Feature Extraction in Edge Detection using Genetic Programming

by

Wenlong Fu

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Statistics and Operations Research.

Victoria University of Wellington
2014

Abstract

Edge detection is important in image processing. Extracting edge features is the main and necessary process in edge detection. Since features in edge detection are implicit, most of the existing edge features only work well on specific images. Using a moving window has a trade-off between noise rejection and localisation accuracy. Genetic Programming (GP) has been widely applied to image processing, and GP has potential for extracting edge features, although there is little work in GP for edge detection. The overall goal of this thesis is to investigate GP for automatic edge feature extraction using different amounts of existing knowledge from only using raw pixel intensities and ground truth to more advanced domain knowledge such as Gaussian filters.

First of all, this thesis conducts an investigation on fundamental low-level edge detector construction with very little prior edge knowledge. Search operators based on a single raw pixel, a block of pixels, and two blocks of pixels are proposed to construct edge detectors. Unlike most existing methods, this GP system automatically searches neighbours and avoids manually predefining a window size. The results show that the evolved edge detectors outperform some existing edge detectors, such as the Sobel edge detector.

Secondly, from the pixel and image views, localisation of detected edges, and observations of GP programs, new fitness functions are suggested in this thesis. It is found that the pixel view is better than the image view to design fitness functions without allowing a distance from predictions to ground truth. However, in terms of edge localisation, the pixel view is worse than the image view to design fitness functions. A new fitness function combining detection accuracy and localisation effectively improves

the performance of evolved edge detectors. When utilising observations of GP programs to construct soft edge maps, two new fitness functions including a restriction on the range of observations are proposed to evolve edge detectors with good soft edge maps on test images.

Thirdly, pixels implicitly selected by the GP system based on full images are analysed. A set of pixels are extracted from the evolved programs and used to construct edge filters. A merge operation is proposed to extract six pixels to construct second-order edge filters. The results show that a rich but compact set of pixels can be extracted from the evolved edge detectors.

Fourthly, GP is utilised to evolve edge detectors based on the Gaussian-based technique. These GP evolved edge detectors are significantly better than the Gaussian gradient and the surround suppression technique. An efficient and effective sampling technique is proposed for evolving Gaussian-based edge detectors. From the results, there are no significant differences between the Gaussian-based edge detectors evolved by a full set of images and by the sampling technique on the training set.

Fifthly, GP is employed to construct features using an existing set of basic features. The distribution of observations of GP programs is estimated. Evolved composite features are proposed using known distribution models to indicate the probability of pixels being discriminated as edge points. It is found that the composite features effectively combine advantages of basic features and can richly indicate edge responses.

Finally, a Bayesian-based GP system is proposed to construct high-level edge features via employing two general algebraic operators and a function developed from a simple Bayesian model. The simple Bayesian model utilises a general multivariate normal density to combine basic features. Experiments show that the GP evolved programs perform better than the simple Bayesian model to obtain composite features.

Overall, this thesis shows that GP has the capability to effectively extract edge features using different degrees of prior knowledge about edges.

Publications Produced

The following fully-referred papers were published during this Ph.D.

1. Wenlong Fu, Mark Johnston, Mengjie Zhang. "Triangular-Distribution-Based Feature Construction Using Genetic Programming for Edge Detection". Proceedings of 2013 IEEE Congress on Evolutionary Computation. IEEE Press. June 20–23, Cancun, Mexico, 2013. pp. 1732–1739.
2. Wenlong Fu, Mark Johnston, Mengjie Zhang. "Genetic Programming for Edge Detection using Multivariate Density". Proceedings of 2013 Genetic and Evolutionary Computation Conference. Amsterdam, The Netherlands. July 06–10, 2013. pp. 917–924. (**Nominated for Best Paper Award**)
3. Wenlong Fu, Mark Johnston, Mengjie Zhang. "Genetic Programming for Automatic Construction of Variant Features in Edge Detection". Proceedings of the 16th European Conference on Applications of Evolutionary Computation (EvoApplications 2013). Lecture Notes in Computer Science. Vol. 7835. Vienna, Austria, April 3–5, 2013. pp. 354–364.
4. Wenlong Fu, Mark Johnston, Mengjie Zhang. "Automatic Construction of Gaussian-Based Edge Detectors Using Genetic Programming". Proceedings of the 16th European Conference on Applications of Evolutionary Computation (EvoApplications 2013). Lecture Notes

- in Computer Science. Vol. 7835. Vienna, Austria, April 3–5, 2013. pp. 365–375. (**Nominated for Best Paper Award**)
5. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Multi-Frequency Transformation for Edge Detection”. Proceedings of the 27th International Conference on Image and Vision Computing New Zealand. ACM Press. Dunedin, New Zealand. 26–28 November 2012.
 6. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Automatic Construction of Invariant Features Using Genetic Programming for Edge Detection”. Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence. Springer. Sydney, Australia, December 2012.
 7. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Figure of Merit Based Fitness Functions in Genetic Programming for Edge Detection”. Proceedings of the Ninth International Conference on Simulated Evolution And Learning (SEAL2012). Springer, Hanoi, Vietnam, December 16–19, 2012. pp 22–31.
 8. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Genetic Programming for Edge Detection Using Blocks to Extract Features”. Proceedings of 2012 Genetic and Evolutionary Computation Conference (GECCO 2012). ACM Press. Philadelphia, USA. 7–11 July 2012. pp. 855–862.
 9. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Genetic Programming for Edge Detection Based on Figure of Merit”. Proceedings of 2012 Genetic and Evolutionary Computation Conference (GECCO (Companion) 2012). ACM Press. Philadelphia, USA. 7–11 July 2012. pp. 1483–1484.
 10. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Genetic Programming for Edge Detection via Balancing Individual Training Images”.

Proceedings of 2012 IEEE Congress on Evolutionary Computation. IEEE Press. 2012. pp. 2597–2604.

11. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Soft Edge Maps From Edge Detectors Evolved by Genetic Programming”. Proceedings of 2012 IEEE Congress on Evolutionary Computation. IEEE Press. 2012. pp. 24–31.
12. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Analysis of Diagonal Derivatives in Edge Detectors Evolved by Genetic Programming”. Proceedings of the Twenty-sixth International Conference on Image and Vision Computing New Zealand. Auckland. 29 Nov–1 Dec 2011. pp. 345–350.
13. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Genetic Programming for Edge Detection Based on Accuracy of Each Training Image”. Proceedings of the 24th Australasian Joint Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence. Vol. 7106. Springer. Perth, Australia, December 5–8, 2011. pp. 301–310.
14. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Hybrid Particle Swarm Optimisation Based on History Information Sharing”. Proceeding of Genetic and Evolutionary Computation Conference (GECCO’11), ACM Press. 2011. pp. 77–84.
15. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Genetic Programming For Edge Detection: A Global Approach”. Proceeding of the 2011 IEEE Congress on Evolutionary Computation. IEEE Press. New Orleans, USA. June 5–8, 2011. pp. 254–261.
16. Wenlong Fu, Mark Johnston, Mengjie Zhang. “A Hybrid Particle Swarm Optimisation with Differential Evolution Approach to Image Segmentation”. Proceedings of the European Conference on Appli-

cations of Evolutionary Computation, Lecture Notes in Computer Science. Vol. 6624 Springer. Torino, Italy 2011. pp. 173–182.

17. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Hybrid Particle Swarm Optimisation Algorithms based on Differential Evolution and Local Search”. Proceedings of the 23rd Australasian Joint Conference on Artificial Intelligence. AI 2010: Advances in Artificial Intelligence. Lecture Notes in Artificial Intelligence. Vol. 6464. Springer. Adelaide, Australia, 2010. pp. 313–322.
18. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Low-level Feature Extraction for Edge Detection using Genetic Programming”. IEEE Transactions on Systems, Man, and Cybernetics (under revision)
19. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Distribution-based Invariant Feature Construction using Genetic Programming for Edge Detection”. Information Systems. (submitted)
20. Wenlong Fu, Mark Johnston, Mengjie Zhang. “New Fitness Functions and New Operators for Automatically Constructing Subjective Edge Detectors Using Genetic Programming”. Applied Soft Computing. (submitted)
21. Wenlong Fu, Mark Johnston, Mengjie Zhang. “Sampling Training Images to Evolve Gaussian-based Edge Detectors Using Genetic Programming”. IEEE Transactions on Systems, Man, and Cybernetics (submitted)

Acknowledgments

I would like to thank my supervisors, Dr Mark Johnston and Prof Mengjie Zhang for their guidance, constant encouragement, valuable suggestions, and constructive feedback in writing this thesis and the published papers. Also, thank you for improving my English speaking and writing skills.

I am grateful for the financial assistance of Prof Mengjie Zhang (Marsden Fund of New Zealand VUW0806) and Victoria University of Wellington (Vice Chancellor's Strategic PhD Scholarship) over the past three years.

Thanks to Bing Xue, Su Nguyen, and the rest of the Evolutionary Computation Research Group for the many interesting discussions.

I also greatly appreciate the support and encouragement of my friends and family.

Terms and Acronyms

ANOVA Analysis of Variance

AUC Area Under Curve

BSD Berkeley Segmentation Dataset

BT Bandpass Transformation

CDF Cumulative Distribution Function

CGP Cartesian Genetic Programming

CI Computational Intelligence

CT Component Tree

DE Differential Evolution

DoG Difference of Gaussian

EC Evolutionary Computation

ES Evolution Strategy

ETHZ Swiss Federal Institute of Technology, Zurich

FOM Figure of Merit

GA Genetic Algorithm

x

GG Gaussian Gradient

GP Genetic Programming

GT Ground Truth

LT Linear Transformation

LoG Laplacian of Gaussian

NT Non-Linear Transformation

PDF Probability Density Function

PSO Particle Swarm Optimisation

ROC Receiver Operating Characteristic

SD Standard Deviation

SS Surround Suppression

SVM Support Vector Machine

Contents

1	Introduction	1
1.1	Motivation	3
1.2	Goals	6
1.3	Major Contributions	7
1.4	Organisation	14
2	Literature Review	17
2.1	Overview of Edge Detection	17
2.1.1	Basic Concepts in Edge detection	18
2.1.2	Taxonomy of Feature Extraction for Edge Detection	20
2.1.3	Other Edge Detection Techniques	31
2.1.4	Performance Evaluation	33
2.2	Overview of Genetic Programming	38
2.2.1	Evolutionary Computation	38
2.2.2	Genetic Programming	40
2.3	Related Work for Edge Detection with GP	44
2.4	Image Datasets	46
2.5	Chapter Summary	50
I	Low-level Edge Feature Construction	51
3	New Search Operators for Low-level Feature Construction	53
3.1	Introduction	53
3.1.1	Chapter Goal	54

3.1.2	Chapter Organisation	55
3.2	GP System Based on Full Images	55
3.2.1	Sets of Terminals and Functions	56
3.2.2	Fitness Function	56
3.3	Search Operators	57
3.3.1	Searching a Single Pixel	57
3.3.2	Searching Single Blocks	60
3.3.3	Searching Double Blocks	64
3.4	Experiment Settings	66
3.4.1	Image Dataset	66
3.4.2	Sets of Terminals and Functions	67
3.4.3	Parameter Settings	67
3.5	Results and Discussions	68
3.5.1	Without <i>IFLess</i>	68
3.5.2	Using <i>IFLess</i>	76
3.5.3	Comparing with Existing Edge Detectors	78
3.5.4	Example Detected Images	80
3.5.5	Summary	85
3.6	Further Discussions	85
3.6.1	Computational Cost	86
3.6.2	Number of Generations	91
3.6.3	Blocks Used by Search Operators	95
3.7	Chapter Summary	97
4	New Fitness Functions for Low-level Feature Construction	101
4.1	Introduction	101
4.1.1	Chapter Goal	104
4.1.2	Chapter Organisation	105
4.2	Evaluations from Pixel and Image Views	105
4.2.1	<i>F</i> -measure	105
4.2.2	FOM	107
4.3	Matching Directions in FOM	108

4.3.1	Fitness Function with a Single Direction Matching . .	109
4.3.2	A Proposed Combination of FOM and F -measure . .	110
4.3.3	Comparisons Among Different FOM Variants	111
4.4	Observation Spread	114
4.4.1	Transformation	114
4.4.2	Fitness Function on Observation Spread	117
4.5	Experiment Settings	119
4.5.1	Image Datasets	119
4.5.2	Sets of Terminals and Functions	121
4.5.3	Fitness Functions	121
4.5.4	Parameter Settings	121
4.6	Results and Discussions	122
4.6.1	Evaluations on Pixel vs Image Views	123
4.6.2	Fitness Functions F vs FOM	134
4.6.3	Testing on the ETHZ Dataset	138
4.6.4	Example Detected Images From F and f_{FOM}	139
4.6.5	Results on Matching Directions in FOM	142
4.6.6	Results on Observation Spread	145
4.7	Further Discussions	163
4.7.1	Pixel vs Image Views	163
4.7.2	F -measure vs FOM	164
4.7.3	Thickness of Detected Edges	165
4.7.4	Parameters in Bandpass Transformation	166
4.8	Chapter Summary	167
5	Pixels Implicitly Selected by GP	169
5.1	Introduction	169
5.1.1	Chapter Goal	170
5.1.2	Chapter Organisation	171
5.2	Analysing Pixels Selected by GP	171
5.2.1	Settings for Evolving Edge Detectors	171
5.2.2	Methods for Extracting Pixels	172

5.2.3	Filters Constructed from Selected Pixels	175
5.3	Experiment Settings	178
5.3.1	Pixels Used to Construct Filters	178
5.3.2	Baselines for the Comparisons	178
5.4	Results and Discussions	179
5.4.1	Images Detected by GP	179
5.4.2	Optimised Linear Filters	181
5.4.3	Estimated Parameters for a Second-Order Filter	186
5.5	Further Discussions	188
5.5.1	Objective Functions F and $f_{F_{CR}}$	188
5.5.2	Linear and Second-order Filters	189
5.5.3	Number of Pixels Used to Construct Edge Detectors .	189
5.5.4	Pixels Selected by GP	192
5.5.5	Computational Cost	193
5.6	Chapter Summary	193

