel binary operation group. These groups contain operators that have been shown in the image processing chapters of this thesis to be useful in industrial and controlled lighting environments. The four groups require different hardware for their implementation.

The single PE design contains modules that enable image operations from each of the four groups to be processed completely within just one PE. The PE is configured by a host computer for the operation it is to perform, and can be reconfigured during the image interframe time. A typical pipeline may contain a first PE configured as a low pass Gaussian filter for noise reduction, a second stage PE configured for edge detection that outputs thick binary lines where an edge exists, and a series of say three PE's, each providing a line thinning function of single pixel removal from either side of the line. If the threshold value set in the edge detection PE is well chosen, then the pipeline output will be an edge map containing single pixel thick lines.

Functional block diagrams of the processing modules within the PE have been presented that it is

expected will lead to a minimum hardware implementation. The more complicated circuit modules have been simulated by programs written in a high level language and found to be functionally correct. The complete PE has been simulated, and the results presented in the next chapter. This simulation provided information that was used to select the internal word widths of the PE. Minimising these word widths, while maintaining a sufficient accuracy for the image processing operators helps to minimise the hardware requirements of the PE.

A modification to the basic PE to enable hexagonally sampled images to be processed has been presented. Few extra circuits were found to be required if a dual standard square-hexagonal PE were to be produced. The extra circuits are required to store two extra arrays of convolution coefficients for use on alternate TV scan lines, to provide switching of taps in the delay lines and between arrays of convolution coefficients, and to provide a new magnitude calculation circuit in the edge detection module. In a video rate system, hexagonally sampled images will be processed at a 13.4% lower pixel clock rate, and the processor latency will be reduced by the same factor in comparison to square sampled images. In a recirculating pipeline the hexagonally sampled images will be processed with a time saving of 13.4%.

A modification to the basic PE to enable a 5x5 local area for parallel binary operations to be performed within the basic 3x3 PE has been presented. Its implementation would allow Chin's [27] optical character thinning operations to be performed.

Some suggestions to reduce the number of multipliers in the multiplier arrays have been made. These should only be considered if it is found to be too difficult to assemble all 18 multipliers on the one VLSI device as they impose restrictions on the image processes that can be performed.

Various pipeline architectures that can be constructed using the proposed PE have been discussed. These included video rate and recirculating pipelines, and also a branching and merging multiple pipeline that it will be possible to construct if the PE latency is kept independent of the selected PE operation.

Chapter 9

The Simulation of the Pipeline Processor Element.

9.1 Introduction.

A pipeline system based on a proposed processor element (PE) has been presented in Chapter 8. In this chapter the results from a simulated pipeline of simulated PEs are presented. The simulation, within its limitations, is used to evaluate the PE specification with respect to the range of operators implemented, the scaling of partial results, and inter-device communication. Comparisons are made between similar function operators that have been realised with different PE sub-modules so that any redundant sub-modules can be identified. Sub-modules for hexagonally sampled image processing have been developed with a minimum of changes from the corresponding "square" ones. This attempt here at maximum compatibility will result in more integrated and hence smaller hardware circuit modules. The results provide a further comparison between the quality of similar processes performed on hexagonal and square sampled images.

9.2 The Simulation of Complete Pipelines.

Many computer operating systems allow the standard output of one program to be piped into the standard input of the next. This technique is explained by Kernighan and Ritchie [94]. A PE simulation program was written and assembled into pipelines of PEs and support programs by operating system commands.

The pipeline support programs include data conversion, data preparation and additive noise generator programs. The pipeline input data conversion program converts data from various image types to decimal number strings, which is a useful format to use during the development of the PE simulation program, as it allows the pipeline to be broken and partial results to be displayed on the user terminal. The output data conversion program converts the processed data, which can be in an array of any size, to a 512x512 byte array for display.

The simulation does not operate at video rate, but on stored images. Two data preparation programs are usually required in the pipeline. The first is located immediately before the first PE and attaches a number of zeros to the borders of the image. These padding pixels are normally associated with real-time video images, and, as discussed in Section 8.2.3.2, provide a separation between images so that processing can continue without cross image interference. The second data preparation program removes the border padding pixels before the output image is displayed or stored.

The additive noise generator pipeline element adds randomly generated noise values, which as a set have a mean value of zero and a predetermined variance, to the image pixel values before they enter the PE section. It was used on computer generated images rather than on real world images in order to create more realistic images on which to test subsequent processes.

9.3 The Processor Element Simulation.

The simulation verifies the design of the PE presented in Chapter 8. A set of low level, local image processes, that have been identified in Chapter 4 as being commonly used in controlled lighting applications, are tested on actual and simulated industrial images. The processes within the PE simulation programs are modular, with each module representing a block diagram component as presented in the many figures in Chapter 8. Most of the PE modules have been simulated exactly as in the specification, with the exception of those listed below. The simulation is only functional, and runs on a conventional SISD computer. It was not possible to model each circuit block as it might be designed for hardware implementation. At a later design stage, a logic level simulation package such as Hilo [59] could be used to simulate the modules at a lower level. Its execution speed would, however, be too slow for it to be used for simulations of complete PEs or pipelines. This high level simulation would be able to provide and analyse test data from the later logic level simulations.

The simulation does not incorporate all of the characteristics of the final design. The transmission of video synchronisation signals through the line delays, and the input video stream ALU are omitted. The ALU function is provided by a service program. The simulation will handle local area sizes of odd dimension from 3x3 upwards, however the local area for parallel logical operators must not exceed 5x5.

Line synchronisation of data was necessary within the PE simulation as near neighbour pixels on various lines have to be assembled into local areas. The control signals provide information on the line lengths of the stored image, but a separate data preparation program, at the front end of the pipeline, extends the line lengths with padding pixels as described in Section 9.2. The number of padding pixels is a function of the image line length. For the simulation it was simpler to separate the image size and square - hexagonal flag from the other control signals and store them as a header in the image file. These fixed length headers are then read before any image data by

every program in the pipeline. The headers remove the necessity of sending a control data frame to the data preparation programs, and of providing separate control data frames for differing sized images. The data padding program generates further control signals that are added to the image header, indicating the expanded image size. The PE uses these for synchronisation.

The local areas were limited to odd sizes so that the central pixel could be unambiguously defined. Operators requiring even sized areas were embedded within the next largest odd area. The local area size for the parallel logical operators was limited to reduce the quantity of control information.

The simulation program can be split into four sections:- data input, main processes, post processes, and data output.

9.3.1 Data Input.

The Control Data Frame is read by the simulation program from a separate file before the image data is streamed in, and thus, as in the theoretical design, can only re-program the PE at the start of a new frame. For each process which the program has to perform, a separate control file is required. The format of the control frame is as defined in Section 8.2.3.6 apart from the exemptions listed in Section 9.3. Here the file contains decimal number strings.

The Line Delays are modelled by a one dimensional dynamically allocated buffer, the size of which is calculated from the image size and the local area size data contained in the control data frame. A pointer indicates the next available location within the buffer and other variables indicate the line length and manage the preloading of the image into the buffer and the flushing of the buffer after all the image pixels have been processed.

9.3.2 The Main Processes.

The program contains several subroutines, each of which performs a function identical to one performed by the corresponding modules defined in the specification given in Chapter 8. The operation requested in the control file calls the subroutines in the correct order to carry out this operation on the image pixel that is currently at the centre of the local area. The subroutines include:-

- *Convolution*. The local area is matrix convoluted in turn with each of the four coefficient arrays specified in the control data frame. The routine can be used with both square and hexagonally sampled data. The input parameters are the local area pixel values and the coefficient values. These are all integer. The four output values are all integers. This subroutine calls the distributed scaling subroutines Scale.1 and Scale.2.
- *Array multiplication*. The local area is matrix multiplied with the first and third of the coefficient arrays to produce two new arrays. The routine can be used with square and hexagonally sampled data. The input parameters are the local area pixel values and the coefficient values. These are all integer. The output arrays are the same size as the input arrays and the values are all integers. This subroutine calls the distributed scaling subroutine Scale.1.
- *Convolution - selection, square sampled data*. This routine selects the convolution sum obtained with the first control file coefficient array for output. The four input values and the output value are all integers.
- *Convolution - selection, hexagonally sampled data*. The convolution sum obtained with the first coefficient array for output is selected when the central pixel of the local area is on an odd line, whereas the convolution sum obtained with the third coefficient array for output is chosen when the central pixel is on an even line. The four input values and the output value

are all integers.

- *Edge detection, square sampled data.* This routine calculates the magnitude of the edge gradient for the pixel at centre of the local area from the responses of the convolutions with the first and second coefficient arrays. The four input values and the output value are all integers. This subroutine calls the distributed scaling subroutine Scale.3.
- *Edge detection, hexagonally sampled data.* The magnitude of the edge gradient is calculated for the local area from the response of the convolutions with the first and second coefficient arrays when the central pixel of the local area is on an odd line, and with the third and fourth arrays when the central pixel of the local area is on an even line. The four input values and the output value are all integers. This subroutine calls the distributed scaling subroutine Scale.3.
- *Sort, square sampled data.* The local area values are ordered by magnitude, and the one requested in the control file is selected for output. Typically the minimum, mean, or maximum value is requested. The input parameters are the local area pixel values and the selection parameter from the control data frame. These are all integer. The output value is an integer.
- *Sort, hexagonally sampled data.* The same operation as the square sort routine is carried out for the small hexagonal local areas. A different subset of the local area pixels are sorted on alternate scan lines. The input parameters are the local area pixel values and the selection parameter from the control data frame. These are all integer. The output value is an integer.
- *Parallel logical operation, square sampled data.* This provides the ten parallel operations on the least significant bits of the data in the local area. The image data can thus be coded with logical one represented by 255 or 1, and logical zero by 0. The logical operations performed are those identified in Section 8.2.2.2. The simulation is limited to local area sizes of up to 5x5. Such a limit is appropriate for hardware implementation in order to restrict the size of

the operators. As a change to the device specification as presented in Table 8.3, the input don't care and input invert control information is entered in the control data frame, and stored, as 10 single integers for each of the 10 parallel operators. Each integer codes a binary string comprising one bit for each pixel in the local area. The remainder of the control information is entered as single bit variables. The input parameters are the local area pixel values and the parallel logical operator control parameters from the control data frame. These are all integer. The output value is an integer that can take the value 0 or 255.

- *Parallel logical operation, hexagonally sampled data* Ten parallel operations on hexagonal local areas are provided. The subroutine is similar in operation to the square routine, except that here the local area size is limited to 7 or 19 elements. As an alternative to switching between two sets of control data for use on odd and even lines, as is done with the convolution operators, switches are used to assemble the correct limited local areas in each case, and to operate on these with one control set. The input parameters are the local area pixel values and the parallel logical operator control parameters from the control data frame. These are all integer. The output value is an integer that can take the value 0 or 255.

9.3.2.1 The Scaling Routines.

Three routines limit the magnitude of the results obtained during these processes. They simulate the distributed scaling scheme described in Sections 8.2.3.3 and 8.2.3.4 which was introduced to limit the hardware size of the PE by reducing the word widths of the various operator stages under program control. The routines divide the integer value at a particular point by a predetermined control file variable, and then limit the word width to a value set within the program. Values that exceed this limit are clipped. This arrangement enabled the effects of limiting to various word widths to be observed.

- *Scale.1.* This is applied at the stage where the local area data are multiplied with the convolution coefficients to limit the individual results. The scaling value chosen reflects the maximum result expected for a particular problem. *Scale.1* is called by subroutine:Convolve and subroutine:Array multiplication. The input parameters are the value to be scaled and the scaling value from the control data frame. These are both integers. The output value is an integer.
- *Scale.2.* This is applied at the stage where the convolution sums are formed to limit the results. Again, the scaling value chosen reflects the maximum value expected at this point. *Scale.2* is called by subroutine:Convolve. The input parameters are the value to be scaled and the scaling value from the control data frame. These are both integers. The output value is an integer.
- *Scale.3.* This is applied during the edge detection operations to limit the word widths during the magnitude calculation phase. *Scale.3* is called by the two edge detection subroutines. The input parameters are the value to be scaled and the scaling value from the control data frame. These are both integers. The output value is an integer.

A second version of the simulation program has been developed with the scaling routines removed so that degradation caused to the operator quality by the scaling can be observed.

9.3.3 The Post Processes.

In the hardware PE design, the data are passed through a pipeline of simple post processes after the main processes have been applied. For each post process, the output is a function of the central pixel and the relevant control data frame values. An equivalent line of processes has been simulated here.