II Gaussian-based Edge Feature Construction 195

6	Gaussian-based Feature Construction	197
6.1	Introduction	197
6.1.1	Chapter Goal	198
6.1.2	Chapter Organisation	199
6.2	Gaussian-based GP System	199
6.2.1	Terminals Based on Gaussian Models	200
6.2.2	Function Set	200
6.2.3	Fitness Function	202
6.3	Experiment Settings	202
6.3.1	Training Data	203
6.3.2	Parameter Settings	205
6.4	Results and Discussions	206
6.4.1	A Single Image as Training Data	206

6.4.2	Two Images as Training Data	208
6.4.3	Sampling	212
6.4.4	GP Edge Detectors vs Existing Gaussian-based Edge Detectors	214
6.4.5	Detected Images from the Best GP Edge Detectors . .	216
6.4.6	Example Evolved GP Gaussian-based Edge Detector	219
6.5	Further Discussions	220
6.5.1	Computational Cost	220
6.5.2	Training Images	221
6.5.3	Thickness of Detected Edges	225
6.5.4	Contributions on Different Gaussian Filters	225
6.5.5	Combination Function $\mathbb{C}$	228
6.6	Chapter Summary	230

III Composite Edge Feature Construction 233

7	Distribution-based Feature Construction	235
7.1	Introduction	235
7.1.1	Chapter Goal	237
7.1.2	Chapter Organisation	239
7.2	Background of Invariant Features	239
7.3	Distribution-based GP for Feature Construction	243
7.3.1	Terminal Set	243
7.3.2	Primitive Functions	244
7.3.3	Composite Features	244
7.3.4	Fitness Function	248
7.4	Experiment Design	249
7.4.1	Training Data	249
7.4.2	Estimated Distribution	249
7.4.3	Parameter Settings	251
7.4.4	Test Performance	251

7.5	Results and Discussions	252
7.5.1	PDF and CDF	252
7.5.2	$GP_{PDF, Norm}$ vs Existing Methods	256
7.5.3	Comparison Among GP, SVM and the Bayesian Model	257
7.5.4	Non-parametric Statistics on the Results from GP, SVM and the Bayesian Model	258
7.5.5	Recall and Precision for Best Evolved Programs . . .	260
7.5.6	Qualitative Evaluation on Detected Images	261
7.5.7	Example Evolved Program	264
7.6	Variant Feature Construction	266
7.6.1	Training Data on Horizontal Edges	266
7.6.2	Results From Variant GP Programs	267
7.6.3	Detected Images	269
7.7	Further Discussions	271
7.7.1	Fixed Threshold vs Estimated Distribution	271
7.7.2	Suppression of Wrong Responses	274
7.7.3	Influence of the Range of Observations of Programs .	275
7.7.4	Thickness of Edges	277
7.7.5	Construction Equations GP_{pdf} and GP_{cdf}	278
7.8	Chapter Summary	280
8	High-level Feature Construction	283
8.1	Introduction	283
8.1.1	Chapter Goal	284
8.1.2	Chapter Organisation	284
8.2	Bayesian-based GP System	285
8.2.1	Bayesian-based Function and Terminal	285
8.2.2	The Other Terminals	287
8.2.3	Fitness Function	288
8.3	Experiment Settings	288
8.4	Results and Discussions	290
8.4.1	Overall Results From $Fit_{t=3}$	290

8.4.2	$Set_{bf,rand}$ using $Fit_{t=3}$ vs Existing Techniques	294
8.4.3	Fitness Functions $Fit_{t=3}$ vs $Fit_{t=30}$	297
8.4.4	Detected Images	299
8.4.5	Example Evolved Bayesian-based Program	300
8.5	Further Discussions	303
8.5.1	Influence of General Algebraic Operators	304
8.5.2	Computational Cost	305
8.5.3	Convergence	307
8.6	Chapter Summary	308
9	Conclusions and Future Work	309
9.1	Achieved Objectives	309
9.2	Main Conclusions	311
9.2.1	Low-level Edge Feature Construction	311
9.2.2	Gaussian-based Edge Feature Construction	314
9.2.3	Composite Edge Feature Construction	315
9.3	Limitations	318
9.4	Future Work	319

List of Figures

1.1	Example of edge features of an image from the Gaussian gradient [28], histogram gradient [123] and normalised standard deviation [55].	2
1.2	Major contribution chapters of this thesis.	14
1.3	Connection of the Major contribution chapters of this thesis.	16
2.1	General edge detection flow.	18
2.2	Different edge types in one dimensional description.	19
2.3	Two example edges in 7×7 window from a two-dimensional graylevel image.	19
2.4	A 5×5 window extracted from an image I at the centre position.	20
2.5	General masks with 3×3 window for gradient edge operators.	22
2.6	Results for an image detected by Canny and LoG.	24
2.7	General algorithm flow for EC	39
2.8	One binary chromosome with eight bits in GA.	39
2.9	An example of the expression " $x * x + 3x$ " in the tree-based GP.	41
2.10	Example for the crossover operation in GP.	42
2.11	Example for the mutation operation in GP.	43
2.12	Three example training images from BSD dataset and their ground truth.	47
2.13	Three example test images from BSD dataset and their ground truth.	48
2.14	Three example images from ETHZ and their ground truth.	49

3.1	Example two-dimensional intensity matrix and its result after calling $s_{-1,0}$.	59
3.2	The 2×2 Robert detector constructed with GP. Nodes " $\sqrt{\quad}$ " and "SQ" are functions <i>sqr</i> t and <i>square</i> , respectively.	60
3.3	Two example blocks of pixels specified by search operator $block_{t,l,w,d}$.	62
3.4	Example of searching neighbours by $block_{t,l,w,d}$.	63
3.5	Examples of using two blocks of pixels.	65
3.6	Example training image 23080 and its sampled result of size 125×125 pixels.	66
3.7	Boxplots ($f_{0.5}$) for evolved edge detectors using different settings and without <i>IFLess</i> .	73
3.8	Three example BSD test images and their ground truth (GT).	80
3.9	Three example BSD test images detected by the Sobel edge detector and the Gaussian Gradient (GG).	81
3.10	Three example test images detected by the best edge detectors from $Set_{5 \times 5}$ and Set_s , respectively.	82
3.11	Three example test images detected by the best edge detectors from Set_b and Set_{tb} , respectively.	83
3.12	Three example test images detected by the best edge detectors from $Set_{s,b}$, $Set_{s,tb}$, and $Set_{s,b,tb}$, respectively.	84
3.13	Occurrences of combinations ($w \times l$ in the evolved edge detectors).	96
4.1	Detection accuracies $f_{i,0.5}$ values for four different images of three detectors.	107
4.2	The same FOM value with different detection results.	109
4.3	Three different example detected results in a 7×7 window.	112
4.4	Different edge maps for detected edges in a BSD test image by the Sobel detector.	115
4.5	Different observation spreads for true edge points and non-edge points.	116

4.6	Some example subimages from 20 training images in BSD and the corresponding ground truth.	120
4.7	Scatter plot for detection results $f_{i,0.5}$ (mean of 30 GP edge detector in each fitness function for each image) based on F with Set_s	125
4.8	Scatter plot for detection results $f_{i,0.5}$ (all 30 GP edge detectors) based on F with Set_s	126
4.9	Scatter plot for detection results f_{FOM_i} (all 30 GP edge detectors) based on F with Set_s	127
4.10	Scatter plot for detection results $f_{i,0.5}$ (mean of 30 GP edge detectors) based on F with $Set_{s,b}$	129
4.11	Scatter plot for detection results f_{FOM_i} (mean of 30 replications) based on f_{FOM} with Set_s	132
4.12	Scatter plot for detection results f_{FOM_i} (mean of 30 replications) based on f_{FOM} with $Set_{s,b}$	134
4.13	Three example BSD images detected by the best edge detectors from F and f_{FOM} with $Set_{s,b}$	140
4.14	Three example ETHZ images detected by the best edge detectors from F and f_{FOM} with $Set_{s,b}$	141
4.15	Three example BSD test images detected by the best edge detector from $f_{FOM,gt}$ and $f_{FOM,0.5}$ with Set_s , respectively. . .	144
4.16	Three example BSD test images detected by the best edge detector from F and f_{FOM} with Set_s , respectively.	145
4.17	Boxplots for test performance F and $f_{F_{avg}}$ of the evolved edge detectors (with threshold 0) from fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s}	147
4.18	Boxplots for F_{max} and $F_{max,avg}$ of the GP evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using LT	151
4.19	Precision and recall for the GP evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using LT	152

4.20	Example soft edge maps on image 78004 from the best (the first row) and worst (the second row) edge detectors from F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using LT	153
4.21	Boxplots for F_{max} and $F_{max,avg}$ of the evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using NT	155
4.22	Precision and recall of the evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using NT	156
4.23	Example soft edge maps on image 78004 after the best (the first row) and worst (the second row) edge detectors using NT	158
4.24	Boxplots for $F_{max,avg}$ and F_{max} of the evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using BT	161
4.25	Precision and recall of the evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using BT	162
4.26	An example soft edge maps on image 78004 after the best (the first row) and worst (the second row) edge detectors using BT	163
4.27	Scatter plots of recall and precision values of average, the best and the worse of the evolved edge detectors from F using $c_{bt_{rg}} = 100$	167
4.28	An example soft edge maps on image 78004 after the best and worst edge detectors using BT width $c_{bt_{rg}} = 100$	167
5.1	Process of finding pixels to construct edge filters.	172
5.2	Six training images from BSD dataset.	173
5.3	Neighbours (at least 15 occurrences) selected from 30 GP edge detectors. Note that numbers are the frequency of occurrences of the neighbours.	174
5.4	Example of the merging operation on a simple GP tree.	175
5.5	Pixels (at least 15 occurrences) selected from 30 GP edge detectors with a merging operation. Note that numbers are the occurrences for the neighbours.	176

5.6	Four example test images from BSD dataset detected by the best GP, Sobel and Laplacian (“Lap”) edge detectors.	180
5.7	Test performance (F) averaged over the 100 BSD test images for linear filters estimated by PSO-DE based on objective functions F and f_{FCR}	182
5.8	Test performance (F) averaged over the 100 BSD test images for second-order filters estimated by PSO-DE based on F and f_{FCR}	184
5.9	Four example detected images by a second-order filter constructed with Set_{nb2}	187
5.10	The convergence for the estimated second-order filters with fitness function F in the training stage.	191
6.1	Six example training images from the BSD dataset and their ground truth.	204
6.2	Five example detected BSD test images by Gaussian gradient (GG) and surround suppression (SS).	216
6.3	Five example BSD test images detected by the GP Gaussian-based edge detectors.	217
6.4	One example GP Gaussian-based edge detector.	219
6.5	Detection performance on images 42078 and 106020, and C4 from Gaussian gradients (GG).	223
6.6	Two example images detected by a Gaussian-based edge detector before and after thinning operations [106].	226
7.1	One example image with different responses on different edges.	237
7.2	Detected image based on Gaussian gradients T_{gg} , histogram gradients T_{hg} and normalised standard deviations T_{sd}	242
7.3	Probabilities for edge points and non-edge points on the observations of an example edge detector.	247

7.4	Histograms of observations of a GP program from each setting ($GP_{PDF, Norm}$, $GP_{PDF, Trial}$, $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$) on the training data.	253
7.5	F_{max} values from $GP_{PDF, Norm}$, $GP_{PDF, Trial}$, $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$ on the 100 BSD test images.	255
7.6	Recall and precision of the best composite features from $GP_{PDF, Norm}$, $GP_{PDF, Trial}$, $GP_{CDF, Norm}$, $GP_{CDF, Unif}$ and SVM, and the Bayesian Model on the 100 BSD Test Images.	260
7.7	Four example images detected by T_{gg} , T_{sd} , T_{hg}	262
7.8	Four example images detected by $GP_{PDF, Norm}$, $GP_{PDF, Trial}$, $GP_{CDF, Norm}$, $GP_{CDF, Unif}$, the Bayesian model, and the linear SVM.	263
7.9	Details for recall and precision of T_{gd} , T_f , T_{hd} and GP (average).	269
7.10	Four example detected images based on the different variant features.	270
7.11	Two example images detected by an evolved program directly using threshold 0 to classify pixels.	273
7.12	Two example images detected by evolved programs from $GP_{PDF, Trial}$ with fitness functions Fit and Fit_{RS}	277
7.13	Two example detected images from GP with CDF and GP_{pdf}	280
8.1	Boxplots for the F_{max} values of the results (30 replications) from the six settings using fitness function $Fit_{t=3}$ on the 100 BSD test images.	291
8.2	F_{max} values for the features constructed by the evolved 30 programs from GP using $Set_{bf, rand}$ and $Fit_{t=3}$	295
8.3	Recall and precision for T_{gg} , T_{sd} , T_{hg} , x_{bayes} , and an evolved program.	296
8.4	Two example images detected by T_{gg} , T_{hg} , T_{sd} , x_{bayes} , and the best evolved programs from GP with $Set_{bf, rand}$ using $Fit_{t=3}$ and $Fit_{t=30}$	299

8.5	Image 69020 detected by an evolved program from $Set_{bf,rand}$ with $Fit_{t=3}$ and the combination x_{bayes}	301
8.6	Two example images detected by two parts of the best evolved method from GP.	302
8.7	Recall and precision for an evolved program, its two parts and T_{hg}	303
8.8	Two example images detected by two evolved programs from Set_{bt} using $Fit_{t=30}$	305
8.9	Means of the best fitness at each generation in each setting. .	307

List of Tables

2.1	Outputs of binary classification problems with confusion matrix.	35
3.1	Search operators in the function set.	67
3.2	Test performance (mean $\pm$ standard deviation) of the GP edge detectors without <i>IFLess</i> on the 100 BSD test images. Note that the time is for testing one BSD image.	69
3.3	Recall and precision of the GP edge detectors without <i>IFLess</i> on the 100 BSD test images.	71
3.4	Statistical p -values (two-samples t -tests) among constructed GP edge detectors from $Set_{5 \times 5}$, Set_s , Set_b , Set_{tb} , $Set_{s,b}$, $Set_{s,tb}$, $Set_{s,b,tb}$ on the 100 BSD test images.	74
3.5	Non-parametric statistical comparison (Kruskal-Wallis statistic [174]) among constructed GP edge detectors from $Set_{5 \times 5}$, Set_s , Set_b , Set_{tb} , $Set_{s,b}$, $Set_{s,tb}$, $Set_{s,b,tb}$ on the 100 BSD test images.	76
3.6	Test performance (mean $\pm$ standard deviation) of GP edge detectors using <i>IFLess</i> on the 100 BSD test images.	77
3.7	Comparison (p -values using one-sample t -tests) among the evolved edge detectors without <i>IFLess</i> , the Sobel edge detector, and the Gaussian gradient.	79
3.8	Approximate means of training time (hours) for GP evolving edge detectors.	86
3.9	Occurrences of the functions and terminals in the evolved edge detectors without <i>IFLess</i>	87

3.10	Occurrences of the functions and terminals in the evolved edge detectors using <i>IFLess</i>	90
3.11	Test performance (mean $\pm$ standard deviation) of GP edge detectors using the maximum generation 120 and without <i>IFLess</i> on the 100 BSD test images.	92
3.12	Occurrences of the functions and terminals in the evolved edge detectors without <i>IFLess</i> and maximum generation 120.	93
3.13	Total occurrences of approaches used in blocks (in the 30 evolved edge detectors without <i>IFLess</i>).	97
4.1	Measurements for three example detected edges by f_{FOM} , $f_{FOM,gt}$, $f_{FOM,\alpha}$ and $f_{FOM,\delta}$	113
4.2	Fitness functions used in this chapter.	122
4.3	Means and standard deviations of test performance F , $f_{F_{avg}}$ and $f_{F_{gm}}$ values of GP edge detectors constructed by Set_s using F variants for the BSD 100 test images.	123
4.4	Comparisons among GP edge detectors constructed by Set_s using F , $f_{F_{avg}}$, $f_{F_{gm}}$, $f_{F_{qf}}$ and $f_{F_{qm}}$ for the BSD 100 test images.	124
4.5	Means and standard deviations of test performance F , $f_{F_{avg}}$ and $f_{F_{gm}}$ values of GP edge detectors constructed by $Set_{s,b}$ using F variants for the BSD 100 test images.	128
4.6	Comparisons among GP edge detectors constructed by $Set_{s,b}$ using F , $f_{F_{avg}}$, $f_{F_{gm}}$, $f_{F_{qf}}$ and $f_{F_{qm}}$ for the BSD 100 test images.	129
4.7	Means and standard deviations of test performance F and f_{FOM} values of GP edge detectors constructed by Set_s using FOM variants for the BSD 100 test images.	130
4.8	Comparisons among GP edge detectors constructed by Set_s using f_{FOM} , $f_{FOM_{avg}}$, $f_{FOM_{gm}}$ and $f_{FOM_{qm}}$ for the BSD 100 test images.	131

4.9	Means and standard deviations of F and f_{FOM} values of GP edge detectors constructed by $Set_{s,b}$ using FOM variants for the BSD 100 test images.	133
4.10	Comparisons among GP edge detectors constructed by $Set_{s,b}$ using f_{FOM} , $f_{FOM_{avg}}$, $f_{FOM_{gm}}$ and $f_{FOM_{qm}}$ for the BSD 100 test images.	133
4.11	Comparisons between GP edge detectors and the Sobel edge detector, for the BSD 100 test images.	135
4.12	Test performance F after using one-to-one assignment on the GP edge detectors constructed by $Set_{s,b}$ with fitness function F or f_{FOM} for the BSD 100 test images.	137
4.13	Comparisons among GP edge detectors constructed by $Set_{s,b}$, the Sobel edge detector, the Marr edge detector [122] and the Canny edge detector for BSD 100 test images after one-to-one assignment.	137
4.14	Test performance (F) for detection results on the ETHZ dataset.	138
4.15	Comparisons among GP edge detectors constructed by $Set_{s,b}$ from fitness functions F and f_{FOM} , the Canny edge detector, the Berkeley and Component Tree (CT) boundary detectors for the ETHZ image dataset.	139
4.16	Test performance on different matching directions (compared with standard FOM).	143
4.17	Test performance (F and $f_{F_{avg}}$) on fixed threshold 0 for fitness functions $f_{F_{cr}}$, f_{F_d} and f_{F_s}	146
4.18	Kruskal-Wallis statistic [174] of pairs (test performance F and $f_{F_{avg}}$) on the results of the GP edge detectors with threshold 0.	148
4.19	Test performance $F_{max,avg}$ and F_{max} of the GP evolved edge detectors after using LT	149

4.20	Kruskal-Wallis statistic [174] of pairs (F_{max} and $F_{max,avg}$) on the soft edge maps of the GP evolved edge detectors after using LT	151
4.21	Test performance F_{max} and $F_{max,avg}$ of the evolved edge detectors after using NT	154
4.22	Kruskal-Wallis statistic [174] of pairs ($F_{max,avg}$ and F_{max}) on the soft edge maps of the GP evolved edge detectors after using NT	156
4.23	Test performance F_{max} and $F_{max,avg}$ of the GP evolved edge detectors after using BT	160
4.24	Kruskal-Wallis statistic [174] of pairs ($F_{max,avg}$ and F_{max}) on the soft edge maps of the GP evolved edge detectors after using BT	161
5.1	Test performance (F) on 100 BSD test images for linear filters estimated by PSO based on objective functions F and $f_{F_{CR}}$	182
5.2	Comparisons ($F, f_{F_{CR}}$) among linear filters, GP edge detectors (only from fitness function F), the Laplacian detector, and the Sobel detector.	183
5.3	Test performance (F) on 100 BSD test images for second-order filters estimated by PSO based on F and $f_{F_{CR}}$	184
5.4	Comparisons ($F, f_{F_{CR}}$) among second-order filters, GP edge detectors (only from fitness function F), the Laplacian detector, and the Sobel detector.	185
5.5	Estimated parameters for an example second-order filter.	187
6.1	Test performance F values and means of recall and precision of GP Gaussian-based edge detectors from each of the six images as training data respectively.	207

6.2	One-way ANOVA (row vs column) for GP Gaussian-based edge detectors from the six images as training data respectively.	207
6.3	Test performance F values and means of recall and precision of GP edge detectors from two images as training data respectively.	209
6.4	One-way ANOVA (row vs column) for GP edge detectors from two images and single image as training data.	210
6.5	One-way ANOVA (row vs column) for GP edge detectors from two images as training data respectively.	211
6.6	F values and means of recall and precision of GP edge detectors from sampling techniques on 20 images.	212
6.7	One-way ANOVA (row vs column) for GP Gaussian-based edge detectors from sampling techniques on 20 images.	213
6.8	One-way ANOVA (row vs column) comparing F values from some GP edge detectors with F_{max} values from Gaussian gradients (GG) and surround suppression (SS) on the BSD test images.	215
6.9	Results (F values) of evolved edge detectors from single type of filter without combination function $\mathbb{C}$ (with training image 23025) on the BSD test images.	227
6.10	One-way ANOVA (row vs column) comparisons (F) among single Gaussian filters used in GP on the BSD test images.	227
6.11	Results (F values) of evolved edge detectors by using single filters and combination function $\mathbb{C}$ (with training image 23025) on the BSD test images.	228
6.12	One-way ANOVA (row vs column) comparisons (F) among single Gaussian filters used in GP (including function $\mathbb{C}$) on the BSD test images.	230

7.1	Comparison of means and ($\pm$) standard deviations F_{max} values among constructed features by GP with PDF and CDF for the BSD test images.	254
7.2	Comparison of F_{max} values among constructed features by GP with PDF (Gaussian distribution), image Gaussian gradients T_{gg} , normalised standard deviations T_{sd} , histogram gradients T_{hg} , Sobel edge detector, a Bayesian model [40], and SVM [27] for the BSD test images.	256
7.3	Statistical p -values (t -tests) for the comparisons among GP based on PDFs and CDFs, SVM and the Bayesian model for the BSD test images.	257
7.4	Statistical p -values (with Mann-Whitney-Wilcoxon tests [102]) for the comparisons among GP based on PDFs and CDFs, SVM and the Bayesian model for the BSD test images.	259
7.5	Kruskal-Wallis statistic [174] of F_{max} values among constructed features by GP with PDFs and CDFs on the BSD test images.	259
7.6	Comparison of F_{max} values among constructed features by GP, image Gaussian derivatives T_{gd} , F -tests T_f , histogram derivatives T_{hd} , and a Bayesian model for the 100 BSD test images.	268
8.1	Settings for terminals and functions in the Bayesian-based GP system.	289
8.2	Test performance F_{max} for the six settings using fitness function $Fit_{t=3}$ on the BSD test image dataset.	291
8.3	Comparison between the results from $Set_{bf,rand}$ using $Fit_{t=3}$, image Gaussian gradients T_{gg} , normalised standard deviations T_{sd} , histogram gradients T_{hg} , Sobel edge detector and the simple Bayesian terminal x_{bayes} on the 100 BSD test images.	294
8.4	Test performance F_{max} for the six settings using $Fit_{t=30}$ on the BSD test image dataset.	297

8.5	Training time (means $\pm$ standard deviations in seconds) from $Fit_{t=3}$ and $Fit_{t=30}$ using the six settings.	306
-----	---	-----

Chapter 1

Introduction

Automatic interpretation of the content of an image is a significant goal in computer vision and image analysis. Over the last five decades, interpretation of image contents by computers has received a lot of attention [7, 8, 10, 50, 80, 114, 152, 173, 194]. In order to extract information from an object, an essential job is to distinguish the object from its background. **Edge detection** is a process of detecting discontinuities in the photometrical, geometrical and physical characteristics of regions (or segments) in an image [8, 152, 194]. In general, the edges extracted from an image can be simplified to describe the image content. Edge detection benefits a wide range of applications, such as image compression [166], image enhancement [103], object recognition [137], object tracking [175], and image retrieval [90].

In edge detection, there are typically three stages: pre-processing, feature extraction, and post-processing [140]. Since pre-processing and post-processing techniques can be easily and effectively applied to different edge detection algorithms, feature extraction is considered as a major component in edge detection [131]. **Features** in edge detection are functions of raw pixel values in an image relative to a local point and are used to discriminate pixels as edge points or non-edge points. A feature can be discrete (binary) or continuous (numeric). A binary feature uses one value

Figure 1.1: Example of edge features of an image from the Gaussian gradient [28], histogram gradient [123] and normalised standard deviation [55].

to represent an edge point and the other value to represent a non-edge point. A continuous feature is usually used to discriminate edge points by threshold techniques [28]. In general, a continuous feature is represented by a soft edge map. In grayscale levels, high grayscale levels (strong responses) are normally indicated by bright points. Usually, a “low-level” edge feature is directly extracted from raw pixels and a “high-level” edge feature is constructed from a set of basic low-level features using a combination method.

The notion of edges is the result of common human experience rather than a formal mathematical definition [124, 140]. Therefore, edge detection is a subjective task and the desired detection result for one image is dependent on human observations [123]. Many different edge features have been used to detect edges, and all these edge features have both strengths and weaknesses [140]. Figure 1.1 shows three different edge features applied to an image from the Berkeley Image Dataset (BSD) [123]. The three edge features are the Gaussian gradient [28], histogram gradient [123] and

normalised standard deviation [55]. Figure 1.1 (b) has problems in area 1 where it gives inappropriately strong responses on non-edge points and weak responses on edge points, area 2 where there is a lot of noise interference, and area 3 where edge points are not detected. Figure 1.1 (c) shows better responses on the edges in area 1 than Figure 1.1 (b), but has a problem in area 2 of noise interference, and only includes a few edge points in area 3 with many false alarms. Figure 1.1 (d) also has a problem in area 1 where it gives too weak responses to the edge points, but is only weakly affected by noise in area 2, and includes partial edges in area 3. In order to obtain suitable edge features for desired edge outputs, a method to extract good edge features efficiently and effectively is required.

To satisfy different tasks based on desired edge maps, automatic feature extraction techniques should be employed. **Genetic Programming** (GP) is an Evolutionary Computation algorithm that has been applied to many fields with considerable success [95]. GP can evolve programs to solve or perform a given task without knowing the details of *how* to achieve it and these programs are often human competitive [96]. GP has been employed to design or approximate edge detectors [41, 62, 71, 150]. Most of the existing methods consider a GP edge detector as a black box, aiming at obtaining edge maps based on different requirements. To understand how to efficiently and effectively use GP in edge detection, analysis of edge detectors evolved by GP is required. The existing work in GP has very limited contribution on edge feature extraction. Since feature extraction is a very important step in edge detection, it is necessary to further develop GP for evolving new edge features, with ground truth as desired outputs, to improve detection performance.

1.1 Motivation

The subjective nature of edge detection (without unique solutions) requires flexible methods to extract edge features to satisfy different edge detection

tasks. An edge detector designed by a human usually only works under some given conditions, such as step edges without texture or noise in images [140]. Based on the known conditions, formulae/models are generally designed by image processing experts. However, designing such formulae is very time consuming, and the designed specific formulae do not generalise well on other (different) tasks. Since edge features are implicit, automatic feature extraction approaches using a computer are much cheaper than human design, and they have potential to work on different edge detection tasks under good design.

In edge detection, machine learning algorithms can be used to combine existing basic features into composite features [123]. A composite feature can be learned from ground truth. However, there are two problems for most machine learning algorithms. Firstly, in most machine learning algorithms, the fixed basic input variables (neighbours of discriminated pixels or existing basic features) must be given. It is challenging to construct edge features for a machine learning algorithm when only the original images and their ground truth are known without any other prior knowledge. Secondly, in many machine learning algorithms, a fixed model typically needs to be given in advance. A fixed model offers only a limited solution space within which new composite features can be generated.