- *Scale factor.* This process is only included in the version of the program without distributed scaling. It reduces the output to a range suitable for display.
- *Threshold.* The central pixel value is compared to a value entered via the control file. If the value is above the threshold, logic 1 is set, otherwise logic 0 is set.
- *Limit.* The output is limited to be within the range 0 to 255. Values outside this range are clipped to the limiting value.
- *Image inversion.* If the control file flag is set the negative of the image is calculated. The new pixel value is equal to 255 minus the old value.

9.3.4 Data Output.

After post processing, the data is put onto the standard output stream. A second channel is used to output messages to the terminal that might form the basis of messages returned to the hardware control unit via the communication ring. Examples of such messages are:-

- *Logical operator activity.* The thinning routines employed are iterative. A counter is incremented every time a pixel value is modified and the counter value displayed after the frame has finished processing. A zero count value indicates that no further thinning is possible.
- *Pixel value outside limit.* This indicates an inappropriate scaling value has been chosen. The message conveys which scaling routine caused the error. However if an incorrect scaling value is chosen, the number of errors, and hence the volume of communication data tends to become large.

9.4 The Simulation Results.

The results show the ability of the pipeline of PEs to process a stream of image data from a stored file, and that sufficient functional blocks have been incorporated in the PE design to achieve the four general algorithmic groups of:-

- Convolution operators.
- Edge detection operators.
- Sort and select operators.
- Parallel logical operators

The following four subsections each contain the results obtained from one of the above four groups of operators. The results indicate the ability of the PE to perform the operation. Within each group similar operators are compared and the numerical differences in their performance analysed. If the differences are insignificant then the PE design can be simplified and its size and cost reduced. In particular if the local area size, or if the range of the convolution coefficient values can be reduced, then the size of the hardware multiplication devices can also be reduced. Distributed scaling options are examined in detail so that the minimum acceptable word width can be identified. Minimising the word width will minimise the size of the hardware modules throughout the fabricated PE.

Some of the results presented require the pipelining of several PEs. This enables simulated pipeline systems to be tested and the usefulness of such systems for controlled illumination and industrial applications to be evaluated.

9.4.1 Convolution Operators.

These operators are used here to construct spatial filters. The design of such filters was discussed in Section 4.4.1.2. Firstly two square system Gaussian low pass filters are compared. The first G_1

has a smaller range of coefficient values than the second G_2 , and so will be easier to implement in hardware. The filters are then compared when distributed scaling is applied, and choices of the optimum filter and the optimum word width made. A high pass point enhancement filter is then investigated to ascertain the optimum word width. Finally filtered hexagonal system images are evaluated both with and without distributed scaling.

The convolution operator is specified by a control data frame variable. Three of the main processing routines, as outlined in Section 9.3.2, are used together with two of the scaling routines. The main processes are *convolution* and either *convolution - selection (square)* or *convolution - selection (hexagonal)*. Tests were made without distributed scaling, and with distributed scaling routines *scale.1* and *scale.2* limiting the word widths to both 8 and then 9 bits for comparison. A single coefficient array is required for square sampled images and two coefficient arrays for hexagonally sampled ones.

9.4.1.1 Square Sampled Images.

Low Pass Filters. From the theoretical discussion presented in Section 4.4.1.2 on low pass filters, two square system Gaussian approximation convolution templates have been chosen for comparison.

$$\begin{array}{rcc}
 G_1 = & \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array} & \begin{array}{ccc} 3 & 7 & 3 \\ 7 & 16 & 7 \\ 3 & 7 & 3 \end{array} \\
 & & G_2 =
 \end{array}$$

Both are approximations in that they receive support from a small local area and contain integer only coefficients. Both filters were presented by Davies [37], who states that G_1 is theoretically better for defect detection and G_2 for defect measurement applications, but that in most cases the differences are not apparent. The two filters were first applied to several images in a PE without distributed scaling, then in a PE with scaling to limit the word widths to 8 bits, and then in a PE with distributed scaling to limit the word widths to 9 bits. Figure 9.1 shows an original image and an

image filtered by G_1 with distributed scaling limiting values to 8 bit word widths. In ideal viewing conditions, on a monochrome TV monitor, no differences could be observed between the G_1 and G_2 filtered images with or without distributed scaling. However the human eye is only capable of detecting grey level brightness changes in a 6 bit resolution system, and so further tests were performed to check for numerical differences in the processed image brightness data and for object edge distortion. The sand core test image in Figure 9.1 is too complicated for this task and so a test image was constructed.



Figure 9.1: 64 x 64 image, Unsmoothed and Smoothed by G_1 with Distributed Scaling.

A test image was constructed containing a rectangle with brightness 50 in a 16x16 background area with brightness 0 (system brightness range 0 to 255). The rectangle was oriented so that its edges did not align with the sampling grid. Figure 9.2 shows the resulting low contrast image together with the numerical pixel values in the vicinity of the leftmost corner. The values displayed for this corner are typical of the values distributed throughout the remainder of the image.

Differences between the effects of applying G_1 and G_2 to the images, without distributed scaling can be observed in Figure 9.3. The differences are small, the observer is hard pressed to determine which filter has the largest a/b ratio (width) as defined in Section 4.4.1.2. The numerical differences are limited to changes in one digit.



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	9
0	0	0	0	0	0	0	0	10	50
0	0	0	0	0	0	0	0	39	50
0	0	0	0	0	0	0	0	18	50
0	0	0	0	0	0	0	0	50	50
0	0	0	0	2	45	50	50	50	50
0	0	0	0	26	50	50	50	50	50
0	0	0	6	48	50	50	50	50	50
0	0	0	22	50	50	50	50	50	50
0	0	0	12	41	50	50	50	50	50
0	0	0	0	0	1	20	46	50	50
0	0	0	0	0	0	0	0	3	28
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 9.2: The Low Contrast Test Image and the Brightness Values in the Vicinity of the Leftmost Corner.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	5	11	9	0	0
0	0	0	0	3	17	31	29	0	0
0	0	0	1	10	30	45	45	0	0
0	0	0	5	22	41	49	49	0	0
0	0	0	1	13	34	47	50	50	0
0	0	0	7	25	43	49	50	50	0
0	0	2	15	36	48	50	50	50	0
0	0	3	16	35	46	49	50	50	0
0	0	1	7	18	31	41	47	49	0
0	0	0	4	11	23	34	43	0	0
0	0	0	0	1	5	14	26	0	0
0	0	0	0	0	0	2	7	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 9.3: The Application of G_1 and G_2 to the Image Without Distributed Scaling.

Figure 9.4 shows the effects of applying G_1 when distributed scaling is used to limit the word widths to 8 bits, and applying G_2 when distributed scaling is used to limit the word widths to 9 bits. Limiting the word widths to 9 here allows a distributed scaling value of 8 to be applied by the first scaler: *scale.1*, and of 7 by the second scaler: *scale.2*, giving an output range of 0 to 255. If the word width had been limited to 8 bits, and integer only scaling allowed, then a scaling of 16 would be needed at *scale.1* and of 4 at *scale.2*, resulting in an overall scaling of 64 and an output in the range 0 to 223. Again the numerical differences between the images in Figure 9.3 and Figure 9.4 are limited to changes in one digit, as are the differences if the convolution with G_2 is performed

Conclusions. The PE applied these square system low pass filters correctly. Davies' [37] conclusion that there is little difference between the results of applying the G_1 and G_2 filters is confirmed. G_1 requires coefficients with values between 1 and 4, while G_2 requires coefficients with values between 3 and 16, a larger range. The G_1 coefficients are also all powers of 2 and so all of the multiplications could be performed by shift left operations. Whether the multiplications are achieved by shift operations or by a general purpose multiplier, G_1 requires less hardware to implement. There is no need to implement G_2 , the PE design can therefore be simplified unless the simplification conflicts with the requirements of any of the following operators.

Distributed scaling with a PE that limits the word widths to 8 bits with rounding, produces errors of the same magnitude as the differences between G_1 and G_2 , while such scaling with a PE that limits the word widths to 9 bits, with rounding, produces errors of less magnitude. No edge distortions could be attributed to either filter or either distributed scaling option. The conclusion is made that for the majority of cases only filter G_1 need be used, and that, for 8 bit image data, a PE that rounds the word widths to 8 bits will not introduce any significant errors.

High Pass Filters. Various high pass filters are achievable. Here a point enhancement filter is studied to investigate the effects of distributed scaling and of limiting the internal word widths to 8 and 9 bits. The template G_3 is presented by Gonzalez and Wintz [65].

$$G_3 = \begin{matrix} & -1 & -1 & -1 \\ -1 & 8 & -1 & \\ -1 & -1 & -1 & \end{matrix}$$

The largest coefficient value is 8. If the word width is limited to 8 bits, then distributed scaling of a division by 8 must be applied, by subroutines: *scale.1*, after the coefficient multiplication. The remaining coefficient values are all -1. For a practical situation a multiplication by up to 8 could be applied by *scale.2* to obtain an image with a suitable average brightness level for viewing.

Some resulting pixel brightness values may then be out of range and need clipping by the scaling subroutine. The simulation interprets a negative scale factor to indicate multiplication by the modulus of the factor.



(a)

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 206 72
0 0 0 0 0 0 0 84 65 57
0 0 0 0 0 0 0 98 11 1
0 0 0 0 0 0 114 37 0 0
0 0 0 0 0 7 79 5 0 0
0 0 0 0 130 26 0 0 0
0 0 0 60 171 49 9 0 0
0 0 0 0 0 95 92 34 4
0 0 0 0 0 0 0 117 74
0 0 0 0 0 0 0 0 18
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

```

(c)



(b)

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 216 80
0 0 0 0 0 0 0 96 80 72
0 0 0 0 0 0 0 104 24 16
0 0 0 0 0 128 48 16 16
0 0 0 0 8 88 16 16 16
0 0 0 0 136 40 16 16 16
0 0 0 64 176 64 24 16 16
0 0 0 0 0 112 104 48 16
0 0 0 0 0 0 0 136 88
0 0 0 0 0 0 0 0 24
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

```

(d)

Figure 9.6: (a) The Low Contrast Test Image (b) After High Pass Filtering (c) Numerical Representation at the Leftmost Corner, No Scaling (d) Numerical Representation After Distributed Scaling and a Word Width Limit of 8 Bits.

Figure 9.6 shows (a) the low contrast rectangle image discussed earlier, (b) the resulting high pass filtered image, (c) the numerical representation at the leftmost corner of the high pass filtered image with no distributed scaling applied, and (d) the numerical representation when distributed scaling has been applied. The values displayed for this corner are typical of the values distributed throughout the remainder of the image.



Figure 9.7: Hexagonally Sampled Sand Core Image, Unsmoothed and Smoothed by G_4 with Distributed Scaling.

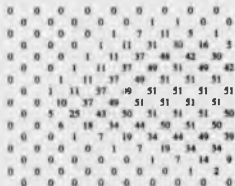
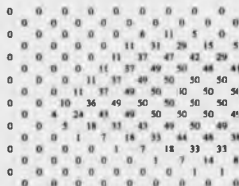


Figure 9.8: The Application of G_4 to the Hexagonally Sampled Low Contrast Rectangle Image, Without and With Distributed Scaling.

majority of cases, the errors introduced by the distributed scaling and the 8 bit word width limit would be acceptable.

9.4.2 Edge Detection Operators.

Tests are performed on edge detectors designed for use on both square and hexagonally sampled images. The design of these operators is described in Section 4.4.1.3. The Sobel detector and the equivalent hexagonal sampling system detector developed as part of this project are compared with each other, both with and without the application of distributed scaling. Distributed scaling routines *scale_1*, *scale_2* and *scale_3* are used. The edge detection operation is selected by a control data

frame variable which sets up a processing path within the PE consisting of the *convolution*, and the *edge detection (square)* or *edge detection (hexagonal)* routines. The *threshold* post process is usually used to produce a binary edge map, but if the threshold control variable is set to zero, an edge enhanced image is output.

9.4.2.1 Square Sampled Images.

To test for edge distortion the low contrast rectangle image is detected both in a PE without distributed scaling, and in a PE with distributed scaling that limits the word widths to 8 bits. In the first test the detector threshold level is set at the highest level that still results in a connected edge outline. This threshold level is independent of whether distributed scaling is applied. The resulting edge map images are also identical. In the second test the detector threshold level has been reduced to a level that is more likely to be used in a practical application. This results in a wider edge, but again the resulting images are identical. One corner of the unprocessed image is displayed in Figure 9.9 together with the processed images. No edge distortion can be attributed to either the edge detection or to the distributed scaling processes. Distortions due to sampling, and in a practical situation, noise, are likely to be larger.