To address the problem of only using original images and their ground truth, GP is a good candidate technique since it can automatically and implicitly search pixels (neighbours) to construct edge features. Although GP has been used to search neighbours to construct edge detectors typically in single images, the evolved edge detectors had limited ability to extract edge features of natural images [150]. How to efficiently and effectively search pixels still needs to be investigated. Since GP has flexible descriptions for constructing features, the solution space for constructed features is naturally larger than the range of possible combinations from most other common machine learning algorithms.

Since most of the existing edge features only work well on some sim-

ple images, automatic edge feature extraction techniques need to be developed for complex natural images. For instance, the Gaussian gradient works well on simple images without textures, but does not work well on texture images. The task of automatic feature construction might use different degrees of prior knowledge. However, when GP employs different degrees of prior knowledge to automatically extract edge features, the following issues should be addressed.

1. When only a set of natural images and their ground truth (as very little prior knowledge) are given, how to effectively construct edge features needs to be addressed. Since the prior knowledge only gives a set of full images, the intensity value of a single pixel is not sufficient to discriminate that pixel as an edge point or a non-edge point. The relationship between a pixel and its neighbours needs to be found. When GP is employed to evolve edge features, it is necessary to design efficient and effective *search operators* for searching suitable neighbours of each discriminated pixel.
2. When evaluating evolved features, different tasks require different suitable criteria. When an edge detection task requires evolved edge detectors with high detection accuracy on a test image dataset across all pixels (from pixel view), whether these evolved edge detectors have a high detection accuracy on each test image (from the image view) is not clear, and vice versa. In other words, how to suitably evaluate the goodness of edge detectors created by GP should be addressed. In order to effectively obtain good edge detectors evolved by GP for different tasks, it is necessary to develop new *fitness functions*.
3. Whether the edge detectors evolved by GP can be reasonably explained is another issue that should be examined. In order to further understand how GP evolves edge detectors, the evolved edge detectors should be analysed. For example, whether *pixels* implicitly

selected by GP are good for construction of edge detectors should be investigated.

4. When prior knowledge from Gaussian-based edge detection is used, how to efficiently and effectively utilise this prior knowledge in GP to evolve edge detectors should be addressed. Gaussian-based edge detection has problems of how to combine different types of Gaussian filters and how to set filter parameters [14]. In order to address the problems in Gaussian-based edge detection, it is necessary to develop and test a *Gaussian-based GP system*.
5. When basic features extracted by a set of given techniques (as prior knowledge) are available, how to efficiently and effectively combine these basic features needs to be investigated. When a set of basic edge features is combined into *composite features* by GP, whether the composite features can combine the strengths of the basic features and reduce the weaknesses of the basic features should be investigated. Also, how to use the composite features evolved by GP to suitably indicate edge responses should be addressed.
6. When prior knowledge involves a particular combination technique (to combine basic features), GP could utilise this technique (of combining basic features, not extracting basic features) to construct *high-level features*. For example, when a Bayesian model [40] is utilised in GP for combining basic features as high-level features, whether the Bayesian model is used to evolve effective high-level features should be investigated.

1.2 Goals

The overall goal of this thesis is to develop a new GP approach to **automatic feature extraction** in edge detection incorporating different degrees

of prior domain knowledge. To achieve this goal, this investigation is conducted based on the following six research objectives.

1. Develop a new method for automatic construction of low-level edge detectors based on *single pixels* and *blocks of pixels* when only a set images and their ground truth are provided.
2. Investigate new fitness functions from the *pixel view* and the *full image view*, incorporating accuracy, localisation of detected edges, and the range of the observations of evolved programs.
3. Analyse the pixels implicitly selected by GP via extracting a suitable set of neighbours of discriminated pixels from GP evolved low-level edge detectors to construct edge filters.
4. Develop a new GP system to efficiently and effectively construct Gaussian-based edge features.
5. Develop a new GP system to construct composite features using a set of basic features in order to combine the strengths of these basic features, reduce the weaknesses of these basic features, and indicate different edge responses by probabilistic techniques.
6. Develop a new GP system incorporating a Bayesian model to construct high-level (higher than in objective 5) edge features using a set of basic features.

1.3 Major Contributions

This thesis investigates a range of GP methods for extracting edge features when different degrees of prior knowledge are given. The prior knowledge ranges from very little prior domain knowledge on edge feature extraction, through to the Gaussian-based edge detection technique, different edge feature extraction techniques, and a Bayesian technique (combin-

ing basic features). The major contributions of this thesis are summarised as follows.

1. This thesis shows how GP can be used to develop low-level edge detectors based on *full images* via using search operators to automatically select raw pixels as inputs, rather than commonly using predefined local regions (windows) in the literature. This thesis proposes a GP method directly using a set of full images to evolve edge detectors. This method evolves low-level edge detectors and compares the evolved edge detectors with some methods using windows in the literature on a benchmark natural image dataset, namely the BSD image dataset [123]. The edge detectors evolved by GP with the proposed search operators have higher detection accuracy and faster detection speed than the edge detectors evolved by GP with windows. Also, the evolved edge detectors based on full images have higher detection accuracy than some existing methods, such as the Sobel edge detector [57] and the Gaussian gradient [14]. The search operators proposed in the GP method can be used to evolve edge detectors with fast detection speed and some texture suppression ability. Part of this work was published as follows.
 - (1) Wenlong Fu, Mark Johnston, Mengjie Zhang. “Genetic Programming for Edge Detection: a Global Approach”. Proceedings of the 2011 IEEE Congress on Evolutionary Computation. IEEE Press. New Orleans, USA. June 5–8, 2011. pp. 254–261.
 - (2) Wenlong Fu, Mark Johnston, Mengjie Zhang. “Genetic Programming for Edge Detection Using Blocks to Extract Features”. Proceedings of the 2012 Genetic and Evolutionary Computation Conference. ACM Press. Philadelphia, USA. 7–11 July 2012. pp. 855–862.
2. This thesis shows how to evolve good low-level edge detectors based on the evaluation criteria using the accuracies of all pixels and the

accuracies of individual full images, localisation, and the range of observations of programs, rather than considering the detection accuracy of all pixels only. Different evaluation criteria are used to propose *new fitness functions* in the GP method based on full images. These new fitness functions test evolved low-level edge detectors and compare these evolved edge detectors over the BSD image dataset. It is found that for the fitness function based on detection accuracy, it is better to perform the evaluation from the pixel view than from the full image view. However, for the fitness function based on localisation, it is better for making the evaluation from the full image view than from the pixel view. If we consider maximising the accuracy over individual full images, the corresponding accuracy over all pixels (across all images) may be worse. A proposed fitness function (combining detection accuracy and localisation of detected edges) improves the test performance of the evolved edge detectors, in terms of detection accuracy and localisation. Also, two proposed fitness functions based on the range of observations of programs are used to evolve edge detectors to obtain soft edge maps. Three different transformation approaches are introduced to transform observations of evolved edge detectors to grayscale levels. In order to obtain rich and suitable contrast of soft edges, a restriction on the range of observations of programs is suggested. Additionally, another popular image dataset, namely the ETHZ dataset [47] is used to show the generalisation ability of the evolved edge detectors. Part of this work was published as follows.

- (3) Wenlong Fu, Mark Johnston, Mengjie Zhang. "Genetic Programming for Edge Detection Based on Accuracy of Each Training Image". Proceedings of the 24th Australasian Joint Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence. Vol. 7106. Springer. Perth, Australia, December 5–8, 2011. pp. 301–310.

- (4) Wenlong Fu, Mark Johnston, Mengjie Zhang. "Soft Edge Maps From Edge Detectors Evolved by Genetic Programming". Proceedings of the 2012 IEEE Congress on Evolutionary Computation. IEEE Press. 2012. pp. 24–31.
 - (5) Wenlong Fu, Mark Johnston, Mengjie Zhang. "Genetic Programming for Edge Detection via Balancing Individual Training Images". Proceedings of the 2012 IEEE Congress on Evolutionary Computation. IEEE Press. 2012. pp. 2597–2604.
 - (6) Wenlong Fu, Mark Johnston, Mengjie Zhang. "Genetic Programming for Edge Detection Based on Figure of Merit". Proceedings of the 2012 Genetic and Evolutionary Computation Conference (GECCO (Companion) 2012). ACM Press. Philadelphia, USA. 7–11 July 2012. pp. 1483–1484.
 - (7) Wenlong Fu, Mark Johnston, Mengjie Zhang. "Figure of Merit Based Fitness Functions in Genetic Programming for Edge Detection". Proceedings of the Ninth International Conference on Simulated Evolution and Learning. Springer, Hanoi, Vietnam, December 16–19, 2012. pp 22–31.
3. This thesis shows that valuable information can be extracted from the evolved low-level edge detectors by analysing low-level edge detectors. Neighbours of a discriminated pixel, which are extracted from GP evolved edge detectors by two proposed selection methods, are not uniformly located around that discriminated pixel. However, existing methods from the literature normally employ all neighbours of a discriminated pixel from a fixed window. In order to validate whether the neighbours implicitly selected by GP are sufficient to find edges, an optimisation based technique is proposed to use raw pixels to construct edge filters. This method tests the resulting edge filters which use the extracted neighbours, and compares those constructed edge filters with the edge filters constructed using only

neighbours from a fixed window. The edge filters constructed by GP extracted neighbours have higher detection accuracy than the edge filters constructed by the fixed windows. This thesis shows that pixels extracted from GP low-level edge detectors are rich but compact to construct edge filters. Part of this work was published as follows.

(8) Wenlong Fu, Mark Johnston, Mengjie Zhang. “Analysis of Diagonal Derivatives in Edge Detectors Evolved by Genetic Programming”. Proceedings of the Twenty-sixth International Conference on Image and Vision Computing New Zealand. Auckland. 29 Nov–1 Dec 2011. pp. 345–350.

4. This thesis shows how GP can be used to develop *Gaussian-based edge detectors* by using self-tuning Gaussian filter parameters and automatically combining different types of Gaussian filters, rather than empirically setting parameters and combining Gaussian filters as in the existing literature. This thesis proposes a GP method directly using a set of full natural images to evolve Gaussian-based edge detectors. This method tests the evolved Gaussian-based edge detectors and compares the evolved edge detectors with two Gaussian-based methods in the literature on the BSD image dataset. The Gaussian-based edge detectors evolved by GP have higher detection accuracy than the Gaussian gradient and the surround suppression method [66]. An efficient and effective sampling technique is proposed to reduce computational cost for evolving edge detectors, and the detection accuracy of the evolved edge detectors is kept. The evolved Gaussian-based edge detectors are analysed, and it is suggested that a good combination is to choose different types of Gaussian filters to achieve high detection performance. Part of this work was published as follows.

(9) Wenlong Fu, Mark Johnston, Mengjie Zhang. “Automatic Construction of Gaussian-Based Edge Detectors Using Genetic Pro-

gramming". Proceedings of the 16th European Conference on Applications of Evolutionary Computation. Lecture Notes in Computer Science. Vol. 7835. Vienna, Austria, April 3-5, 2013. pp. 365–375. (Nominated for Best Paper Award)

5. This thesis shows how GP can be used to effectively construct *composite features* from a set of basic features. This thesis proposes a GP method with estimated distributions to effectively construct composite features with rich edge responses. Rather than using a combination from a fixed model as in the existing literature, this GP method automatically combines basic features. Instead of directly using observations of evolved programs as edge responses, this GP method utilises estimated distributions of observations of programs to indicate edge responses. This method tests the evolved composite (invariant and variant) features and compares the evolved features with all employed basic features and two combination techniques in the existing literature on the BSD image dataset. The evolved features have higher detection accuracy and richer edge responses than the combination from a Bayesian model [40] and the combination from a linear Support Vector Machine [27]. Part of this work was published as follows.
 - (10) Wenlong Fu, Mark Johnston, Mengjie Zhang. "Automatic Construction of Invariant Features Using Genetic Programming for Edge Detection". Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence. Springer. Sydney, Australia, December 2012. pp 144–155.
 - (11) Wenlong Fu, Mark Johnston, Mengjie Zhang. "Genetic Programming for Automatic Construction of Variant Features in Edge Detection". Proceedings of the 16th European Conference on Applications of Evolutionary Computation. Lecture Notes

in Computer Science. Vol. 7835. Vienna, Austria, April 3–5, 2013. pp. 354–364.

- (12) Wenlong Fu, Mark Johnston, Mengjie Zhang. “Triangular-Distribution-Based Feature Construction Using Genetic Programming for Edge Detection”. Proceedings of the 2013 IEEE Congress on Evolutionary Computation. IEEE Press. June 20–23, Cancun, Mexico, 2013. pp. 1732–1739.

6. This thesis shows how GP can be used to develop *Bayesian-based programs* to construct high-level features via utilising a Bayesian model to select and combine basic features, rather than directly combining all basic features in the Bayesian model. This thesis proposes a Bayesian-based GP method utilising the Bayesian model. This method tests the evolved high-level features and compares the high-level features with all employed basic features and the combination of the Bayesian model on the BSD image dataset. The evolved high-level features have higher detection accuracy than the basic features and the combination of the Bayesian model. The evolved Bayesian-based programs are analysed, and it is suggested that the Bayesian technique works quite well with GP. Also, it shows that GP could further develop the Bayesian technique, such as selecting features and designing model structures. Part of this work was published as follows.

- (13) Wenlong Fu, Mark Johnston, Mengjie Zhang. “Genetic Programming for Edge Detection using Multivariate Density”. Proceedings of the 2013 Genetic and Evolutionary Computation Conference. Amsterdam, The Netherlands. July 6–10, 2013. pp. 917–924. (Nominated for Best Paper Award)

Figure 1.2: Major contribution chapters of this thesis.

1.4 Organisation

The main contribution chapters of this thesis are Chapters 3 to 8. Each of thesis chapters correspondingly addresses a major research goal from Section 1.2.

From the degrees of prior knowledge given in an edge detection task, these chapters range from very little prior knowledge to rich prior knowledge. Figure 1.2 shows the connection among these chapters. Chapters 3, 4, and 5 investigate low-level feature extraction with very little prior knowledge. Chapter 6 investigates evolving Gaussian-based edge detectors. Chapters 7 and 8 investigate combining basic features when different prior knowledge is given. The outline of this thesis is as follows.

- *Chapter 2: Literature Review*

This chapter reviews popular edge detection techniques and existing work in GP for edge detection. Their strengths and limitations are discussed. Also, two benchmark image datasets for edge detection are described.

- *Chapter 3: New Search Operators for Low-level Feature Construction*

This chapter proposes three search operators to select pixels to construct low-level features. A GP system is developed based on full images with very little prior knowledge. The evolved edge detectors from different search operators are analysed to further understand how GP evolves effective low-level edge features.

- *Chapter 4: New Fitness Functions for Low-level Feature Construction*

The fitness function in the GP system (in Chapter 3) is investigated further. New fitness functions based on the pixel and image views, accuracy, localisation of detected edges, and range of the observations of evolved programs are proposed and tested.

- *Chapter 5: Pixels Implicitly Selected by GP*

This chapter analyses pixels implicitly selected by the GP system in Chapters 3 and 4. Pixels extracted from GP evolved low-level edge detectors are used to construct edge filters.

- *Chapter 6: Gaussian-based Feature Construction*

A Gaussian-based GP system is developed to effectively utilise the Gaussian-based technique to evolve edge detectors. The influence of using single images and combinations of images, respectively, as the training data is investigated.

- *Chapter 7: Distribution-based Feature Construction*

A GP system is developed to construct invariant and variant (to image rotation) composite features when a set of basic features is given. In contrast to Chapter 6, more prior knowledge (not only the Gaussian-based technique, but also other edge feature extraction techniques) is provided for further development. Composite GP features based on estimated distributions are investigated for indicating different edge responses.

Figure 1.3: Connection of the Major contribution chapters of this thesis.

- *Chapter 8: High-level Feature Construction*

This chapter proposes a Bayesian-based GP system to evolve programs to construct high-level features. A simple Bayesian model is introduced to construct programs for combining basic features. Different from Chapter 7, the Bayesian technique of combining basic features is provided for constructing composite features. The development of the Bayesian technique is addressed in this chapter.

- *Chapter 9: Conclusions and Future Work*

The overall conclusion is drawn in this chapter. Future research directions are also suggested in this chapter.

Figure 1.3 shows the connection of the major contribution chapters of this thesis. The GP system based on full images employs the proposed search operators in Chapter 3. Chapter 4 investigates different evaluation criteria in the GP system based on full images. The programs evolved by the GP system are analysed in Chapter 5. After the investigation of GP evolved low-level edge detectors, Chapter 6 introduces Gaussian-based edge detection technique in the GP system. Chapter 7 further investigates the combinations of different feature extraction techniques in GP by using common mathematical functions. Instead of using common mathematical functions, Chapter 8 develops a Bayesian-based combination technique in GP for combining a set of predefined features.

Chapter 2

Literature Review

This chapter begins with background of edge detection, mainly including methods for extracting edge features and the performance evaluation for edge detection. The limitations of existing extraction methods for edge detection are discussed, which supports the motivation of this thesis. Also, background of evolutionary computation and genetic programming is introduced. It is assumed that the reader is familiar with basic concepts in image processing, such as filters and convolution operators as described in [152].

2.1 Overview of Edge Detection

Edge detection usually contains three stages: pre-processing, feature extraction and post-processing [131, 140]. Figure 2.1 shows a general edge detection process flow. For an image I , an intermediate result I' will be obtained after pre-processing. The pre-processing stage mainly focuses on noise reduction and texture suppression while preserving edges and not blurring boundaries between different areas [140]. The feature extraction stage is divided into two phases, namely response computation and feature manipulation. In the response phase, the computation can come from gradients [28] or statistics [111], and a set of features R is obtained.

Figure 2.1: General edge detection flow.

In the feature manipulation phase, feature selection [38] and further feature construction [66, 123] are included, and the output is a set of features M . The feature extraction stage is the main and necessary stage in edge detection [14, 140]. The purpose of extracting features is to use them to classify pixels as edge points or non-edge points. After post-processing, a final edge map B is obtained. Post-processing mainly focuses on marking edge points, thinning edges [106], removing stand-alone edge points and linking broken edge points [140, 152].

In general, a normal pre-processing technique, such as Gaussian filtering [14], or a popular post-processing technique, such as thinning operations [106], can generally collaborate with different feature extraction methods [140]. This survey mainly covers the stage of extracting features.

2.1.1 Basic Concepts in Edge detection

This section describes the concept of edge type, and discusses features for edges.

Edge Types

Features for marking pixels in images as edge points or non-edge points are typically extracted based on edge types [152, 140]. Assuming that an edge exists between two different regions, the type of the edge can be modelled as ramp edge, stair edge or texture edge. These edge types commonly consider both regions to be wide enough. When one region is very narrow, the special (edge) type is called line edge. Figure 2.2 shows the four edge types in a one-dimensional model. The top of Figure 2.2

Figure 2.2: Different edge types in one dimensional description.

Figure 2.3: Two example edges in 7×7 window from a two-dimensional graylevel image.

(a) shows an ideal step edge, which is considered as a special case of the ramp edge at the bottom. When more than two regions meet together, a junction edge is used to describe this situation. However, most features are extracted based on two different regions, and a junction edge is considered as a simple combination of edges and separated as different boundaries.

Figure 2.3 shows two example edges for an ideal step edge and a normal ramp edge in a 7×7 window. The values in the window are grayscale levels. Figure 2.3 (a) describes an ideal step edge at the middle column of the window (or the column to the right of the middle column). In general, the location of the ramp edge in Figure 2.3 (b) is the second column or the sixth column, but some mark the middle column with grayscale level 30 as the true edge.

$I_{-2,-2}$	$I_{-1,-2}$	$I_{0,-2}$	$I_{1,-1}$	$I_{2,-2}$
$I_{-2,-1}$	$I_{-1,-1}$	$I_{0,-1}$	$I_{1,-1}$	$I_{2,-1}$
$I_{-2,0}$	$I_{-1,0}$	$I_{0,0}$	$I_{1,0}$	$I_{2,0}$
$I_{-2,1}$	$I_{-1,1}$	$I_{0,1}$	$I_{1,1}$	$I_{2,1}$
$I_{-2,2}$	$I_{-1,2}$	$I_{0,2}$	$I_{1,2}$	$I_{2,2}$

Figure 2.4: A 5×5 window extracted from an image I at the centre position.

Features and Edges

Edge features are normally employed to mark pixels as edge points when their values are larger than a threshold. Figure 2.4 describes a 5×5 window extracted from an image I relative to the centre position $I_{0,0}$, where the subscript indices (x, y) of $I_{x,y}$ are the relative positions around $I_{0,0}$. Generally, features are extracted from a local area. To obtain a feature value for the pixel $I_{0,0}$, its neighbours, such as $I_{0,-1}$, $I_{0,1}$, $I_{0,2}$, etc., are used in calculations, such as the mean of the intensities of these pixels.

From the image rotation point of view, features are categorised as invariant features and variant features. Variant features are used to extract directional edges. For instance, in order to extract a vertical step edge (Figure 2.2(a)), the neighbours $I_{x,y}$ ($x < 0$, see Figure 2.4) are considered as one area, and the neighbours $I_{x,y}$ ($x > 0$) are considered as the other area. If the difference of the intensities of both areas ($x < 0$ and $x > 0$) (as an edge feature) is large than a threshold, $I_{0,0}$ will be marked as an edge point. Variant features are affected by image rotation, but the invariant features are not affected by image rotation.

2.1.2 Taxonomy of Feature Extraction for Edge Detection

In early research on edge detection, there is very little knowledge about edges incorporated into the methods, and the main work was on step

edges [50, 57]. The early work mainly focused on low-level feature extraction (based on pixels) via investigating step edges. Considering noise in images, Gaussian-based techniques [28, 14] have been developed to extract edges. To date, there have been many approaches to extracting edge features [140]. This subsection describes differentiation [57], statistics [24], mathematical morphology [157], machine learning and optimisation [63], phase congruency and local energy [132], multiresolution [162], and a combination of local features [123].

Differentiation

High gradient magnitudes of pixels are often used to indicate discontinuities of the input luminance profile, such as step edges [140]. The first-order derivative and the second-order derivative have been used to extract edge features [152]. A fixed window is used to approximate the derivative of the intensity value of a discriminated pixel. First-order derivative edge detection fundamentally contains methods based on two orthogonal directions or a set of directional derivatives. Methods using two orthogonal directions includes the Sobel detector, the Prewitt detector [153], and others [118, 152]. For calculating derivatives, weight masks are simply used for computation. Two general masks with a 3×3 window are used to approximate pixel (intensity) gradient by edge detectors with two orthogonal directions. Figure 2.5 shows the two masks, where $a = 2$ for the 3×3 Sobel detector, $a = 1$ for the Prewitt detector, and $a = \sqrt{2}$ for the Frei-Chen edge detector [49]. To calculate the response of the horizontal derivative R_x in Figure 2.5, Equation (2.1) gives the definition, where $\otimes$ is the convolution operator.

$$R_x = \begin{bmatrix} 1 & 0 & -1 \\ a & 0 & -a \\ 1 & 0 & -1 \end{bmatrix} \otimes I \quad (2.1)$$

However, these edge detectors using first-order derivatives have limited power for removing noise. In order to remove noise, Gaussian filters

1	0	-1	-1	-a	-1
a	0	-a	0	0	0
1	0	-1	1	a	1

(a) Mask for R_x (b) Mask for R_y

Figure 2.5: General masks with 3×3 window for gradient edge operators.

have been integrated into masks by weighting functions [5, 119]. Canny further studied Gaussian filters and proposed the well-known Canny detector [28]. The Canny detector [28] is derived from an optimal filter based on the local maxima resulting from the convolution of a filter with the signal affected by white noise in one dimension. The optimal filter is approximated by the derivative of a Gaussian function $g_\sigma(x)$ with scale parameter σ (see Equation (2.2)) [14]. Canny edge detectors use adaptive hysteresis thresholding (double thresholds to filter noise and connect edge points with weak responses to edge points with strong responses) to eliminate breaking of edge contours, but they are sensitive to weak edges and susceptible to spurious and unstable boundaries with non-significant change in intensity [14, 140].

$$\frac{\partial g_\sigma(x)}{\partial x} \approx -\frac{x}{\sigma^2} \exp\left(-\frac{x^2}{\sigma^2}\right) \quad (2.2)$$

Other approaches compute gradients in several directions by convolution of an image with a set of template masks. Prewitt suggested eight gain-normalised compass gradient masks [152]. Some multiple direction gradient detectors with 45° difference are the Kirsh detector [88] and Robinson detector [158].

Second-order differentiation can be used to accentuate edges. Some researchers have indicated that significant changes (such as intensities) in the second-order derivative occur at edges [152]. Invariant and variant (to image rotation) second-order differentiation are employed to detect edges.

One invariant method is a set of Laplacian detectors. In the 3×3 Laplacian detector R_{lap} (Equation (2.3)), the differentiation is between the centre pixel and its four neighbours in different directions (0° , 90° , 180° , 270°). Zero crossing of the second-order derivative in the gradient direction indicates the presence of edges [68].

$$R_{lap} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \otimes I \quad (2.3)$$

In order to smooth images, Gaussian filters are also utilised for the second-order derivative detector. Marr and Hildrith [122] have proposed the Laplacian of Gaussian (LoG) detector in which a Gaussian filter is used to smooth images. The Difference of Gaussians (DoG) is a second-order derivative filter, approximating LoG well [14]. In second-order directional derivative techniques, after the edge direction is estimated, the one-dimensional second-order derivative relative to the estimated edge direction is calculated. Details about comparison of accuracy of the second-order directional derivative can be found in [182]. More details about Gaussian-based edge detection will be described in Chapter 6. A common computational framework for the derivative detectors via using different parameters in a fixed window has been presented by Ganesan and Bhattacharyya [57]. The framework suggests that a set of parameters or a general equation (outer product of two basis operators) could be generally employed to calculate derivatives for differentiation-based edge detectors.

Figure 2.6 shows one image from the Berkeley Segmentation Dataset (BSD) [124] detected by the Canny detector and the LoG detector. Here a low threshold is used. The Canny detector has connected edges at the boundary of the hill reflection, but the LoG detector has broken edges on the reflection.

The computation of the derivatives of a digital image is an ill-posed problem [17] because there are no unique solutions for derivatives (different numerical approximations of the discrete to the continuous). To

Figure 2.6: Results for an image detected by Canny and LoG.

compute derivatives of a pixel based on “continuous” areas, a function was employed to fit a local area around the pixel, then the derivative is calculated from the fitting function [133, 178, 179, 186]. Lalignant and Truchetet [105], considering edge detection as a signal processing problem, proposed a non-linear derivative schema by splitting a “signal” into a positive signal and a negative signal. The non-linear derivative schema indicated that the detected results had good localisation. Boolean derivatives have been introduced to identify edge localisation in binary images [1].

A limitation in edge detectors based on differentiation is that they cannot distinguish between texture edges on one hand and region boundaries and object contours on the other hand [140]. These edge detectors also have difficulty to detect edges in low contrast area [128, 145]. However, edge detectors based on differentiation are fast to extract edge features.