Further tests were performed on simulated edges at angles between 0 and 90 degrees in 2 degree steps to measure edge magnitude accuracy. The original test is presented in Section 4.4.1.3 where the results were used to compare the square and hexagonal system edge detectors. The comparison here is between edge detection without distributed scaling, with distributed scaling which limits the word width to 8 bits, and with distributed scaling which limits the word widths to 9 bits. By setting the PE threshold level to 0, the magnitude of the edge detection is output as an 8 bit integer value without a thresholding process being applied. The edge height is 10 brightness units in each case and the detector width is 2 pixels, so the gradient output should be 5. All the PE simulation models

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 9
0 0 0 0 0 0 0 0 0 9
0 0 0 0 0 0 0 10 50 36
0 0 0 0 0 0 0 0 59 50
0 0 0 0 0 0 0 18 50 50
0 0 0 0 0 2 45 50 50 50
0 0 0 0 0 26 50 50 50 50
0 0 0 0 6 48 50 50 50
0 0 0 0 22 50 50 50 50
0 0 0 0 12 41 50 50 50
0 0 0 0 0 1 20 46 50
0 0 0 0 0 0 0 0 3 28
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

```

(a)

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 * * *
0 0 0 0 0 0 0 * 0 0 0
0 0 0 0 0 0 * 0 0 0
0 0 0 0 * 0 0 0 0 0
0 0 0 0 * 0 0 0 0 0
0 0 0 0 * 0 0 0 0 0
0 0 0 0 * 0 0 0 0 0
0 0 0 0 0 * * 0 0 0
0 0 0 0 0 0 0 0 * *
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

```

(c)

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 * * *
0 0 0 0 0 0 0 * 0 0 0
0 0 0 0 0 * * 0 0 0
0 0 0 0 * 0 0 0 0 0
0 0 0 0 * 0 0 0 0 0
0 0 0 0 * 0 0 0 0 0
0 0 0 0 0 * * 0 0 0
0 0 0 0 0 0 0 * *
0 0 0 0 0 0 0 0 * *
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

```

(d)

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 * * *
0 0 0 0 0 0 0 * * *
0 0 0 0 0 0 * * 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0

```

(e)

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 * * *
0 0 0 0 0 0 * * * *
0 0 0 0 0 0 * * 0 0
0 0 0 0 0 * * 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0
0 0 0 0 * * 0 0 0 0

```

(b)

Figure 9.9: (a) The Low Contrast Rectangle Image. (b) Edge Image with High Threshold Level (c) Edge Image with High Threshold Level and Distributed Scaling (d) Edge Image with Low Threshold Level (e) Edge Image with Low Threshold Level and Distributed Scaling.

produce the correct integer output for each orientation of this edge. However by modifying the simulation programs to output double precision floating point magnitude values for each edge, a more exact measure of the errors introduced by the distributed scaling can be obtained. These results are plotted in Figure 9.10.

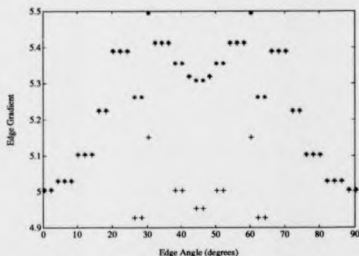


Figure 9.10: Square Operator Edge Gradient Response for Edges, Gradient = 5.0, Oriented $0 \Rightarrow 90^\circ$. Key: * Without Distributed Scaling, + With Distributed Scaling and a Word Width of 8 Bits.

Continuous curves cannot be drawn in this figure as, apart from at the output, the PE's internal integer-only arithmetic has introduced discontinuities in the data. The curve in Figure 5.4 was obtained by a program which used double precision arithmetic throughout. However, without distributed scaling, the points plotted in Figure 9.10 correspond to the equivalent points on the curve of Figure 5.4 closely, the error magnitudes here being up to 0.1 of a brightness unit larger. This extra error is also attributed to the PE's integer only arithmetic. In Figure 9.10 the points plotted when the distributed scaling subroutine limits the word widths to 8 bits, exhibit lower magnitude errors in comparison to the points without distributed scaling. The discrepancy is up to -0.4 brightness units. In this example the word width limiting errors compensate for the detector errors, but in general they should be considered as corrupting the image by a magnitude similar to

that caused by the detector. The points plotted when the distributed scaling subroutine limits the word widths to 9 bits are identical to the points plotted with no distributed scaling. A 9 bit word width limit is therefore preferable, but an 8 bit limit will be less expensive to implement. The additional errors introduced by the 8 bit scheme are small and so the results from this part of the study indicate the 8 bit limit to be sufficient.

High Pass Edge Filter. The edge magnitude values produced when the threshold level is set to 0 can be used to produce edge enhanced images, either by themselves or by combining the enhanced information with the original image in a subsequent PE. Directional edge filters can also be realised by omitting one of the orthogonal pair of detectors, or by designing a detector for $\pm 45^\circ$ edges. Figure 9.11 shows an edge magnitude image produced by two orthogonal detectors. Comparison with the image produced by the point enhancement filter in Figure 9.6 shows a more even edge enhancement with this new filter.



Figure 9.11: High Pass Edge Filtered Version of the Low Contrast Rectangle.

9.4.2.2 Hexagonally Sampled Images.

The same tests for edge distortion and edge gradient magnitude accuracy were performed for hexagonally sampled images. Distortion comparison, with and without distributed scaling, was performed on the hexagonally sampled version of the low contrast rectangle image. Figure 9.12

shows one corner of this image together with a pair of edge detected images that have been formed with the threshold level set as high as possible while still maintaining a connected image, and a pair with a lower threshold level. The latter results in a thicker edge, but is more likely to be used in practice, perhaps followed by a line thinning operator. The top left edge aligns well with the grid of sampling points and has lead to a strong response in all the edge detected images. In contrast the lower left edge has not aligned well and some aliasing can be observed in all the images. The edges have not been distorted by the operation, or to any significant degree, by the sampling grid. The left hand pair of edge images, formed without distributed scaling, are identical to the right hand pair that were formed with 8 bit distributed scaling.

Further tests were performed on simulated edges at angles between 0 and 90 degrees in 2 degree steps to measure edge magnitude accuracy. The original test is presented in Section 4.4.1.3 where the results were used to compare the square and hexagonal system edge detectors. Here the comparison is between edge detection in PEs without distributed scaling, with distributed scaling limiting the word widths to 8 bits, and with distributed scaling limiting the word widths to 9 bits. By setting the PE threshold level to 0, the magnitude of the detected edge is output as an 8 bit integer value without a thresholding process being applied. The edge height is 10 brightness units in each case and the detector width is 2 pixels, so the gradient output should be 5. The edge gradient points for the PE without distributed scaling are plotted in Figure 9.13. The edge gradient points for all three PEs are identical. The plotted points are of similar value to those on the curve plotted in Figure 5.4. This was obtained by a program employing double precision arithmetic throughout, whereas here the PE arithmetic is mostly integer. This explains the small differences between the two figures. Distributed scaling with the word widths limited to 8 bits appears to be optimum for the hexagonal detector.

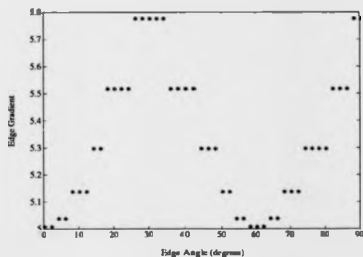


Figure 9.13: Hexagonal Operator Edge Gradient Response for Edges, Gradient = 5.0, Oriented 0 to 90°. Key: * Without Distributed Scaling.

9.4.2.3 An Alternative Thresholding Strategy.

With the previous scheme, for square sampled images, the edge gradient magnitude is calculated as the square root of the sum of the squares of the responses of the two detection templates. A similar calculation is used on hexagonally sampled images. The square root operator is necessary if high pass edge filters are to be realised, but an alternative of comparing the square of the gradient magnitude with the square of the threshold exists if they are not. The design of the scaling modules forces values that exceed the permitted range to take the end value of the range, but with the thresholding operation, outputs are set to one level if the input is above the threshold, or the other level if the input is below it. The input can be the nonlinear magnitude squared signal described above, if the *scale_3* and threshold squared constants are chosen correctly. This results in the errors introduced by the square root approximation module described in Section 8.2.3.4 being eliminated. This method is described graphically in Figure 9.14.

The PE simulation incorporating 8 bit distributed scaling was modified to incorporate this

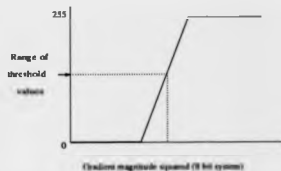


Figure 9.14: A Typical Gradient Magnitude Squared to Threshold Transformation.

thresholding technique. It is easy to implement in practice. Figure 9.15 shows the edge detected low contrast rectangle image. The result is nearly identical to that of image (c) in Figure 9.9 which was produced using the conventional thresholding technique.

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 * * *
0 0 0 0 0 0 0 * * *
0 0 0 0 0 0 * * *
0 0 0 0 0 * * * 0
0 0 0 * * * 0 0 0
0 0 0 * * 0 0 0 0
0 0 0 * * 0 0 0 0
0 0 0 * * * 0 0 0
0 0 0 * * * * *
0 0 0 0 0 0 * * *
0 0 0 0 0 0 0 0 *
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

```

Figure 9.15: An Edge Detected Low Contrast Rectangle Image, Alternative Threshold Procedure.

9.4.2.4 Discussion.

None of the tests applied to the PEs with images sampled on either grid, or with any word width limiting alternatives tried, reveal any edge discontinuities or edge distortions due to the edge detection operators. The more precise edge magnitude measurements indicate that for the square system detector, a word width limit of 9 bits is required if the PE is to produce identical results to

a non scaled output PE. However the errors introduced by an 8 bit limit are small and likely to be acceptable in the majority of applications. For the hexagonal detector an 8 bit word width appears optimum. It is not a general result that the word width can be limited more for a hexagonal than for a square process. For these edge detection operators, less scaling is required with the hexagonal detector where the total scaling requires a division by 6, distributed as 1 2 3, as opposed to the square detector where the total scaling requires a division by 8, distributed as 2 2 2. Scaling only operates on single pixels and so does not depend on the sampling scheme, but the hexagonal local operators used during this project have fewer members in their regions of support, and consequently require less scaling. As the edge step size is increased, the errors due to the detectors increase proportionately, whereas the scaling errors remain constant. The step size of 10 (total system resolution 255), used to produce the results above, was thought to be the smallest that will need to be detected in practice. Eight bit scaling produces good results with such an edge, but for lower edges, the inherent fixed magnitude errors which are produced may require a nine bit word width PE to be used. The alternative thresholding procedure, in which the square of the edge magnitude is compared with a threshold value, works satisfactorily, and the edge enhancement filter presented in this section enhances edges more evenly than the point enhancement filter of Section 9.4.1.

9.4.3 Sort and Select Operators.

Median filters and grey scale morphological operators are used to test the PE simulation for this class of operators. Median filters are described in Section 4.4.1.2, and grey scale morphological operators in Section 4.4.1.5. These operations are specified by a control data frame variable which sets up a processing path within the PE consisting of the *array multiplication*, and the *sort-square sampled data* or *sort-hexagonally sampled data* subroutines. The distributed scaling routine: *scale.1* is always run, but usually the multiplication coefficients take values of 1 or 0, and so the scaling has no effect. However, certain grey scale morphological structuring elements may require a 3-D shape,

with brightness being the 3rd dimension, which will require the scaling routine to be implemented.

9.4.3.1 Median Filters.

The median technique implemented in the simulation is the standard true median algorithm as opposed to a pseudo-median algorithm. As the scaling routine is passive, the results here are the same as those obtained by any other implementation. Only a simple comparison between equivalent square and hexagonally filtered images is included here. The array multiplication allows masks to limit the local area that will produce the median value so that the various edge preserving techniques described by Bovik et al. [16] can be implemented. Figure 9.16 shows an unsmoothed square sampled image together with a median filtered version. The local area was a 3×3 square. In comparison Figure 9.17 shows an unsmoothed hexagonally sampled image together with a median filtered version. The local area contained 7 elements arranged hexagonally. The results show smoothing as expected.

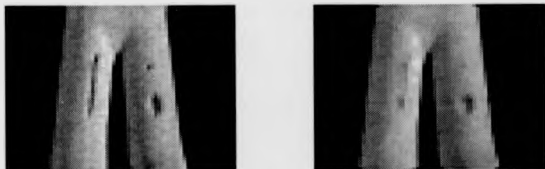


Figure 9.16: Square Sampled 64×64 image, Unsmoothed and Smoothed by a Square 3×3 Median Filter.



Figure 9.17: Hexagonally Sampled 64 x 64 image, Unsmoothed and Smoothed by a 7 Element Median Filter.