Statistics

Unlike differentiation, statistical edge detection is data driven (processed by data) to extract edge features. Typically, all pixels in a local area are

divided into two groups based on an assumed edge through the centre pixel in the area. The difference between the two groups are used to discriminate the centre pixel as an edge point or a non-edge point. Several *two*-sample techniques have been developed for finding edges. Non-parametric statistical techniques for extracting edge features include a linear rank test [24], the Wilcoxon test [111], the Kolmogorov-Smirnov test [111], the Mann-Whitney test and the squared ranks test for variances [59], and other rank tests [24, 110]. Also, the likelihood ratio [78], χ^2 [33], and the *t*-test [111] have been employed to extract edge features. Taking the *t*-test as an example, the *p*-value can be used for comparing the intensities of the pixels in the divided two groups. A low *p*-value indicates the pixel as an edge point. Other approaches focus on the distribution of the gradient on a neighbourhood around each pixel. Statistical analysis of the co-occurrence matrix (occurrences of one graylevel to another graylevel with an offset in an image) [142, 168], covariance matrix (of gradients) [3], and joint probability density of neighbouring pixels (with intensities) [181] have been used to detect contours and lines. Wavelet coefficients correspond to hidden states in wavelet-domain hidden Markov models, and they are utilised for edge detection [34, 176].

Statistical approaches based on a fixed neighbourhood are better than differentiation-based methods for detecting textures edges, such as using the co-occurrence matrix [142] to extract edge features. Statistical approaches can detect contours and lines, but they need large computation times and their performance does not significantly outperform the other edge detectors [140].

Mathematical Morphology

Morphological image processing aims to modify the spatial form or structure of objects in an image [169]. The three operations of dilation, erosion and thinning are normally used in edge detection. With dilation, an object grows uniformly in spatial extent; with erosion, an object shrinks

uniformly; and with thinning, an object transfers to a simplified topological equivalence. Morphological gradient uses the difference between the maximum and the minimum value of the image on a fixed neighbourhood [157]. The opening top-hat transformation and closing top-hat transformation have also been utilised to detect edges [113, 199]. The opening top-hat transformation ($TH_c(I)$) of image I is shown in Equation (2.4), where $I \circ se$ means the opening operation and se is the structuring element. A pseudo top-hat mathematical morphological approach has been proposed to detect and preserve thin edges in dark areas [31]. The top-hat transformation has been used to enhance details in the presence of shading [199].

$$TH_c(I) = I - (I \circ se) \quad (2.4)$$

The morphological detectors, by means of computing value ranks, are generally more computationally expensive than methods based on differentiation. The operations opening and closing are similar to filters using statistical ranking techniques, but are usually faster than detectors based on statistics and not robust to the outliers.

Phase Congruency and Local Energy

Methods based on the Fourier transform or other transforms in a fixed window mainly look at phase and local energy. Oppenheim and Lim [138] indicated the importance of phase for human perception of images. Morone and Owens [132] introduced phase congruency and showed that the maxima of the energy function with quadrature pairs occurs at points of maximum phase congruency. Here, a quadrature pair is a set of two linear operators with the same amplitude response but phase responses shifted by 90 degrees. For instance, a combination of two filters, which are different directional outputs of phase in the Gabor filter, is formed as a quadrature pair. Instead of the analytic signal, the quantity of local energy is computed. Quadrature pair filters, such as Gabor, log-Gabor, Gaussian and Cauchy functions, have also been studied [22]. Kovessi [94] extended the

theory behind the calculation of phase congruency in a number of ways. Kovessi addressed the localisation of features, and indicated that the choice of scale in high-pass filtering only affected the related significance of features without degrading their localisation by the proposed approaches.

Xiao and Hou employed symmetry phase congruency to detect the features (in Mach bands and in sinusoidal grating) which cannot simply find the local maxima [187]. Mach bands are perceived by the human visual system because of an illusion of stripes next to the boundary between dark and light regions of an image. Also, a multi-direction shear transform addressed the edge information captured from different directions [189].

Approaches based on phase congruency and local energy are appropriate for step and impulse discontinuities as well as for ramps and other luminance profiles [147]. Some can predict features, such as Mach bands. However, local energy methods perform similarly to the faster and conceptually simpler differentiation-based methods in natural images [123].

Machine Learning and Optimisation

Using only weak knowledge about edges, machine learning methods classify each individual pixel by extracting features from a fixed neighbourhood. Fuzzy classifiers have been utilized to find the maximum hesitation degree of different templates [29, 109] based on a fixed window. A Support Vector Machine (SVM) used a 3×3 window as input to classify the centre pixel as an edge point or a non-edge point [63]. A cellular neural network with a 3×3 window was optimised by a differential evolution algorithm for edge detection [13]. Ten edge patterns based on a 6×6 window were classified by a fuzzy neural network with adaptive fuzzification on patterns [117]. The parameters of a 3×3 filter for designing edge detectors were optimised by Particle Swarm Optimisation (PSO) [2] and Genetic Algorithms [108]. For other optimisation based methods, SVMs were utilised to approximate derivatives based on the least mean square [200], and the Canny detector was approximated by GP [41] and a Neural Network [101]

based on a small window.

Methods based on machine learning can detect edges without the limitation of edge types but are dependent on the training datasets. These methods can be adaptive to different image datasets, but their performances are affected by the way used to extract features and the training data. Ground truth is required for most machine learning methods and the training time is long because an image dataset generally is very large.

Multiresolution

In general, fine scale detection is affected by noise, and coarse scale detection can be tolerant to noise in a range but with some loss of detail [14]. Using different scales to extract edge features aims to find fine edges and avoid noise. Multi-scale methods are generally based on Gaussian filters. The basic premise of using multiple Gaussian filters (multi-scale edge detectors) is that different subregions of an image have varying noise and edge types; therefore a special filter is used to smooth a relevant subregion of the image. Schunck [162] chose the width of the smallest Gaussian filter, and then utilised two different filters whose window sizes are related to the parameter scale. How to set the number of filters was not discussed and too larger scales for the smallest filter may lose important details [14]. Lindeberg [112] proposed obtaining a local scale from a generalization of the non-maxima suppression performed in the scale space, but a new parameter γ related to edge strength at different scales was needed.

There are three common directions for using the multi-scale technique. The first direction proceeds from a coarse solution to a fine solution, namely edge focusing [16]. In this method, a large scale (high σ) Gaussian filter is used to detect edges, and then the next smaller scale is used to find the locations of edges. The Sobel operators and coarse-to-fine edge tracking were introduced in [116]. The Gaussian filters with different scales are employed to smooth images, and the gradients for the smoothed images are extracted by the Sobel edge detector. Multi-scale Gaussian filters are used

to reverse the effect of the blurring caused by large scale Gaussian filters. However, how to set the scale at each level and choose the threshold at each level is hard. The threshold at the coarsest level determines the detected edge quality. The second direction is from fine to coarse [104]. When coarse solutions are used to detect edges, the localisation error needs to be addressed. Again, how to choose scales is not clear. The third direction is to use adaptive Gaussian filters to detect edges [15]. Assuming that noise follows a Gaussian distribution with a known variance, this method smooths areas using a large scale to filter out noise. For real images, the noise variance has to be estimated.

Wavelet theory provides a general framework for multiresolution analysis [35, 120]. In particular, the multi-scale wavelet representation avoids the problem of finding an appropriate scale, and normally produces an improved detection of the Canny detector. However, isotropic (e.g., circularly symmetric) Gaussian filters have poor angular accuracy and have difficulty in distinguishing edges close together [201]. A family of steerable and scalable anisotropic Gaussian filters replaces isotropic Gaussian filters for addressing these problems [58, 146, 191].

In multiresolution approaches, methods from a coarse solution to a fine solution can achieve both noise rejection and good edge localisation [16, 116], but the complex topological structure with different scale filters makes these methods computationally unattractive [139]. How to effectively tune scales of (Gaussian) filters and combine (Gaussian) filters in multiresolution approaches needs to be addressed.

Combination of Local Features

Different feature extraction methods based on fixed neighbourhoods provide different features. Based on different features extracted from a fixed window, Martin, Fowlkes and Malik [123] used density estimation, classification trees, logistic regression, hierarchical mixtures of experts [83] and SVM as classifiers, and showed that these methods have similar accuracy

but different time for training and evaluating the relevant models. The logistic regression model is suggested as the preferred choice because it requires less time for training and evaluating. Features can be extracted by different filters, such as Netzberg filter, in which a set of orientation-tuned pairs of symmetric (even) and antisymmetric (odd) filters are combined as a joint distribution to evaluate pixels on or off an edge [93]. A Boosted Edge Learning algorithm [38] using approximately 50000 features for natural images only has similar evaluation performance to a contour detector proposed in [123] with nine local features.

Biological knowledge has been employed to combine gradients for edge detection. To improve detection of object contours and region boundaries in natural scenes, surround suppression, inspired by the mechanism of non-classical receptive field inhibition, has been proposed [65, 66]. Surround suppression is used to suppress edges that are due to texture. In surround suppression, an operation, called *inhibition*, is used to suppress texture responses. One proposed approach used computation of the gradient at different resolutions, followed by Bayesian denoising of the edge image, and then a surround inhibition step was applied [139]. In general, the response from a DoG is used in the inhibited term, and the response on the gradient of a Gaussian filter is the inhibited context. In surround suppression, different filters have been used, such as Gabor filters [66] and steerable filters [141].

Summary

Features for edge detection are normally extracted from fixed windows. The early research of edge feature extraction mainly focused on the differentiation of intensity values of raw pixels. Including some limited knowledge about edges, different techniques have been applied to extract edge features over the recent two decades, such as Gaussian-based filtering, statistics, and machine learning. Since fixed windows are employed and the window size is a trade-off between noise rejection and localisation,

there are still problems of how to set appropriate window sizes to extract features.

2.1.3 Other Edge Detection Techniques

In general, features are extracted for each pixel via moving fixed size windows. In reality, edge points on edges are connected. So a combination of the values of one feature from a set of connected pixels could be utilised to detect these pixels as edge points together. Also, since different edges exist in an image, a way of extracting edge features might utilise different local windows.

Methods based on Groups of Edge Points

Based on the connection of curves, some approaches look for partial continuous curves (sets of connected pixels). These approaches are often modelled as minimum cost optimisation problems [154]. Active contours [86] have been used to evolve a curve by the minimisation of an energy function, which normally contains an external part and an internal part. The external part concerns the gradient (extracted based on the selected pixels). The internal part addresses the smoothness of the curve (selected pixels). Various cost functions have been proposed, including geometrical, probabilistic and psychophysical models [140]. A detector for detecting multiple face contours adaptively estimated mean intensities for each separated region and used a single curve to capture multiple regions with different intensities based on the curve evolution approach [77]. Recently, the active contour algorithm as a post-processing technique was used to group edges for edge detection [184]. Particle Swarm Optimisation has been used to find optimal short curves [163]. Algorithms for finding short curves mainly focus on a local region, which are different from active contour methods based on a full image.

Methods based on minimum cost optimisation often need a large amount

of computation to search for the optima. Active contour approaches are good for searching object boundaries but face the problem of initialisation and local optima caused by noise [140]. Methods for searching short curves in a local region may lose the global information and may not work well for texture images.

Adaptive neighbourhood processing

Adaptive neighbourhood processing aims at applying different features for different edge points. From the view of a single pixel, a feature discriminates it as an edge point, but another feature might discriminate it as a non-edge point.

The idea of using an adaptive neighbourhood based on a single pixel was proposed by Gordon and Rangayyan [64]. For contrast enhancement, optima will be chosen based on results from different size windows. An adaptive window was used for filtering noise but preserving edges [172]. To adaptively choose the window size, a signal activity index was introduced [172], which reflects the degree of local roughness of the signal within the window. When the signal activity index is sufficiently large, the window size should be reduced to suppress the blurring neighbourhood effect. If the index is smaller than a threshold, the window size will be increased to suppress noise.

Adaptive morphology [20, 21, 121] is another method for adaptive neighbourhood processing. Adaptive structuring elements (e.g., variant size and different shapes) [20, 21, 121] might be used to detect edges, but they have been seldom used for detecting edges of natural images.

Context for neighbourhood processing considers different information (such as gradient) around a discriminated pixel. A gradient-adjusted prediction method [193] calculated a pixel's gradient using the current context of the pixel. The context includes information from patterns such as sharp horizontal edge, sharp vertical edge, smooth area, horizontal edge, weak horizontal edge, vertical edge and weak vertical edge.

Generally, adaptive neighbourhood processing [36, 37], combining abstract algebra and adaptive structuring elements or structuring elements, has been suggested for image processing, such as image enhancement. Although generally adaptive neighbourhood processing can work for edge detection, experiments and comparison with other methods need to be done.

These existing methods based on adaptive neighbourhood processing choose different settings to extract features, which can suppress the noise and improve the localisation accuracy. However, these settings are based on fixed assumed edge information and may not extract features which do not exist in the assumed information. Another problem is how to define criteria to adaptively change ways of extracting different edges.

2.1.4 Performance Evaluation

Evaluating the performance of edge detectors is difficult because of the subjective characteristic of an edge [115]. There is no generic and accurate model describing edges in an image. Even the ground truth is subjective since it is combined from several human observers. Canny [28] suggested three important general principles for error measurement: good detection, good localisation and single response. Good detection means that a detector should have low probability of failing to detect an edge and incorrectly mark a pixel (not on an edge) as an edge point. Good localisation means that a detected edge point should be as close as possible to the true edge point. Single response requires that only one side of a boundary is labelled. In practice, if using a non-maximum suppression [28] or morphological thinning operation [106], the problem of thick responses is easily solved [115].

Evaluation from human observers is dependent on the objects of interest to the observers; it is expensive and cannot be performed automatically. Since the qualitative measures usually involve interaction with humans,

this thesis mainly focuses on the quantitative measures, assigning one or more numerical values to edge maps. Various evaluation methods from quantitative measures exist [115]. They have been developed for detection accuracy (statistics), edge localisation and cost computation (such as active contours [86]). Methods from cost computation do not need ground truth [69, 89], but they usually make use of a larger amount of information from the images [115], such as image derivatives [86] and the regularity in the internal and external regions of the contour [30]. Since they are heavily computational and only focus on how good a result looks (such as a minimum cost), and the details of detected edges are not indicated, these methods are employed only for certain stages [115]. Since it is hard to directly evaluate a detector, ground truth is popular to use for detector performance evaluation [69, 123].