9.4.3.2 Grey Level Morphology.

Using the sort and select operator it is possible to perform grey level morphological erosions and dilations with the PE. Then by following one operation by the other in subsequent PE's, morphological grey level opening and closing can be performed. The structuring element can be shaped in 3-D and is applied at the array multiplication stage. The resulting data are then sorted and a control data frame variable used to select the maximum value of the set for dilation, or the minimum value for erosion. Figure 9.18 shows a grey level test image consisting of five unconnected items. After closing with a uniform brightness 3x3 structuring element, the five items have merged into two major items indicating the presence of two objects within the image.



Figure 9.18: Square Sampled Five Item Input Image Together With Grey Level Morphologically Closed Version.

Figure 9.19 shows a hexagonally sampled test image, again consisting of five unconnected items. The next image in the sequence shows the test image after dilation by a seven pixel structuring element, and the final image is the closure of the test image by the structuring element. Again the five items have merged to indicate the presence of two objects.

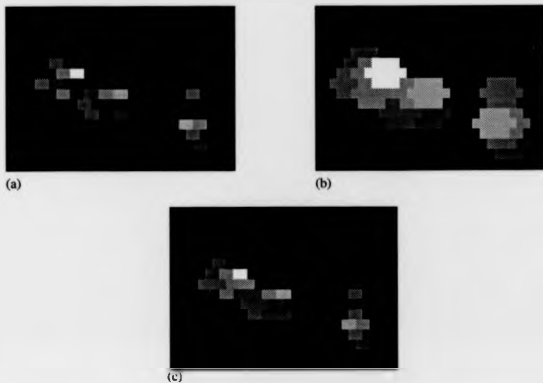


Figure 9.19: (a) Hexagonally Sampled Five Item Input Image. (b) Grey Level Morphologically Dilated Version. (c) Grey Level Morphologically Closed Version.

9.4.3.3 Discussion.

The results show the correct implementation of these operators by the simulation, on both square and hexagonally sampled data.

9.4.4 Parallel Logical Operators.

The results show that the logical units and the flexible interconnections implemented are sufficient to provide morphological and line thinning operations on both square and hexagonally sampled data.

This operator provides up to ten parallel binary operations on the local area pixels. The results of these are then combined logically to produce an output pixel or to toggle the central pixel of the local area and output its new value. The operation is selected by a control data frame variable which sets up a processing path within the PE consisting of the *parallel logical operation (square)* or the *parallel logical operation (hexagonal)* routines. The distributed scaling routines are not called as all results are binary and limited to 1 bit except for the output which is expanded to take the value 0 or 255.

An array of control data frame variables configure the logical units of the operator as illustrated in Figure 8.13. The control variables were calculated and the parallel operations individually tested before the final results presented below were produced. The execution is illustrated here by morphological and thinning operations.

9.4.4.1 Binary Morphological Operations.

For both square and hexagonally sampled images the dilation and erosion operators have been realised. For the square system a square 3x3 structuring element is used and for the hexagonal a seven pixel structuring element. In each case only one of the parallel operators is required as these are simple operators. As with the grey level morphological operators opening and closing can be performed by two PEs. Figure 9.20 (a) shows a letter "P" with a single pixel protrusion and a single pixel indent on its perimeter. This image is shown dilated in part (b) and then, after opening, in part (c). The opening has removed the indent. Part (d) shows the original image eroded, and part (e) the

closed image. After closing the protrusion has been removed.

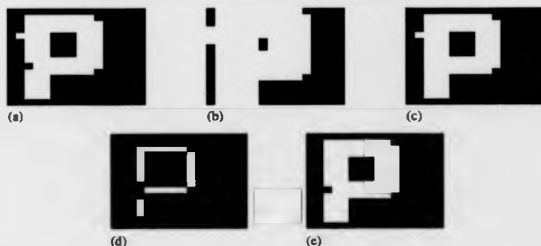


Figure 9.20: (a) Binary Image. (b) Morphologically Dilated. (c) Morphologically Closed. (d) Morphologically Eroded. (e) Morphologically Opened.

9.4.4.2 Thinning Operations.

These operators are introduced in Section 4.4.1.4 and in Section 8.2.2.2 the "Thin-OPTA" algorithm developed by Chin et al. [27] is presented as being suitable for pipeline processing. A similar thinning algorithm for use with hexagonally sampled images was developed in Section 6.3. The Thin-OPTA algorithm and its hexagonal equivalent have been realised as parallel logical operators and simulated.

Thin-OPTA Implementation. The algorithm requires the application of eight 3×3 thinning templates, a 4×3 restoring template, and a 3×4 restoring template to the local area surrounding each pixel. This requires a PE with a 5×5 local area capability because a 4×4 , even sized, local area cannot be formed with this simulation. The templates contain logic 1, logic 0, and don't care conditions. Further don't care conditions are used to pad the operators out to the 5×5 area. A series of ten control file variables are set to change the local area pixels to logic 1 in the don't care positions. The next series of ten control file variables are used to invert the local area pixels in the logic 0 positions

so that any pixel logic 0 values become logic 1s. Referring back to Figure 8.13, the first series of AND gates operate for each template, and a logic 1 output results if the data pattern specified by the template exists. The outputs from the eight thinning template operators are then ORed to produce a "Thin" signal and the outputs from the two restoring templates ORed to produce a "Restore" signal that, if true, prevents the "Thin" signal from switching the local area central pixel to zero before output.

Figure 9.21 shows an edge detected sand core image and an image that has been thinned by a series of five PEs running the Thin-OPTA program. The resulting skeleton contains no gaps attributable to Thin-OPTA and is a good fit to the medial axis of the original image.



Figure 9.21: Edge Detected Sand Core Image Together With Thin-OPTA Image.

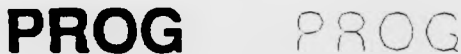


Figure 9.22: Original Square Sampled Text Image Together With Thin-OPTA Image.

With printed character thinning the PE simulation produced identical results to the Thin-OPTA program used to produce the results obtained in Section 6.3, by a non pipelined implementation of the algorithm. The thinning demonstrated in Figure 9.22 required an initial morphological

smoothing followed by 14 Thin-OPTA PE stages to produce the skeleton. This pipeline assembly demonstrates the ability to mix PEs of different local area size, the morphological operators having 3x3 local areas, and the Thin-OPTA PEs having 5x5 local areas.

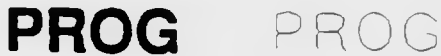
Hexagonal Thinner Implementation. This algorithm requires the application of six, seven element, thinning templates and three restoring templates that require up to five rows of elements. This requires a PE with a 5x5 local area capability. Figure 9.23 shows an edge detected sand core image and an image that has been thinned by a series of five PEs running the hexagonal thinning program. The resulting skeleton contains no gaps attributable to the program and it is a good fit to the medial axis of the original image.



Figure 9.23: Hexagonally Sampled Edge Detected Sand Core Image Together With Thinned Image.

As for the square sampled print image, the pipeline produces identical results to the program used to obtain the results in Section 6.3. As with that implementation, the morphological smoothing before thinning is unnecessary with the hexagonal algorithm. 13 PE stages are required to produce the skeleton displayed in Figure 9.24.

Conclusions. The parallel template application within the single PE design reduces the number of PEs required for a thinning operation. The sand core image required 5 PEs and the printed character image 14 PEs to effect thinning. Ten times as many PEs would be required if the parallel logical



PROG PROG

Figure 9.24: Original Hexagonally Sampled Text Image Together With Thinned Image.

operators were not available. The number of PEs required to thin a binary image is a function of the thickness and complexity of the features within the image. A modification to the simulation enabled a count of the thinning activity to be output to the user terminal. A similar feature should be incorporated in the hardware PE so that it can report thinning activity to the control computer via the communications channel, enabling the control computer to tailor the length of the pipeline accordingly.

9.5 Conclusions.

The PE simulation shows that the PE described in Chapter 8 can perform the range of image processes identified in Chapter 4 as being required for controlled lighting and industrial image processing. The simulation has confirmed the overall system specification, showing that processing can be performed on a stream of data passing along a pipeline of individually programmed PEs arranged as a pipeline. Various sizes of images were processed successfully, and the scheme of padding the surrounds of images with null pixel values to avoid inter-image interference was shown to be correct. PEs incorporating 3x3 and 5x5 local areas can be realised and mixed together successfully in single pipelines.

The internal modules within the PE have been simulated as separate subroutines and are functionally correct. The PE simulation is able to assemble square and hexagonal local areas using a one dimensional buffer to represent the line delays. The main processes to be used for a particular

application are selected by an appropriate control data frame variable. The main processes are sufficient to process the diverse set of applications presented in this chapter. The differences between the square and hexagonal versions of the processes are small, and so the hardware devices will require little extra circuit area to process images on the two sampling grids.

The distributed scaling routines have been implemented in two versions. One limits the word width to 8 bits, and the other to 9 bits. The 8 bit word width limit is satisfactory for use with most operators, but introduced some errors of equivalent magnitude to the image operator errors, with the convolution operators on the square grid system. In addition, with these operators, it is sometimes more difficult to distribute the scaling factors and to maintain the correct overall scaling with the 8 bit limit than with the 9 bit limit. The 9 bit word limit introduces few errors to the square system processes tested. For hexagonal system processes few errors are introduced by either the 8 or 9 bit word width limit.

The post processes implemented are all necessary except for the look up table, which may be useful to correct for overall scaling errors introduced by distributed scaling with an 8 bit word width, but these errors are better corrected by implementing a 9 bit word width limit. Two output messages, that can provide useful information to the host control computer, are a parallel logical operator activity count, and a count of the number of pixels that are outside the normal 0 to 255 range during the processing of a complete image. The control data frame specified in Chapter 8 has been shown to be appropriate.

Four distinct groups of main processes are implemented: Convolution, edge detection, sort and select, and parallel logical. Often within a group, several similar operators have been compared, as have operators for the square and hexagonal sampling grids.

Convolution operators. On the square grid two Gaussian low pass spatial filters have been implemented, G_1 with coefficients from the set $\{1,2,4\}$ and G_2 with coefficients from the set

{3,7,16}. Small numerical differences can be seen in the image data processed by the 2 filters, and further differences when the word widths are limited to 8 or 9 bits. No distortions could be attributable to either word width limit. The numerical differences are so small they cannot be observed in images displayed on a TV monitor. It is concluded that the simpler G_1 operator with its lower coefficient values, and the 8 bit word width limit are all that need be implemented in an eventual PE fabrication. On the hexagonal sampling grid a single filter, G_4 , has been implemented. Its performance was similar to that of the square system filters. A high pass filter has been implemented with an 8 bit word width limit. It produces larger numerical errors than the low pass filters, but these are acceptable for human viewing. A 9 bit word width limit PE would produce more accurate results for a machine vision application, but it is unlikely that such a filter would be required in practice.

Edge detection operators. Operators have been implemented on equivalent square and hexagonal image data, both with and without distributed scaling. Results with word width limits of both 8 and 9 bits are presented. The operators implemented on the PE produced similar results to those obtained by other means in Chapter 5. No distortion or gaps in edges could be attributed to the distributed scaling technique or to the word width limits.

Further tests on edge magnitude values reveal that for the square system, the word width must not be limited to below 9 bits, if the limiting is not to effect the results. For the hexagonal system the word width must be at least 8 bits. However, if a word width of 8 bits is used with the square system, for edges with a height greater than 10 brightness units in a 255 unit system, then the errors are small compared to the other errors inherent in the detection process.

The edge enhancement filter produced visibly acceptable results, but the 8 bit word width limit introduced some numerical errors in the processed output images. Both this edge enhancement filter and the point enhancement filter presented earlier will be implemented in the final PE design.

as either may be optimal for a particular application.

Sort and select operators. The results show the correct implementation of the median and grey scale morphological operators by the PE simulation on both square and hexagonally sampled data.

Parallel logical operators have been implemented on both the square and hexagonal systems. A binary morphological operator requiring a 3×3 local area was implemented together with a 5×5 local area thinning operator. The simulations demonstrate the correct implementations of the algorithms, and the compact nature of the pipeline when algorithms, such as those for thinning which require the application of many templates, are processed in parallel. The parallel logical operator circuit that was developed in Chapter 8 is thought to be adequate for most applications.

Summary. The simulation results show that all the required operators are achievable, but that the more complicated Gaussian G_2 filter is redundant. It would thus be possible to reduce the coefficient word width to 5 bits, but this would limit possible future uses of the device, and should only be considered at the fabrication stage if the device is becoming too large. An 8 bit word width has been shown to be adequate, but a 9 bit width produces slightly more accurate results. It is easier to achieve the correct overall scaling with a 9 bit device, but this advantage disappears if the coefficient word width is kept to 6 bits with the remaining operators. An 8 bit word width PE is therefore considered further in the fabrication chapter, but if space on the device permits, the design can be upgraded to 9 bits.

Chapter 10

Conclusions and Further Work.

The work presented in this thesis has covered the three main areas of the processing of hexagonally sampled images, the processing of images from an industrial or a controlled lighting environment, and the design of a computer architecture for the fast, economic, and reliable processing of both hexagonal and square sampled images.

10.1 Conclusions.

10.1.1 The Hexagonal Sampling of Images and Their Processing.

10.1.1.1 Sampling.

Efficient biological vision systems have evolved that sample images with hexagonal arrays of sensors. Earlier image processing workers have shown that for 2D circularly band limited signals of equal high frequency resolution, about 13% fewer samples are required if the sampling points are arranged in a regular hexagonal grid, than if they are in a square grid pattern. This spatial relationship between sensors leads to a hexagonal interconnection of processing elements in the initial stages of the image processing. These initial stages, where the outputs of a visual process are still related to the geometry of the sensor grid have been referred to as low level operations.

The earlier chapters of the thesis have reviewed image sampling and low level processing. Hexagonal, square, and rectangular sampling grid geometries have been explained, and the various pixel shapes that can be associated with the individual sampling points, and used to tile completely the image area, have been presented. Tiles that completely cover the image without gaps are useful for display devices. With raster scanned devices such as the TV monitor, square and rectangular shaped pixels can be used to build up the image, for display, line by line. Such pixels are considered to exhibit equal brightness over their entire area, whereas a sophisticated image reconstruction scheme may interpolate the values between the sampling points. As discussed in Section 3.4.3, advantages exist for computer graphics routines operating on hexagonal grids as there are more axes of symmetry and the aliasing of straight line features can be reduced in comparison to square grid routines. The human interpretation of both straight line and circular features is easier with hexagonal grid sampled images. This is discussed in Section 6.2.

As observed in Section 3.3.1.2, the active area of the image sensor may or may not be equal to the pixel shape used when the image is displayed. Even if it is, the response within the area of the sensor may not be uniform. For a 2D array of sensors, the signal band-limiting will be a function of the size of the sensor area, the transfer function of the sensor with respect to the position within the active area, the optical components before the sensors, and the electronic processing after the sensors. As discussed in Chapter 3, circularly band-limited signals exhibit equal high frequency resolution in all spatial directions. A system constructed with sensors which produce circularly band-limited signals will process an image with equal quality, independently of its orientation with respect to the array of sensors. It is concluded that this could be important for many applications, such as a case where small defects must be detected in a series of randomly oriented parts on a production line.

Some sensors such as the vacuum tube TV camera will produce a good approximation to a circularly band-limited signal, whereas others such as CCD arrays may not. Individual CCD

elements may be square or rectangular in shape, but it is concluded that modelling the signals as square or rectangular band-limited signals can be inaccurate. For many applications the relationship between the sampling point geometry and the precise value sampled will be unimportant, but where precise distance measurements, precise measurement of edge angle or magnitude, or the detection of small features is required, then consideration of this relationship should be given.

It is concluded that the ideal sensing system should sample the image on a 2D hexagonal grid, as a minimum number of sampling points are required, and that the signals should be circularly band limited before sampling so that equal high frequency resolution can be achieved in all spatial directions.

10.1.1.2 Processing.

Having identified regular hexagonal sampling as being an efficient scheme, low level image processes that would operate on such a scheme were evaluated. As discussed in Chapter 4 hexagonal morphological, distance measuring, and some Fourier plane image processes have been developed by previous researchers.

For the processing of images from controlled lighting environments such as are found in many industrial situations, image processing operations which require only a small local image area for support are often sufficient. A group of efficient and accurate operators for use on square grid sampled images has been identified, and equivalent hexagonal scheme operators designed and compared with them as a part of this project. It was found that the symmetry of the hexagonal grid leads to the formation of highly populated concentric shells of equidistant neighbours surrounding the central pixel of a local area. This was found to make operator design and coefficient calculation easy compared with designs for the square grid. The basic operators designed include Gaussian filters, median filters, point enhancement filters, and edge detectors. These were easy to design and were found to performed equally as well as their square system counterparts. In general they

require about 13% fewer members for a given physical local area, and, additionally, are quicker to compute as fewer individual coefficient values are required than with the square scheme.

In Chapter 5 the new hexagonal scheme edge detector was extensively compared with the square system Sobel detector. For both detectors, implemented on a modern computer, it was found that the exact magnitude could be calculated in less time than that required by the approximate magnitude calculation used by some researchers to reduce processing time. The simpler operators used with the hexagonal scheme were found to lead to faster processing, and, together with a reduced number of sampling points required with the hexagonal grid, to a processing time saving approaching 44% as compared to the square grid operators. This time saving is, however, machine dependent.

Comparable magnitude and angular accuracy was demonstrated for the edge detection operators on both sampling schemes. However, whereas the square system Sobel operator produced low errors with sampled values that were calculated by averaging square spatial support regions, the hexagonal operator produced low errors when the sampled values were calculated by averaging circular spatial support regions. When designing high accuracy processing systems, the characteristics of the sensors should be investigated. It is likely that sensors producing circularly band-limited signals will be easier to design than sensors processing an average value for the intensity over a pixel shaped region. This should lead to an advantage for the processing of hexagonally sampled images.

A new binary hexagonal image processing operator was developed as an equivalent to the square sampling scheme thinning operator designed by Chin [27]. The hexagonal operator can be coded to compute in less time than the square, as there are fewer templates required. The quality of the results is discussed in Section 10.1.2.3.

10.1.2 Image Processing in Industrial or Controlled Lighting Environments.

To be accepted for use by a competitive industry, a vision system must be reliable, quick in operation, and cost effective. The local image processing operators that have been designed for use

with hexagonally sampled images, have been shown to process images obtained in an industrial environment both efficiently and reliably in comparison to the equivalent square system processes.

10.1.2.1 Image Processing Operators.

The reliable processing of images captured in controlled lighting conditions can be obtained by using operators with support from small local areas. This point is widely accepted by industry where a good illumination system design can reduce the complexity of the image processing and thus result in a less expensive computer system. For controlled lighting conditions, the signal to noise ratio of images is high, and small support region operators were found to be sufficient for the examples reported in this thesis.

In Chapter 6, following extensive studies of the individual operators in Chapters 4 and 5, two industrial examples were studied to ascertain a useful set of processing operators that would form the basis of the operators implemented by a pipeline processor. The first concerned grey-level image processing to identify small defects in the surfaces of sand cores used in automobile engine casting, and the second concerned printed character processing. Comparisons were made between similar processes implemented on square and hexagonally sampled images. The simpler hexagonal operators and the reduced number of sampling points lead to faster processing of the hexagonally sampled images, and their quality was often found to be superior to the equivalently processed square images.

10.1.2.2 Surface Defect Detection in Sand Cores.

It is observed, in the thesis, that for a defect detection system, it is efficient to cover the surface with the lowest resolution images that will allow the minimum sized significant defect to be detected reliably, as a minimum number of images will then require processing for a given sized object. If defect categorisation is required, further images can then be obtained.

For this example it was required to detect small defects in the surfaces of the cores. The captured images were not ideal in that the pixel aspect ratio was 4:3 as opposed to 1:1 or the regular hexagonal equivalent, but the hexagonal images were easier for human interpretation, and the processed hexagonal images produced better representations of the defect's size and shape. The hexagonal processing was found to be more resistant to image noise and provided more complete defect outlines.

Detection reliability was investigated further by modelling the same defects on both the true square and hexagonal grids. A model was developed in which the smoothly changing edges of the real defects were reproduced. In general with this model, and with a further enhancement to the model that superimposed random noise on to the image data, there were only small effects attributable to the locking of the defect shapes with the two sampling grids, and it was shown that the square and hexagonal sampling systems were able to perform defect detection with equal reliability. The hexagonal system was thus shown to be advantageous as equal detection reliability was obtained with 13% fewer sampling points and simpler image processing operators that could complete the detection process in less time. In a specific detection example in which a long thin defect caused by a scratch in the sand core's surface was modelled, the hexagonal grid system was found to operate more optimally than the square in that the connectivity of the defect's outline could be more easily maintained at the ends of the defect where tight curves existed.

The technique of examining a small defect, and then modelling it, has the additional use that the modelled defect can then be scaled, moved, or oriented differently with respect to the sampling grid, and that a more confident figure for detection reliability can be obtained.

10.1.2.3 Printed Character Processing.

Printed character processing was chosen as an example with which to test a new hexagonal thinning and skeleton forming algorithm with Chin's [27] well known square system algorithm. The hexag-

onal processing was found to be superior with respect to both computation time and the quality of the resulting skeleton. The hexagonal skeletons were positioned more closely to the medial axes of the shapes and the hexagonal algorithm did not require a morphological pre-filtering stage. The algorithms are both recursive, and so the reduced number of sampling points within a hexagonal image results in fewer passes of the operators through the image. This saving is in addition to those resulting from the simpler hexagonal operators.

10.1.3 Computer Architecture.

Having identified a set of image processing operators for use with images derived from industrial and controlled lighting environments, various pipeline architectures, some of which were novel arrangements, were designed to provide cost effective real-time image processing for industrial applications in particular. A flexible, programmable PE has been designed that, as a single device, can perform all of the hexagonal and square operations that have been identified during this project as being required for industrial image processing.

10.1.3.1 Pipeline Architectures.

The pipeline architecture was chosen because the resulting processor can operate directly on the digitised raster scanned output of a TV camera or a similar sensing device. The image frame store and reformatting logic, necessary with other architectures, are not always required. Small local image areas can easily be assembled within the individual PEs, and small areas are all that are required by the image processing operators identified earlier. A wide variety of pipeline architectures can be achieved using this PE including:-

- Single video rate pipelines in which each PE is programmed to perform a different operation as a part of the overall process.

- Video rate branching and merging pipelines in which pipelines of PEs can be connected in parallel and partial results combined by a following PE's input ALU.
- Recirculating pipelines in which data has to be passed several times through a pipeline situated between two image frame stores to effect complete processing. With this system image data can be scanned in different patterns and passed through the pipeline.
- Multiple recirculating pipelines which provide several pipeline processing interconnections between frame stores. This improves the image processing rate.

No previous PE design has been flexible enough to be suitable for all of these architectures. The expected low cost and small size of the PE will allow the above architectures to be realised as single circuit board components. These flexible pipelines can act as low level pre-processors for a host computer that provides the higher level image processes. The host can provide the programming information for the PEs in the pipeline, and transmit this via the serial communications link. The PEs can also send information back to the control unit. This, together with information derived from the high level processing stages and from low level co-processors (such as histogram and Hough transform co-processors), can be used to calculate modifications to the PE programs and adaptively control the pipeline.

The video rate processing capability of the proposed non-recirculating pipelines can be used in real-time systems involving human interaction where the image processing enhances an image of the object being worked on. With such systems the PE latency must be minimised so that unacceptable delays between the initialisation of an action by the human and the feedback are not introduced. The most significant latency introduced by the proposed PE is the delay required in order to assemble the local area. This delay is equal to the scan time of only a few image lines, whereas for systems employing a frame store, frame time delays are introduced which are two orders of magnitude greater.

For an industrial machine vision inspection system, the real-time rate may be as high as the video rate, but commonly, the rate is determined by the object part rate. Latency may not be such a constraint on the pipeline design as it is with human interactive systems, because the vision system can be placed at one point on the line and the required action taken at some time later. With such a system it may be possible to use a vision system employing a recirculating pipeline.

A unique feature of the proposed PE is that the latency introduced is independent of the process being performed. This feature will simplify production line design and simulations as the latency can be calculated before the image processing application program has been completely designed. The latency is simply the number of PEs on the pipeline processor circuit board multiplied by the unit delay.

10.1.3.2 The Design of the PE.

A dual device capable of processing both square and hexagonally sampled images has been proposed and simulated. It would appear a better prospect to construct than a single sampling standard device. Hexagonal operators have been shown to be superior in many respects, but a pipeline processor may be required as an enhancement for applications already developed for the square system, or for use with sensors that adapt more readily to the square sampling grid.