Ground truth is the desired output for a given input image. Ground truth can be obtained from artificial images [69], natural images using observer evaluation [123] or a combination of different detectors [46]. Natural images with ground truth have datasets only using a single observer, such as the Sowerby dataset [93], and using multiple observers, such as the Berkeley Segmentation dataset [123]. Since ground truth comes from human observations, it is subjective. Normally, the quantitative measures for edge detection are objective. Therefore, the quantitative measures using ground truth are a good compromise between completely subjective performance evaluation approaches and completely objective performance evaluation approaches [140].

Probability

From the detection view, probability is utilised to evaluate edge detectors. Edge detection can be considered as a binary classification problem and the counts of the outputs can be described as in Table 2.1. To evaluate a detector for an image I , N_I is the total number of pixels, N_{TP} is the number of pixels which are on the actual edges and marked as edge points, N_P is

Table 2.1: Outputs of binary classification problems with confusion matrix.

	True Edge (True)	Background (False)	Total
Positive	N_{TP}	N_{FP} (Type I errors)	N_P
Negative	N_{FN} (Type II errors)	N_{TN}	N_N
Total	N_T	N_F	N_I

the total number of pixels being predicted as edge points (Positive), N_N is the total number of pixels being predicted as non-edge points (Negative), and the others have similar meanings with related information.

The measure *accuracy* is defined as in Equation (2.5), *precision* in Equation (2.6), *recall (sensitivity)* in Equation (2.7), *specificity* in Equation (2.8), *false positive (false alarm rate, type I error rate)* in Equation (2.9), and *false negative (type II error rate)* in Equation (2.10). Here, accuracy describes the detection result for correctly marking pixels as edge points or non-edge points, precision indicates the confidence of marking edge points, and recall indicates the ability of finding all true edge points. For non-edge points, specificity is similar to sensitivity. For error marking, there are false alarm and false negative.

$$p_{acc} = \frac{N_{TP} + N_{TN}}{N_I} \quad (2.5)$$

$$p_{pre} = \frac{N_{TP}}{N_P} \quad (2.6)$$

$$p_{rec} = \frac{N_{TP}}{N_T} \quad (2.7)$$

$$p_{spe} = \frac{N_{TN}}{N_F} \quad (2.8)$$

$$p_{fp} = \frac{N_{FP}}{N_F} \quad (2.9)$$

$$p_{fn} = \frac{N_{FN}}{N_T} \quad (2.10)$$

Statistical measures for detection performance have *phi coefficient* r_ϕ

(Equation (2.11), also known as *Matthews correlation coefficient* in machine learning [125]), *binary noise-to-signal ratio* f_{NSR} (Equation (2.12)) [9] and log-likelihood ratio (see Bhattacharyya bound [92]). The *F-measure* technique (Equation (2.13)) takes into account the recall and precision with a weight factor α ($0 < \alpha < 1$) [123]. The weight factor α can adjust between recall and precision, and it is normally set to 0.5 for balancing recall and precision [81, 91, 123]. For the different objectives, such as sensitivity versus precision, or sensitivity versus specificity, the receiver operating characteristic curve (ROC) has been employed to evaluate their performance [25, 123, 192]. The area under the curve (AUC) is one metric for comparing the related performance based on the ROC curves [25]. These metrics consider different objectives and are useful for binary classification problems [18]. However, statistical measures are a poor measure of detected shapes in edge detection.

$$r_\phi = \frac{N_{TP}N_{TN} - N_{FP}N_{FN}}{\sqrt{N_P N_N N_T N_F}} \quad (2.11)$$

$$f_{NSR} = \frac{N_{FP}}{N_{TN}} \quad (2.12)$$

$$f_\alpha = \frac{p_{rec}p_{pre}}{\alpha p_{rec} + (1 - \alpha)p_{pre}} \quad (2.13)$$

Localisation

Localisation is another problem in edge detection. For example, the *F-measure* technique only allows a small offset for edge points and a true edge point is only matched to a single predicted edge point [123]. Pratt [152] proposed the figure of merit (FOM) to evaluate an edge detector with the focus on localisation. FOM (f_{FOM}) is described in Equation (2.14), where $dt_1(i)$ is the distance from one predicted edge point to the nearest true edge point, Set_{N_P} is the set of pixels predicted as edge points, and α_d is a weight factor on the distance. FOM evaluates edge detectors from the viewpoint of localisation accuracy, and is sensitive to false positive (*type I* errors), but not false negative (*type II* errors). The Hausdorff distance [79] utilised the

maximum distance from a point in one set to the nearest point in the other set, which has an appropriate topological property [9] but is sensitive to noise. A metric inspired by the Hausdorff distance takes into account both distance between ground truth and predicted edges [9].

$$f_{FOM} = \frac{1}{\max\{N_T, N_P\}} \sum_{i \in Set_{N_P}} \frac{1}{1 + \alpha_d dt_1(i)^2} \quad (2.14)$$

Also, the average of point-to-set distances [144, 180], and other methods have been investigated for indicating false positives [115]. Localisation-based methods consider offsets and the matching directions between predictions and ground truth, but most of them are too sensitive to false positives [115].

A new methodology for evaluating edge detectors has been proposed based on completeness, discriminability, precision and robustness [131]. Completeness aims at evaluating the ability of detecting all possible edges in noiseless images, discriminability is used to discriminate the important and non-important edges, precision focuses on the ability to detect edges as close to the ideal edges as possible, and robustness measures the ability to detect edges in noisy images. The four quality measurements can be seen as generalisations of FOM, the F -measure technique, ROC, etc. This methodology also suggests evaluating detectors based on extracted features, rather than final binary edge maps, in order to avoid possible bias introduced by the application-dependent process of generating binary edge maps from soft edge maps [131].

To summarise, methods for evaluation of edge detectors employ a single indicator, such as Hausdorff distance, or multiple metrics, such as recall and precision. There is neither theoretical basis nor empirical evidence to prefer a specific performance indicator [115, 140].

2.2 Overview of Genetic Programming

This section presents background on Evolutionary Computation and Genetic Programming.

2.2.1 Evolutionary Computation

Evolutionary Computation (EC) is one of branch of Computational Intelligence (CI), inspired by the modern principles of biological evolution and Darwin's theories on natural selection [97, 190]. As a subfield of artificial intelligence, EC has been used to optimise problems in different domains using only weak background knowledge [43, 87]. The main model in EC is to employ the natural phenomena of genetic inheritance and selection for survival. Therefore, approaches in EC mainly employ a population of candidate solutions, updating the population and outputting the best one(s) as the final solution(s). Each candidate solution is called an individual. An individual is encoded to contain information of problem solutions. To determine whether an individual is a good solution or a bad solution, objective functions from users are used to evaluate the individual, and the score is called fitness. Such objective functions are called fitness functions. The fitness determines the goodness of an individual.

The general algorithm flow for EC is shown in Figure 2.7. More candidates will be found generation by generation after operating individuals (in "Population Operation"). In a generation (iteration), some individuals will create offspring, and old and new individuals will compete to survive to the next generation.

Popular algorithms in EC include Evolutionary Algorithms (EAs), Swarm Intelligence and Learning Classifier Systems [42]. Evolutionary algorithms were firstly introduced in EC, containing Genetic Algorithms (GAs), Genetic Programming (GP), Evolution Strategy (ES) and Evolutionary Programming (EP) [42]. Each of these algorithms evolves a population via genetic variation from two genetic operators: *crossover* and *mutation*. Crossover

Figure 2.7: General algorithm flow for EC

Figure 2.8: One binary chromosome with eight bits in GA.

exchanges information (genetic material) existing in individuals. Mutation changes information existing in an individual. A *selection* mechanism is used to choose individuals for creating new individuals via the operators crossover and mutation.

GA is one of the most popular algorithms in EC [61, 143], and was first proposed by Holland [73]. An individual in GA is encoded as a fixed length string or numerical list, called a chromosome. Figure 2.8 shows an eight bit binary chromosome. A method for decoding is used to interpret the chromosome, and a fitness function will assign a fitness for it. However, the fixed string is limited in length for the processes of encoding and decoding based on problems.

2.2.2 Genetic Programming

GP inherits properties from EC (GAs) and automatic programming, and aims to find programs using flexible encoding methods. GP supports to use varied length chromosome to encode a program and a fitness function to evaluate the outputs that the program generates. The operator crossover needs syntactically correct operation, and the operator mutation is a random rearrangement of statements. Automatically learning a set of computer programs for a particular task is very helpful for solving different problems [95, 96].

Compared with the standard GA [73] using fixed length strings to represent solutions, GP uses trees or other flexible structures. GP trees can vary in length. Candidate solutions in GP are randomly constructed from a set of primitive functions and terminals. Tree-based GP can use computer programming languages, such as *Lisp* [95, 97], to represent individuals.

Representations in Genetic Programming

A terminal set and a function set are used to construct one chromosome in GP. A terminal usually contains input variables and constants. A function set contains primitive functions. For example, " $x * x + 3 * \sin(x)$ ", for encoding one chromosome in GP, the terminal set must contain the input variable x . A constant, such as 3, may exist in the terminal set. The function set here contains " $*$ ", " $\sin$ " and " $+$ ".

How to represent a chromosome is dependent on the representation. The *tree-based* structure is the most popular representation in GP [96, 98]. A tree structure typically contains internal nodes and leaves. In a tree-based GP system, an internal node is a function, and a leaf is a terminal [97]. All possible functions form the function set. The terminal set typically includes input variables and constants. The arguments of a function are the children of the node storing the function. If each node is randomly

Figure 2.9: An example of the expression “ $x * x + 3x$ ” in the tree-based GP.

generated, it is possible that different (such as numerical and logical) types of values are incorrectly manipulated together. The strongly typed Genetic Programming (STGP) [129] restricts that the type of value from each child of an internal node must be the same as the type of value of the relative argument of the function in the internal node.

Figure 2.9 gives an example of the expression “ $x * x + 3 * x$ ” in the tree-based GP. Here the terminal set includes $\{x, 3\}$, and the function set includes $\{*, +\}$.

Other common representations include Linear Genetic Programming (LGP) [26, 136], Cartesian Genetic Programming (CGP) [127, 130], and Grammar-based Genetic Programming [126, 185]. As the tree-based GP is the most commonly used representation [107] and the operations are relatively simple, this thesis uses the tree-based GP. In the rest of this section, we describe the important aspects of the tree-based GP. For representation convenience, we simply call this representation *GP* in the rest of this thesis.

Initialisation in Genetic Programming

Like other EC algorithms, an initial population is required in GP. There are three typical ways of randomly generating the initial population: the *full* method, the *grow* method, and the *ramped half-and-half* method (the combination of the full and grow methods) [11, 151].

In a GP system, the initialisation of an individual starts from the root (randomly selecting a function from the function set). The children of the root will be randomly generated as function nodes or terminal nodes. If a

Figure 2.10: Example for the crossover operation in GP.

function node is created, the initialisation will iterate to create new nodes until all new created nodes are terminals. A user defined *maximum depth* is used to restrict the initial individuals in the full and grow methods. While randomly generating a GP program, all function nodes will be restricted to choose terminals as their children if the depth of the program reaches the maximum depth. The full method restricts all terminals in the same depth, however, the grow method allows terminals in different depths. The ramped half-and-half method has half of population created by the full method and half created by the grow method.

Crossover, Mutation and Reproduction

GP usually uses both genetic operators crossover and mutation. Figure 2.10 gives an example of the crossover operation. Parents exchange their right sub-tree (see Figure 2.10 (a)); namely the nodes and leaves in the dot line rectangles will be exchanged. After the crossover operation, the two children are shown in Figure 2.10 (b).

Figures 2.11 gives an example of the mutation operation. The parent chooses a leaf in the dot line rectangle (see Figure 2.11 (a)), “*x*”, and the leaf is replaced with a new sub-tree (see bold circle part in Figure 2.11 (b)).

The operator *reproduction* directly copies the selected individuals to the next generation [97]. In general, elitism (the best performing individuals) will be used so that the performance of the best individual does not de-

Figure 2.11: Example for the mutation operation in GP.

crease.

Selection

Fitness-proportional selection and tournament selection are two common approaches to selecting individuals to generate new individuals [11, 97, 151, 188]. Tournament selection is a popular way in GP [95, 97] and is used in this thesis. In tournament selection, a set of individuals randomly chosen from the population are compared with each other. The best of them will be selected as parents for generating new individuals. Tournament sizes can be used to adjust the selective pressure of tournament selection. If a tournament size of one is used, the selection is equal to random selection.

Applications of Genetic Programming

GP only needs weak specific domain knowledge and has the ability to create new programs. More than 8000 recorded uses of GP are covered in [107]. The popular fields of applications using GP include symbolic regression [159], classification [44], image and signal processing [100, 151], and hardware design [96]. Details of many more applications using GP are given in [96, 107, 151].

Computer vision aims at duplicating the ability of human vision to capture and understand images [173]. Image processing and image understanding is an important part of computer vision [7, 80]. Poli [150] employed GP for image analysis, such as image enhancement, feature detection, and image segmentation. GP also has been used to filter noise [148]. Object detection [100] and image classification [6, 195] are popular topics in GP. The terminal sets for image processing usually use simple statistical features, such as mean and variance in different regions and some constants [195]. The function sets have the primitive mathematical functions, such as “+”, “-”, “*”, “/”, logical operations, such as “and”, “or”, and image operators, such as a Gaussian filter, and morphological operators.

2.3 Related Work for Edge Detection with GP

GP has been used for feature construction [99, 135] and feature extraction [134, 198]. Features constructed by GP represent rich information and improve classification accuracy [99, 135]. The purpose of feature extraction is to reduce redundancies in data and transform data into a reduced representative set of features [198]. In image processing, there are a wide range of applications using GP [6, 72, 151]. However, there has been only a very limited number of works using GP for edge detection to date. The existing methods for feature extraction in edge detection can be roughly divided into two categories as follows.

Approximation Based on Desired Features

In order to obtain good edge features based on a one-dimensional step edge response, Harris and Buxton [71] utilised GP to approximate responses on one-dimensional edge signals and then employed the evolved programs (similar to the Gaussian gradient) to extract edge features based on a fixed window.