Specification. A device has been proposed in Chapter 8 that will process images with sizes of up to 833x625 pixels, at the video rate. A basic device has been specified in which a square 3x3 or equivalent hexagonal local image area is assembled. The PE has two 8 bit image data inputs that are combined by the input ALU, and a single 8 bit output. This device is sufficient for most of the required image processing operators, and the design can be easily expanded for larger local area sizes. The required operators have characteristics that separate them into the four distinct groups of convolution, sort and select, parallel logical, and edge operators. To process these operators the

PE first multiplies the members of the local image area with two sets of filter coefficients and then further processes these results in one of four operator modules:-

- The convolution operator. The outputs from the multipliers are added and scaled to produce a new output value. This is performed by a tree of adders as shown in Figure 8.7. The precision of the results at each level of this tree structure has been carefully chosen to limit the size of the subsequent adders in the tree.
- The sort and select operator. An efficient algorithm was identified based on a successive approximation operation on the individual bits of the pixel values in the local area. The algorithm was developed by Danielsson [33], but an improved implementation was developed as a part of this project as shown in Figure 8.12.
- The parallel binary operator. Many logical operations can be performed on the local image area under the control of signals sent from the pipeline's host computer. The circuit is shown in Figure 8.13.
- The edge detection operator. The module produces an enhanced grey-level, or an edge map output. The circuit is shown in Figure 8.8. The magnitude operator is described by Denyer and Renshaw [42] and its application, as described in the thesis, for image processing will result in an accurate edge detection circuit.

After the above four selectable processing modules, a series of common post processes may be applied to provide thresholding, brightness inversion, and lateral shifting to the partially processed images. Programming information is input to the PE's control unit and this selects which modules are to be used and provides various coefficients for the operator calculations.

Each of the above circuit modules will operate at the video rate and they have been designed to occupy a minimum of VLSI device area. Little extra circuitry is required to realise the processing

of both square and hexagonally sampled data. For video rate processing the frame rate is identical for hexagonal and square image processes, however, the pixel rate is reduced by over 13% for hexagonal processes. For processes that are not real-time the clock rate can be kept the same, and the frame rate increased by this factor for hexagonal image processing.

Simulation. The PE has been functionally simulated and test images processed, firstly, to check the specification, secondly, as further confirmation that the set of operators identified earlier were all required, thirdly, to confirm the required control information, and finally, to compare the square and hexagonal image processing results. Distributed scaling routines that additionally include the limiting of the precision of the arithmetic operations at various points within the PE have been used. By limiting the precision, the data word widths and the size of subsequent processing modules are also limited. However, this limiting of the precision leads to errors in the processed images. The simulation has been used to find an optimum word width limit that allows accurate results for the majority of processes, while minimising the PE circuit size. An 8 bit word limit was found to be acceptable. Both hexagonal and square processes produced high quality output images. Some operators were found to be redundant, and their omission from the final design has allowed the range of the convolution coefficients to be reduced, resulting in a simpler design.

10.2 Further Work.

10.2.1 Computer Vision and Hexagonal Sampling Grids.

10.2.1.1 Operators Requiring Large Areas of Support.

The hexagonal image operators that have been implemented all receive support from only small local areas. These operators are more efficient than their square sampling grid counterparts as fewer points are required for a given area, fewer different coefficient values are involved, and there are

more axes of symmetry. The use of hexagonal operators requiring larger areas of support should be explored. With such square system processes, the operators are often separable which leads to more efficient processing. Hexagonal operators are not separable, but highly efficient processing can be achieved as with Mersereau's [124] HFFT. A hexagonal sub-pixel accuracy edge detector may perform optimally when compared to a square system equivalent, as the estimation of edge points on such a regular grid with equal high frequency information coded in all directions should result in greater accuracy in specifying the edge position.

10.2.1.2 The Design and Response of Sensor Elements.

Further work should be undertaken concerning the design of sensor elements, such as CCD arrays, for hexagonal sampling. Additionally the responses of single elements, and individual elements within an array, need to be measured to determine the band limiting of the signals within the 2D image plane. Sensor designs can then be optimised for the particular sampling geometry. Hexagonal or circular band limiting is optimum for the hexagonal sampling grid, and square band limiting for the square grid. The hexagonal arrangement of rectangular shaped elements in a CCD array is easily achieved, and such arrays are less expensive to manufacture than square arrays. Inexpensive TV cameras for domestic use have been produced with hexagonal CCD arrays. A high quality camera now needs to be designed.

10.2.1.3 Additional Case Studies.

The images in the case studies that have been described in Chapter 6 and some of the images used to test the pipeline processor simulation were from diverse application areas. Even so, further case studies need to be pursued to help build libraries of relevant processes. These can be pursued with the PE simulation, but research and development will be hastened once a hardware PE has been realised.

10.2.1.4 The Adoption of Hexagonal Image Sampling.

Several advantages have been found for hexagonal processing. The number of sampling points required to cover an area is 13.4% less, computation time savings in excess of 44% have been demonstrated, higher quality processed images can result, and human interpretation of the images can be easier, than with the square equivalents. However, equipment manufacturers, researchers, and application engineers may be reluctant to change to hexagonal sampling because new sampling equipment and low level image processing software, together with the interfacing of these to higher level processors and processes, will need to be engineered and introduced concurrently. Whole libraries of low level processing routines that have been written and tested for use with development systems, and trusted macro programs containing combinations of operators will need to be rewritten. For a new manufacturer entering the field without any commitment to previous products, the advantages offered by hexagonal processing will provide their products with a competitive edge.

Some products, such as document readers, are mass produced for a highly competitive market. The hardware and software for such products is likely to be re-designed for each new system. For such products the change to hexagonal sampling and processing will enhance its profitability.

10.2.2 Pipeline Processors.

10.2.2.1 The Completion of the PE Design and Fabrication.

To date the PE has been specified and block diagrams of its overall structure and internal processing modules produced. These diagrams can be found in Chapter 8. The design of these modules have been taken to a level where individual functional components such as multipliers, adders and latches have been identified. Both the individual modules and the complete PE have been functionally simulated using programs written in a high level language. The circuits for each of these modules now need to be investigated for VLSI implementation. It is expected that the PE will

be able to be fabricated as a single VLSI device.

The PE circuit elements that will require the largest areas of silicon for their fabrication are likely to be the line delays and the multiplier arrays. There are two commercially available VLSI devices that contain such circuit elements and operate at the video rate. The Hitachi HM6302 [78] is an inexpensive integrated device that can store two eight bit wide digitised video lines of variable length up to 1024 bytes long, and the Inmos A110 [14, 87] is a programmable single chip filter, described in Section 7.5.2.1, that contains three 1120 byte programmable length line delays and twenty one 8x8 bit multipliers. The PE device proposed in this thesis requires two 8 bit wide video line delays with programmable lengths of up to 833 bytes each. Being shorter, these will require less silicon area than the line delays in either the HM6302 or the A110 device. The number of multipliers in the proposed PE is eighteen of size 8x6 bits located in the multiplier arrays, and two of 13x6 bits located in the convolution modules. These should occupy less silicon area than the larger and more numerous multipliers included in the A110 device, but if a reduced specification PE device occupying a smaller area is required, then the number of multipliers can be reduced by nine if the alternative edge detection module proposed in Section 8.3.2.2 is used. The existence of these two commercial VLSI devices indicate that it should be possible to construct the PE as a single integrated device that will operate at the video rate.

The fabricated device will operate several orders of magnitude faster than the PE simulation, and enable complicated image processes to be evaluated within a reasonable time period.

10.2.2.2 The Host Computer.

The host computer originates the control signals for the pipeline, provides the higher level processing, and reports the information obtained to a human operator or further systems. The image data interface between the pipeline and the host will need careful design. The data rate will have been reduced by the pipeline in that grey level information will probably have been reduced to binary

information. Further reduction could be achieved by including a chain coding device operating on the pipeline output data stream within the interface, but probably one or more image frame stores capable of storing complete images should be incorporated.

The high level processes will take the information from the low level processes computed in the pipeline, and from these spatially related data, produce results that are no longer ordered as in the original sampling grid. A pipeline processor is not appropriate for high level tasks. A SISD computer, a MIMD array, or a pyramid processor would be more appropriate. Valkenburg et al. [179] are researching a MIMD, transputer based array, as a high level processor that is interfaced to a pipeline. Such a machine could form the basis of the host computer for many of the pipeline architectures reported here.

10.2.2.3 Adaptive Pipeline Control.

The control computer will choose both the parameter values that an operator will use, together with the operator function, for every PE within a pipeline. An adaptive control mechanism will allow various parameters, measured by the system, to be used in the calculation of new coefficients and operator configurations for system optimisation. The high speed control communication possible within the system will allow such control to be performed in real-time. Initially simple adaptive control algorithms will be developed. For example, a system that adapts to changes in lighting level, or a system that chooses the shape of the local area required by a median filter so that it can preserve edge information, could readily be developed. Eventually more complicated adaptive control mechanisms could be researched, perhaps with an artificial neural network control computer that can be trained to respond correctly to the differing conditions.

10.3 In Summary.

To be accepted for use by industry an image processing system must be fast, reliable and cost effective. Each of these requirements has been addressed in this thesis. Improvements in processing speed have been achieved by using the hexagonal image sampling grid and the hexagonal processing operators designed for use with it. Further improvements can result with the use of the video rate pipeline processors presented. The set of processing operators assembled have been shown to reliably process images captured under controlled lighting conditions, and additionally hexagonal operators have been shown to process noisy images more reliably than their square system counterparts. Finally the proposed PE, when fabricated as a VLSI device, will enable cost effective systems to be constructed.

Bibliography

- [1] R J Ahlers. Industrial robots for measurement and inspection purposes. *Proc SPIE*, 1008:2-12, 1989.
- [2] N Akiyama. Recent progress in pattern recognition technology for automated production lines. *Bulletin of Japanese Soc Precision Engineering*, 18:165-70, 1984.
- [3] Alvey. Research strategy for speech signal and image processing. *Alvey Directorate, UK*, 1987.
- [4] M Annaratone and H T Kung et al. Warp architecture and implementation. *Proc. IEEE 13th Int. Symposium on Computer Architecture*, pages 346-356, 1986.
- [5] D Aspinall. Structures for parallel processing. *IEE Comput. and Cont. Engg. Journ.*, 1(1):15-22, 1989.
- [6] Automatrix UK Ltd, Coventry, UK. *Personal Communication*, 1990.
- [7] D H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattrec*, 13(2):111-122, 1981.
- [8] B G Batchlor. Integrating vision and AI for industrial applications. *Proc SPIE*, 1193(15), 1990.

- [9] B G Batchelor and C C Bowman et al. Developments in image processing for industrial inspection. *Proc SPIE*, 730:34-46, 1986.
- [10] B G Batchelor and S M Cotter. Detection of cracks using image processing algorithms implemented in hardware. *Image and Vision Comput.*, 1(1):21-29, 1983.
- [11] B G Batchelor, D A Hill, and D C Hodgson. *Automated Visual Inspection*. IFS Publications North Holland, 1985.
- [12] K E Batcher. Design of a massively parallel processor. *IEEE Trans Computers*, 29:836-840, 1980.
- [13] S B M Bell, F C Holroyd, and D C Mason. A digital geometry for hexagonal pixels. *Image and Vision Computing*, 7(3):194-204, 1989.
- [14] N Birch. The Inmos ims A110 digital image processing chip. *IEE Colloquium, At Savo Place 16/12/88, Pages 7/1-7/4*, 1988.
- [15] G Boccignone et al. Investigation on a structural solution of merged characters in ocr. In *Recent Issues in Pattern Analysis and Recognition*, Ed V Cantoni, Springer Verlag, pages 303-315, 1987.
- [16] S D Bovik, T S Huang, and D C Munson. The effect of median filtering on edge estimation and detection. *IEEE Trans PAMI*, 9(2):181-194, 1987.
- [17] C C Bowman. Getting the most from your pipelined processor. *Proc SPIE*, 1004:202-210, 1988.
- [18] C C Bowman and B G Batchelor. Kiwivision a high speed architecture for machine vision. *Proc SPIE*, 849:42-51, 1987.

- [19] P J Burt et al. A pipelined pyramid machine. In *Pyramidal Systems for Computer Vision*, Ed V Cantoni and S Levialdi, Springer Verlag, Berlin, Germany, pages 133-152, 1986.
- [20] J Canny. A computational approach to edge detection. *IEEE Trans PAMI*, 8(6):679-698, 1986.
- [21] J F Canny. Finding edges and lines in images. *Technical Report 729*, MIT, Artificial Intelligence Lab, USA, 1983.
- [22] V Cantoni and S Levialdi. Multiprocessor computing for images. *Proc IEEE*, 76(8):959-969, 1988.
- [23] D P Casasent. Acousto optic processor for inspecting cigarette labels. *Proc SPIE*, 1197:181-190, 1990.
- [24] A Chianese et al. A structural method for handprinted character recognition. In *Recent Issues in Pattern Analysis and Recognition*, Ed V Cantoni, Springer Verlag, pages 289-303, 1987.
- [25] R T Chin. Automated visual inspection 1981 to 1987. *Comput Vision Graphics and Image Processing*, 41:346-381, 1988.
- [26] R T Chin and C A Harlow. Automated visual inspection a survey. *IEEE PAMI*, 4(6):557-573, 1982.
- [27] R T Chin et al. A one pass thinning algorithm and its parallel implementation. *Computer Vision Graphics and Image Processing*, 40:30-40, 1987.
- [28] R W Connors et al. Identifying and locating surface defects in wood. *IEEE Trans PAMI*, 5(6):573-582, 1984.
- [29] Contex Vision, Linköping Sweden. *Product Literature*, 1989.

- [30] J F Craine and G R Martin. *Microcomputers in Engineering and Science*. Addison Wesley, 1985.
- [31] T R Crimmins and W M Brown. Image algebra and automatic shape recognition. *IEEE Trans on Aerospace and Electronic Systems*, 21(1):60-67, 1985.
- [32] J D Crisman and J A Webb. The warp machine on navlab. *IEEE PAMI*, 13(5):451-465, 1991.
- [33] Per-Erik Danielsson. Getting the median faster. *Computer Vision Graphics and Image Processing*, 17(1):71-78, 1981.
- [34] Databuc Inc., Peabody, MA, USA. *Maxvideo System*, 1989.
- [35] E R Davies. Circularity a new principle underlying the design of accurate edge orientation operators. *Image and Vision Computing*, 2(3):134-42, 1984.
- [36] E R Davies. Image space transforms for detecting straight edges in industrial images. *Pat Recog Lett*, 4:185-192, 1986.
- [37] E R Davies. Design of optimal gaussian operators in small neighbourhoods. *Image and Vision Computing*, 5(3):199-205, 1987.
- [38] E R Davies and A I C Johnstone. Methodology for optimising cost speed tradeoffs in real time inspection hardware. *IEE Proc pt e*, 136(1):62-69, 1989.
- [39] E R Davies and A P N Plummer. Thinning algorithms a critique and a new methodology. *Pattern Recognition*, 14:53-63, 1981.
- [40] L S Davis. A survey of edge detection techniques. *Comput Graphics and Image Processing*, 4:248-70, 1975.

- [41] E De Micheli et al. Localisation and noise in edge detection. *IEEE Trans PAMI*, 11(10):1106-1116, 1989.
- [42] P Denyer and D Renshaw. *VLSI Signal Processing: A Bit Serial Approach*. Addison Wesley, Wokingham, UK, 1985.
- [43] E S Deutsch. Thinning algorithms on rectangular hexagonal and triangular arrays. *Communications of the ACM*, 15(9):827-837, 1972.
- [44] H S Don and K S Fu et al. Metal surface inspection using image processing techniques. *IEEE Syst Man Cybern SMC*, 14(1):139-146, 1984.
- [45] R O Duda and E H Hart. Use of the hough transformation to detect lines and curves in pictures. *Commns of ACM P11-15*, 15(1), 1972.
- [46] R O Duda and E H Hart. *Pattern Classification and Scene Analysis*. J Wiley, New York, USA, 1973.
- [47] D E Dudgeon and R M Mersereau. *Multidimensional Digital Signal Processing*. Prentice Hall Inc, Englewood Cliffs, NJ, USA, 1984.
- [48] M J B Duff. *Real Applications on Clip4*. In *Integrated Technology for Parallel Image Processing*, Ed. S Levaldi, Academic Press, London, UK, 1985. pages 153-165.
- [49] M J B Duff, B M Jones, and L J Townsend. Parallel processing pattern recognition system UCPR1. *Nucl. Instr. Met.*, 52:284-288, 1967.
- [50] R T Dunlay and S B Seida. Parallel off road perception processing on the ALV. *Proc. SPIE*, 1007:pp40-48, 1989.
- [51] C R Dyer. Introduction to the special section on computer architectures and parallel algorithms. *IEEE Trans PAMI*, 11(3):225-226, 1989.

- [52] A E Filip. A baker's dozen magnitude approximation and their detection statistics. *IEEE Trans Aerospace and Electronic Sys*, 12:87-89, 1976.
- [53] A L Fisher and P T Highnam. Computing the hough transform on a scan line array processor. *IEEE Trans PAMI*, 11(3):262-265, 1989.
- [54] M J Flynn. Very high speed computing systems. *Proc IEEE*, 54(12):1901-1909, 1966.
- [55] T J Fountain. *Processor Arrays Architecture and Applications*. Academic Press, London, UK, 1987.
- [56] T J Fountain. Array architectures for iconic and symbolic image processing. *Int. J. of Pattern Recognition and Artificial Intelligence*, Singapore, 2(3):407-424, 1989.
- [57] T J Fountain, K N Mathews, and J B Duff. The Clip7A image processor. *IEEE PAMI*, 10(3):310-319, 1988.
- [58] W Frei and C C Chen. Fast boundary detection a generalization and a new algorithm. *IEEE Trans on Computers*, 26(10):988-998, 1977.
- [59] Genrad Ltd, Maidenhead, Berks, UK. *Hilo the Universal Logic Simulation System*, 1988.
- [60] A M Gillies et al. Application of mathematical morphology to handwritten zip code recognition. *SPIE Visual Communications and Image Processing IV, Philadelphia*, 1199:380-389, 1989.
- [61] H Ginosar et al. Adaptive sensitivity cameras and sensors. *Advanced Imaging*, 5(3):50-52, 1990.
- [62] K Goebbels and G Ferraro. Automation of surface defect detection and evaluation. *SPIE Automated Inspection and High Speed Vision Architectures*, 849:117-124, 1987.

- [63] M J E Golay. Hexagonal parallel pattern transformations. *IEEE Transactions on Computers*, 18(8):733-740, 1969.
- [64] M D Goldstein and M Nagler. Real time inspection of a large set of surface defects in metal parts. *Proc. SPIE*, 849:184-190, 1987.
- [65] R C Gonzalez and P Wintz. *Digital Image Processing*. Addison Wesley, Reading, MA, USA, 1977.
- [66] D Graham and Y C Choong. A vision system for surface defects inspection using an objects model. *Proc IEE Conf on Image Processing York UK*, pages 119-123, 1982.
- [67] J Green. A new super camera for vision systems. *Robotics Age*, 7(7):6-9, 1985.
- [68] W B Green. *Digital Image Processing*. Van Nostrand, Chapter 3, 1983.
- [69] T Gross, H T Kung, M Lam, and J A Webb. WARP as a machine for low level vision. *Proc IEEE Int Conf on Robotics and Automation*, pages 790-800, 1985.
- [70] W Handler et al. Multiprozessoren für breite anwendungsgebiete erlangen, general purpose array. *GI NTG Fachtagung Architektur und Betrieb von Rechensystemen Informatik Fachberichte*, Springer Verlag, Berlin, Germany, pages 195-208, 1984.
- [71] R M Haralick. Edge and region analysis for digital image data. *Computer Graphics and Image Processing*, 12(1):60-73, 1980.
- [72] R M Haralick, S R Sternberg, and X Zhuang. Image analysis using mathematical morphology. *IEEE Trans PAMI*, 9(4):532-550, 1987.
- [73] R M Haralick et al. Textural features for image classification. *IEEE Syst Man Cybern SMC*, 3:610-621, 1973.

- [74] P Hartman and S Tanimoto. A hexagonal pyramid data structure for image processing. *IEEE SMC*, 14(2):247-256, 1984.
- [75] H L F Helmholtz. *Treatise on Physiological Optics*. Translated by JPC Southall, Dover Publications, New York, USA, 1962. Original Publication 1909.
- [76] D J Herold et al. Pattern recognition using a cmac based learning system. *Proc SPIE*, 1004:84-90, 1990.
- [77] W D Hillis. *The connection machine*. MIT Press, Cambridge, MA, USA, 1985.
- [78] Hitachi Electronic Components Ltd, Watford, Herts, UK. *HM6302 Video Line Delay*, 1990.
- [79] R M Hodgson and S J McNeill. The design of image processing systems for real-time inspection applications. *Proc IEE Conf on Image Processing London*, pages 126-129, 1986.
- [80] J Hollingum. *Machine Vision the Eyes of Automation*. IFS Publications, UK, 1984.
- [81] B K P Horn. *Robot Vision*. Mit Press, Cambridge, MA, USA, 1986.
- [82] P V C Hough. Method and means for recognizing complex patterns. *U.S. Patent 3,069,654 Dec. 18th*, 1962.
- [83] G Huang. A robotic alignment and inspection system for semiconductor processing. *Proc 3rd Int Conf on Robot Vision and Sensory Cont Cambridge ma USA*, 1983.
- [84] T S Huang. A fast two dimensional median filter algorithm. *Proc IEEE Conf on Patt Recogn and Image Analysis*, pages 128-131, 1978.
- [85] A Huertas and G Medioni. Detection of intensity changes with subpixel accuracy using laplacian - gaussian masks. *IEEE Trans PAMI*, 8(5):651-664, 1986.

- [86] J Illingworth and J Kittler. A survey of the hough transform. *Computer Vision Graphics and Image Processing* 44, pages 87-116, 1988.
- [87] Inmos Ltd., Bristol, UK, 2nd Edition. *The Transputer Databook*, 1989.
- [88] Intel Corp., Portland, OR, USA. *The Intel Concurrent Computer*, 1985.
- [89] B K Jang and R T Chin. Analysis of thinning algorithms using mathematical morphology. *IEEE PAMI*, 12(6):541-551, 1990.
- [90] R A Jarvis. A laser time of flight range scanner for robotic vision. *IEEE Trans PAMI*, 5(5):505-512, 1983.
- [91] R A Jarvis. A perspective on range finding techniques for computer vision. *IEEE trans PAMI*, 5(2):122-139, 1983.
- [92] B Kanga-Parsi and B Kamgar-Parsi. Evaluation of quantization error in computer vision. *IEEE PAMI*, 11(9):929-940, 1989.
- [93] B Kamgar-Parsi et al. Quantization error in spatial sampling comparison between square and hexagonal pixels. *IEEE Comput Soc Conf on Comput Vision and Patt Recog*, pages 604-611, 1989.
- [94] B W Kemighan and D M Ritchie. *The C Programming Language*. Prentice Hall Inc, Englewood Cliffs, New Jersey, USA, 1978.
- [95] D D Kerrick and A C Bovik. Microprocessor based recognition of handprinted characters from a tablet input. *Pattern Recognition*, 21(5):525-537, 1988.
- [96] Kontron Electronics Ltd, Watford, Herts, UK. *ProgRes 3000, the Camera for Ultra High Resolution in Color*, 1989.

- [97] H T Kung. Systolic algorithms for the CMU WARP processor. *7th. Int Conf on Pattern Recognition, Montreal*, pages 570-577, 1984.
- [98] H T Kung and J A Webb. Global operations on the CMU WARP machine. *Proc. AIAA Comput. in Aerospace, Long Beach, CA, USA, October, 1985*, pages 209-218, 1986.
- [99] H T Kung and J A Webb. Mapping image processing operations onto a linear systolic machine. *Distributed Computing*, 1:246-257, 1986.
- [100] F Lenoir et al. Parallel architecture for mathematical morphology. *Proc. SPIE*, 1199:471-482, 1989.
- [101] L N Lester and J Sandor. Computer graphics on a hexagonal grid. *Comput and Graphics*, 8(4):401-409, 1984.
- [102] H Li and J R Kender. Special issue on computer vision scanning the issue. *Proc. IEEE*, 76(8):859-862, 1988.
- [103] H Li and M Maresca. Polymorphic torus architecture for computer vision. *IEEE Trans PAMI*, 12(3):233-243, 1989.
- [104] R A Lofufo and E L Dagless et al. Real time mad edge following for mobile robot navigation. *IEE Electronics and Communication Engineering Journal, Feb.*, pages 35-40, 1990.
- [105] R M Loughheed. A high speed recirculating neighborhood processing architecture. *Proc. SPIE*, 534:22-33, 1985.
- [106] R M Loughheed and D L Mccubhrey. The cytocomputer a practical pipelined image processor. *7th. Int Symposium in Comput Architecture*, pages 271-277, 1980.
- [107] R L Luck. Using PIPE for inspection applications. *Proc. SPIE*, 730:12-19, 1986.

- [108] R L Luck. Implementing an image understanding system architecture using pipe. *SPIE Automated Inspection and High Speed Vision Architectures*, 849:35-41, 1987.
- [109] E Luczak and A Rosenfeld. Distance on a hexagonal grid. *IEEE Trans Comput*, 25:532-533, 1976.
- [110] J R Mandeville. Novel method for analysis of printed circuit images. *IBM J of Res Develop*, 29(1):73-86, 1985.
- [111] P Maragos. A representation theory for morphological image and signal processing. *IEEE Trans PAMI*, 11(6):586-599, 1989.
- [112] M Maresca et al. Parallel architectures for vision. *Proc IEEE*, 76(8):970-981, 1988.
- [113] D Marr. *Vision a Computational Investigation into the Human Representation and Processing of Visual Information*. W H Freeman and Co, San Francisco, 1981.
- [114] D Marr and E Hildreth. Theory of edge detection. *Proc Royal Soc London B207*, pages 187-217, 1980.
- [115] L Masih and T J Stonham. Convergence in a learning network with pattern feedback. *Patr Recog Proc 4th Int Conf. Cambridge, UK. Ed J Kittler. Springer Verlag*, pages 100-109, 1988.
- [116] G Mathern. *Random Sets and Integral Geometry*. Wiley, New York, USA, 1975.
- [117] Matrox Electronic Systems Ltd, Dorval, Quebec, Canada. *Pip Video Digitizer Users Manual*, 1987.
- [118] F F Mazda. *Electronic Engineers Reference Book*. Butterworth and Co Ltd, London, UK, 5th Edition, 1983.

- [119] J D McCafferty et al. Edge detection algorithm and its video rate implementation. *Image and Vision Computing*, 5(2):155-160, 1987.
- [120] J R McClellan. Personal communication. *Allen Brady Inc, Milwaukee, WI, USA*, 1989.
- [121] J R McClellan. Characterization and correction of image acquisition system response for machine vision. *Proc Spie*, 1194:62-75, 1990.
- [122] D L McCubbrey and M Loughheed. Morphological image analysis using a raster pipeline processor. *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, 1985.
- [123] C D McIlroy et al. Hardware for real time image processing. *IEE Proc Part E*, 131(6):223-229, 1984.
- [124] R M Mersereau. The processing of hexagonally sampled two dimensional signals. *Proc IEEE*, 67(6):930-949, 1979.
- [125] B Michaelis. Durchmessermeßgerät mit integriertem mikrorechner. *Bild und ton*, 41(12):362-364, 1988.
- [126] J P Mylopoulos and T Pavlidis. On the topological properties of quantized spaces: II connectivity and order of connectivity. *J of the Assocn for Comput Machinery*, 18(2):247-254, 1971.
- [127] V S Nalwa and T O Binford. On edge detection. *IEEE Trans PAMI*, 8(6):699-714, 1986.
- [128] D Nitzan. Three dimensional vision structure for robot applications. *IEEE Trans PAMI*, 10(3):291-309, 1988.
- [129] M Nixon. Shape inspection of crystal glassware. *IEE 3rd Int Conf Image Process and Applications*, 307:227-231, 1989.

- [130] G R Nudd et al. *The Application of Three Dimensional Microelectronics to Image Analysis*. In *Integrated Technology for Parallel Image Processing*, Ed S Leviadi, Academic Press, London, 1985. pp256-282.
- [131] G R Nudd et al. WPM a multiple simd architecture for image processing. *IEE 3rd Int Conf on Image Proc, Warwick, UK, Publication no 307*, pages 161-165, 1989.
- [132] J H Nurre. Flaw detection of modeled surfaces. *Proc SPIE*, 1004:48-54, 1990.
- [133] J H Nurre and E L Hall. Robot vision overview. *SPIE Digital Image Processing*, 528:216-239, 1985.
- [134] H Nyquist. Certain topics in telegraph transmission theory. *Trans AIEE*, 47:617-644, 1928.
- [135] J P Oakley and M J Cunningham. A function space model for digital image sampling and its application to image reconstruction. *Computer Vision Graphics and Image Processing*, 49:171-197, 1990.
- [136] R Oliver. Principles of the use of radio isotope tracers in clinical and research investigations. *Pergamon Press, Oxford, UK*, 1971.
- [137] T Ozaki et al. An automatic visual inspection system for industrial printing. *SPIE Automated Inspection and Measurement, Ed J W Chen*, 730:194-201, 1986.
- [138] W A Perkins. Inspector, a computer vision system which learns to inspect parts. *IEEE Trans PAMI*, 5:584-592, 1983.
- [139] W A Perkins. Model based inspection of printed circuit boards. *Patt rec*, 17:485-491, 1984.
- [140] V H Perry and A Cowey. The ganglion cell and cone distributions in the monkeys retina: Implications for central magnification factors. *Vision Research*, 25:1795-1810, 1985.

- [141] D P Petersen and D Middleton. Sampling and reconstruction of wave number limited functions in n dimensional euclidean spaces. *Information and Control*, 5:279-323, 1962.
- [142] J Porri, T P Pollard, J B Pridmore, J B Bowen, J E W Mayhew, and J P Frisby. Tina a 3D vision system for pick and place. *Image and Vision Computing*, 6(2):91-99, 1988.
- [143] W K Pratt et al. Pseudomedian filter. *SPIE Architectures and Algorithms for Digital Image Processing* 2, 534:34-43, 1985.
- [144] K Preston. Feature extraction by golay hexagonal pattern transforms. *IEEE Transactions on Computers*, 20(9):1007-1014, 1971.
- [145] K Preston et al. Basics of cellular logic with some applications in medical image processing. *Proc IEEE*, 67(5):826-856, 1979.
- [146] J G Proakis and D G Manolakis. *Introduction to Didital Signal Processing*. Macmillan Publishing Co, New York, USA, 1988.
- [147] R C Reitan. *Personal Communication*. Applied Vision Systems Inc, Minnesota USA, 1989.
- [148] F M Rhodes et al. A monolithic hough transform processor based on restructurable VLSI. *IEEE PAMI*, 10(1):106-110, 1988.
- [149] R W Richardson. Flexible vision control system for precision robotic arc welding. *Proc SPIE*, 1008:13-22, 1989.
- [150] G E Rivard. Direct fast fourier transform of bivariate functions. *IEEE Trans ASSP*, 25:250-252, 1977.
- [151] I N Robinson and W R Moore. A paralcl processor array architecture and its implementation in silicon. *Proc IEEE Conf on Custom Integrated Circuits*. Rochester, NY USA, pages 41-45, 1982.

- [152] A Rosenfeld. Connectivity in digital pictures. *J of Assoc for Comput Machinery*, 17(1):146-160, 1970.
- [153] A Rosenfeld. *Computer Architectures for Machine Vision*. In *Machine Vision Algorithms Architectures and Systems* Ed H Freeman, Academic Press, London, UK, 1988. pages 97-101.
- [154] A Rosenfeld and A C Kak. *Digital Picture Processing*, volume 1. Academic Press, New York, USA, 1982.
- [155] A Rosenfeld and J L Pfaltz. Distance functions on digital pictures. *Patt rec.*, 1:33-61, 1968.
- [156] P A Ruetz and R W Brodersen. A custom chip set for real time image processing. *Conf Iccasp Tokyo*, pages 801-804, 1986.
- [157] J Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.
- [158] J Serra. Introduction to mathematical morphology. *Computer Vision Graphics and Image Processing*, 35:283-305, 1986.
- [159] C E Shannon. Communication in the presence of noise. *Proc IRE*, 37:10-21, 1949.
- [160] L G Shapiro, R M Haralic, and T C Pong. The visual components of an automated inspection task. *IEEE Conference on Artificial Intelligence Applications Denver*, pages 207-10, 1984.
- [161] F Y C Shih and O R Mitchell. Threshold decomposition of gray scale morphology into binary morphology. *IEEE Trans PAMI*, 11(1):31-42, 1989.
- [162] D B Skillicom. A taxonomy for computer architectures. *IEEE Computer*, 21(11):46-57, 1988.
- [163] J Sklansky. On the hough technique for curve detection. *IEEE Trans Comput*, 27(10):923-926, 1978.

- [164] R W Smith. Computer processing of line images a survey. *Pat. rec.*, 20(1):7-15, 1987.
- [165] S R Sternberg. Greyscale morphology. *Computer Vision, Graphics and Image Processing*, 35:333-355, 1986.
- [166] G C Stockman and A K Agrawala. Equivalence of hough curve detection to template matching. *Comms ACM*, 20(11):820-822, 1977.
- [167] Q F Stout. Mapping vision algorithms to parallel architectures. *Proc IEEE*, 76(8):982-995, 1988.
- [168] H Strecker. Automatic xray testing of castings an approach based on local feature operator and flexible image matching. *Proc IEEE 6th Pattern Recognition Conf.*, 1:451-455, 1982.
- [169] B V Suresh et al. A real time automated visual inspection system for hot steel slabs. *IEEE Trans PAMI*, 5:563-572, 1983.
- [170] S Tanimoto and T Pavlidis. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, 4:104-119, 1975.
- [171] S L Tanimoto et al. A prototype pyramid machine for hierarchical cellular logic. In *Parallel Computer Vision*, Ed L Uhr, Academic Press, USA, pages 43-83, 1987.
- [172] R H Thibadeau. Automated visual inspection as skilled perception. *Proc. Soc. Manufacturing Engg. Conf. on Vision, Detroit, USA*, 5:1-19, 1985.
- [173] R D Tillet et al. Image analysis for biological objects. *IEE 3rd Int Conf Image Process and Applications*, 307:207-211, 1989.
- [174] V Torre and T A Poggio. On edge detection. *IEEE Trans PAMI*, 8(2):147-163, 1986.
- [175] M H. Trivedi et al. Robosight: Robotic vision system for inspection and manipulation. *Proc SPIE*, 1008:66-72, 1989.

- [176] R Y Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off the shelf TV cameras and lenses. *IEEE J of Robotics and Automation*, 3(4):323-344, 1987.
- [177] C Tsatsoulis and Fu K S. A computer vision system for assembly inspection. *Proc SPIE*, 521:352-7, 1984.
- [178] M Unser and F Ade. Decision procedure and feature extraction for automated inspection of texture. *Patn Recognition Lett*, 2:202-213, 1984.
- [179] R J Valkenburg and C C Bowman. Kiwivision II a hybrid pipelined multitransputer architecture for machine vision. *Proc SPIE*, 1004:91-96, 1988.
- [180] R J Valkenburg, R Tekiela, C C Bowman, and O J Olsson. Parallel implementation of vision algorithms on a hybrid pipelined multitransputer architecture. *Proc SPIE*, 1197:247-253, 1990.
- [181] F Van der Heyden. Evaluation of edge detector algorithms. *3rd Int Conf on Image Processing, IEE Conf pub 307*, pages 618-622, 1989.
- [182] A M Wallace. Industrial applications of computer vision since 1982. *Proc IEE Part E*, 135(3):117-136, 1988.
- [183] A B Watson. Recursive in place algorithm for the hexagonal orthogonal oriented quadrature image pyramid. *Proc SPIE*, 1099:194-200, 1989.
- [184] A B Watson and A J Ahumada. A hexagonal orthogonal oriented pyramid as a model of image representation in visual cortex. *IEEE Trans BME*, 36(1):97-106, 1989.
- [185] B Wehner. Parallel recirculating pipeline for signal and image processing. *SPIE*, 1058:27-33, 1989.

- [186] J S Weska. A survey of threshold selection techniques. *Computer Graphics and Image Processing*, 7(2):259-265, 1978.
- [187] R N West and W J Stocker. Automatic inspection of cylinder bores. *Metrology and Inspection*, 9(4):9-12, 1977.
- [188] D J Whitehouse and M J Phillips. Sampling in a two-dimensional plane. *J. Physics A, Math. Gen.*, 18:2465-2477, 1985.
- [189] L Yencharis. The 1989 imaging chip survey. *Advanced Imaging*, 4(8):36-41, 1989.
- [190] H K Yuen, J Princen, J Illingworth, and J Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71-77, 1990.

Guidelines specified in the University of Warwick document: *Requirements for the Presentation of Research Theses, Revised 17 October 1989* have been used in the preparation of this thesis.

THE BRITISH LIBRARY
BRITISH THESIS SERVICE

Visual Inspection:

Image Sampling, Algorithms and Architectures.

TITLE

AUTHOR Richard C. Staunton

DEGREE

AWARDING BODY The University of Warwick

DATE 1991.

THESIS
NUMBER

THIS THESIS HAS BEEN MICROFILMED EXACTLY AS RECEIVED

The quality of this reproduction is dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction.

Some pages may have indistinct print, especially if the original papers were poorly produced or if the awarding body sent an inferior copy.

If pages are missing, please contact the awarding body which granted the degree.

Previously copyrighted materials (journal articles, published texts, etc.) are not filmed.

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no information derived from it may be published without the author's prior written consent.

Reproduction of this thesis, other than as permitted under the United Kingdom Copyright Designs and Patents Act 1988, or under specific agreement with the copyright holder, is prohibited.

1	2	3	4	5	6	REDUCTION X	21	
cms						CAMERA	1	
							No. of pages	