Different from designing one-dimensional signals, features similar to existing edge features have also been evolved by GP. The target is to fit the outputs of the existing feature extraction methods. GP was used to effectively approximate the outputs from the Sobel detector [70, 74] and the Canny detector [41].

Approximating existing edge detectors or desired signal responses can be considered as regression problems. However, the desired outputs are dependent on the human knowledge for understanding edge responses. Features extracted based on desired outputs (edge responses) can only represent the known (limited) edge information.

Classification of Pixels

Edge detection can be considered as a binary classification problem when edge detectors discriminate each individual pixel as an edge point or a non-edge point [150]. The outputs of edge detectors are usually evaluated based on detection accuracy after thresholding.

The simplest way to construct edge detectors is based on raw pixels. Pixels nearby a discriminated pixel are used as inputs to construct a low-level edge detector. Zhang and Rockett [197] used 13×13 windows as edge patterns to extract effective features via multi-objective GP, with objectives of Bayesian error, classification error and the number of nodes.

In order to find good edge detectors, existing image operators have been used to construct GP edge detectors. In [183], morphological operators erosion and dilation were used as the terminal set and the evolved detectors classified pixels as edge points or non-edge points. To detect boundaries, some image operators were employed to effectively combine a boundary detector by GP [84].

Also, GP has also been used to design detectors for some specific tasks. In [62], a 64-bit digital circuit for detecting edges was evolved by GP based on artificial images. In [19], programs were designed to find edge pixels in one image. These programs start at a pixel and then “walk” to find edge

points.

In summary, these evolved edge detectors (solved as approximation problems) focused on different response magnitudes on edges, but the edge detectors evolved from ground truth (solved as classification problems) focused mainly on detection accuracy. Generally, detection accuracy is based on desired outputs. Ground truth is easily obtained, but the knowledge about edges and their responses magnitudes are often limited.

2.4 Image Datasets

This thesis focuses mainly on extracted edge *features* on natural images. Synthetic images [46, 149], range images [160], and medical images [199] are not addressed here. This section briefly reviews the large datasets used in this thesis.

BSD Dataset

The Berkeley Segmentation dataset (BSD) image dataset is very popular to test for edge detection [38], boundary detection [91] and image segmentation [4]. The BSD consists of natural images (of size 481×321 pixels) with ground truth provided. There are two versions of the BSD image dataset: BSD 300 [123] and BSD 500 [4]. The BSD 500 extends the BSD 300, and the BSD 500 includes validation images in order to avoid overfitting training. Since the BSD 300 has been used as a benchmark image dataset [38, 123, 155], so only the BSD 300 is used in this thesis. There are 200 images in the training dataset and 100 images in the test dataset. For fairness of judgement of edges, the ground truth provided for every image is combined from five to ten people as a graylevel image.

Figure 2.12 shows three example training images and their ground truth. To present the human observations, brighter edges in the ground truth mean that a larger number of people marked them. Edges with

Figure 2.12: Three example training images from BSD dataset and their ground truth.

lower intensities (but not black), such as the edge in the background (middle right) of image 216053, mean that a smaller number of people marked them. Pixels coloured black (grayscale level 0) mean that they are non-edge points. Figure 2.13 shows three example test images from the BSD test images. The test images in BSD dataset are different from the training images and are used to test the generalisation ability of detectors trained using the training set of images.

Since there are many images in the BSD dataset and there are redundancies among these images, we extract a small training dataset from the 200 training images. The small training dataset contains the following 20 images: 42078, 106020, 68077, 23080, 216053, 61060, 41004, 113044, 134008, 161062, 163014, 189011, 207056, 236017, 249061, 253036, 271031, 299091, 311081, and 385028. The small training data is indicated by S_{20} .

Note that the small image dataset S_{20} includes different edge information and it is expected that S_{20} can be used to train good edge detectors. Since using the full set of all training images (200) is heavily computational, S_{20} is used to reduce computational cost in GP. Also, S_{20} is used

Figure 2.13: Three example test images from BSD dataset and their ground truth.

to check whether a small set of training images can be used to train edge detectors which can detect edges on a large set of unseen images. Using a minimal set of training images to effectively train classifiers is another research direction [45] which is not addressed in this thesis.

Note that the BSD dataset is based on segmentation results, and is not unique for true edges. Edge detection can be further manipulated for boundary detection, and the results from edge detection affect the boundary detection results. Wang, Ge and Liu [171] proposed evaluation for edge detection through boundary detection. Note that edges among different texture areas exist in the BSD image dataset. This dataset is used to find edges in textured and untextured images. In this thesis, the ground truth of BSD images are employed as desired outputs for edges.

ETHZ Shape Classes Dataset

The ETHZ (Swiss Federal Institute of Technology, Zurich) shape classes dataset is another well known object recognition dataset [39, 161], which

Figure 2.14: Three example images from ETHZ and their ground truth.

highlights the shape of objects in the segmentations. The image dataset contains different sub datasets: apple logos, bottles, giraffes, mugs and swans. The range of sizes of images in the dataset is large. Figure 2.14 shows three example images from ETHZ shape classes dataset. Figure 2.14 (a) presents image “biker” in the set of apple logos, (b) image “road” in the set of giraffes, and (c) image “black3” in the set of swans.

This image dataset mainly includes true edges from the objects of interest to an observer [39]. Different from the BSD dataset, in the ETHZ dataset, edges are not marked based on segmentation results, so some edges in the ETHZ dataset are not closed, such as single lines (whose end-points are not connected).

The ETHZ dataset is not a priori split into training and testing sets of images. In this thesis, the ETHZ dataset is only employed to *test* edge detectors trained by the BSD image dataset.

2.5 Chapter Summary

This chapter provided a brief literature review of background concepts in edge detection and existing methods for extracting edge features. The extraction methods mainly include differentiation, statistics, mathematical morphology, phase congruency and local energy, machine learning, multiresolution, and combinations of local edge features. Also, the edge detection techniques marking groups of pixels and adaptive neighbourhood processing were discussed. Methods using differentiation are fast at extracting edge features but are affected by noise and textures. Some statistical techniques can extract edge features from textures but take heavy computation. Mathematical morphology also has problems in extracting features from textures boundaries. Methods using phase congruency and local energy can extract edge features in some texture boundaries, but still have problems of suppressing other textures. Multiresolution and combinations of local techniques can improve detection from single extracted edge features, but how to combine and how to set parameters need to be addressed.

The performance evaluation of edge detectors was mainly focused on detection accuracy and localisation. There is neither theoretical basis nor empirical evidence to prefer a specific performance indicator.

The background of EC and GP was briefly introduced. GP has been widely applied in many different fields. The little existing work on edge detection using GP indicates that more investigations on GP for edge detection are required.

Finally, the main image datasets used in this thesis were described.

Part I

**Low-level Edge Feature
Construction**

Chapter 3

New Search Operators for Low-level Feature Construction

3.1 Introduction

This chapter mainly concerns edge feature extraction using very little prior knowledge, i.e., only including training images and their ground truth. The literature reviewed in the previous chapter has presented several feature extraction methods based on differentiation (Section 2.1.2 on page 21). Differentiation has been popularly applied to edge detection [140]. Neighbours of a discriminated pixel within a fixed area, such as a moving window, are utilised to extract edge features by differentiation. In general, a large window is used to filter noise, but blurs edges; a small window is used to find edge details, but is sensitive to noise. Therefore, the window size is a trade-off between noise rejection and localisation accuracy [14, 140]. In order to address the problem of the window size, automatically searching across neighbours is required. To avoid to using windows, search operations need to be based on full individual images.

Since edge features are implicit, there are no generic approaches to extracting edge features from images. The intensity value of a single pixel is not often sufficient to discriminate that pixel as an edge point or a non-

edge point. To automatically search for neighbouring pixels to use for constructing edge features, searching operations need to be designed for edge detection.

First of all, to easily search across single pixels based on full images, a search operator based on a single pixel should be designed. This search operator is used to find neighbours to construct edge detectors.

However, existing methods based on single pixels find it difficult to suppress textures [57, 140]. A filter for filtering noise or suppressing textures needs to consider a set of pixels in a local region [140]. Therefore, the search operator with single pixels is not sufficient to suppress textures, and a search operator based on a single *block* of pixels is required.

When considering a search operator using a single block of pixels, manually setting the size of the block should be avoided. The search operator is expected to have the ability to automatically select a suitable size block. Also, how to specify a block of pixels for extracting edge features is another issue. Normally, a single variable extracted from a set of pixels is used for edge detection. Methods for transforming a set of pixels to a single variable, such as using the mean of a set of pixels' intensities as a single variable, should be investigated. If search operators with a single block are combined by a common operator, such as subtraction, the possible combinations are determined by the block size and the method for extracting a single variable from a set of pixels. When many search operators with a single block are combined to construct edge features, the number of the possible combinations is large. It is possible that some combinations have redundancies and some combinations poorly extract edge features.

3.1.1 Chapter Goal

The goal of this chapter is to investigate automatically searching pixels (including a discriminated pixel itself and its neighbours) to construct low-level edge detectors using GP. A GP system based on full images as input

is proposed for automatic construction of low-level edge detectors. Rather than manually determining neighbours in a window, different search operators in this GP system are proposed to search neighbours for each discriminated pixel based on full images. To initially investigate the combinations of search operators with a single block of pixels, a search operator with two blocks of pixels is also proposed. Specifically, the following research objectives will be investigated.

- How to use GP to construct low-level edge detectors with full images, rather than using a moving window.
- Whether constructing search operators using a single block of pixels is significantly better than using a single pixel only to construct edge features.
- Whether constructing search operators using two blocks of pixels is better than using a single block of pixels to evolve low-level edge detectors.
- Whether the edge detectors evolved by GP are better than some existing common low-level edge detectors.

3.1.2 Chapter Organisation

In the remainder of this chapter, the second section introduces the GP system using full images to construct edge detectors. The third section proposes three different search operators for constructing edge features by GP. The fourth section gives the experiment settings. The fifth section presents the results of the experiments with discussions. The sixth section provides further discussions. The last section draws a summary for this chapter.

3.2 GP System Based on Full Images

This section describes the proposed GP system based on full images.

3.2.1 Sets of Terminals and Functions

This GP system uses a full image as a terminal. In general, constants are helpful for constructing GP programs in many applications [97, 151]. Besides the full image I , the terminal set contains random constants. The range of random constants rnd is from -10 to 10 based on initial experiments.

Three search operators and some common operators are utilised as functions to construct edge detectors. Here, search operators are expected to find neighbours of a discriminated pixel. Common operators are those that are usually used in a wide variety of applications [44, 96, 97, 126, 151], such as addition. For the function set, the common arithmetic operators include the addition (+), subtraction (−), multiplication (*), division (÷), absolute (*abs*), square (*square*) and square root (*sqrt*). In order to investigate the influence from discontinuous filters, the logical operators IF and $<$ are combined as $IFLess(A, B, C, D)$ as a “common” operator. $IFLess(A, B, C, D)$ means that if $A < B$ is true, the function will return C , otherwise it will return D , where A , B , C and D are two-dimensional matrices (as the same size of input image I). All functions work on each element of a matrix, such as each pixel of the input image I . The $+$, $-$, $*$, *abs*, *square* have their usual meanings. The square root function *sqrt* is protected, which produces a result of 0 for negative inputs. Division $\div$ is also protected, producing a result of 1 for a 0 divisor.

Since neighbours must be found for constructing edge detectors, search operators are necessary for the function set. In order to investigate the influence on searching operations in evolved programs, different search operators are introduced in the next section.

3.2.2 Fitness Function

The class labels for edge detection are only “edge point” or “non-edge point”, and the main class is “edge point”. For the output of a GP pro-

gram, 0 is employed as the threshold for discriminating a pixel as an edge point (larger than 0) or a non-edge point (less than or equal to 0), and all images use the intensities of pixels as integers from 0 to 255. Since only low-level edge detectors are evolved in this chapter and search operators are employed to find pixels to construct GP detectors, the output is directly evaluated without special post-processing techniques, following [131]. The fitness function employed is $f_{0.5}$ ($\alpha = 0.5$, see Equation (2.13) on page 36). The fitness function $f_{0.5}$ balances recall (Equation (2.7)) and precision (Equation (2.6)) so that the evolved edge detectors are expected to have ability to find as many edge points as possible and reduce the number of wrongly marked edge points to as few as possible.

3.3 Search Operators

An original simple task in edge detection is to construct an edge detector when only given some images and their ground truth. Since the intensity value of a pixel is not sufficient to mark that pixel as an edge point or a non-edge point, and an edge includes some local characteristics in an image, the relationship between a pixel and its neighbours needs to be found. In order to find neighbours and use them to establish a relationship (neighbourhood) for discriminating a pixel as an edge point or a non-edge point, search operators are employed.

Three search operators are employed for automatic extraction of edge features in this GP system. These three search operators select a single pixel, a single block of pixels and two blocks of pixels.

3.3.1 Searching a Single Pixel

In order to search neighbours, or these neighbours' intermediate results, for discriminating each pixel as an edge point or a non-edge point, a shifting function $s_{n,m}$ is developed based on the four functions utilised in [150].

Each function used in [150] shifts a two-dimensional matrix by one row or one column. Function $s_{n,m}$ shifts its argument (a single two-dimensional matrix input) by n columns and m rows. If n is positive, a right shifting operation performs on the input, otherwise a left shifting operation performs on its argument. If m is positive, the two-dimensional input shifts down, otherwise shifts up. Its argument can come from image I or an intermediate result of a subtree, $sub(I)$, constructed by the GP system.

Note that if the two-dimensional input is rnd , rnd is considered as a two-dimensional matrix with its elements being equal to a single random constant. The shifting operation performs on the bits of rnd , and its value is multiplied by 2^{n+m} so that the GP system can generate a large range of different constants. Here n and m are randomly selected from $\{-2, -1, 0, 1, 2\}$. When a new shifting function $s_{n,m}$ is generated and its argument is image I , each pixel or one of its neighbours in a 5×5 window has equal probability to be selected. If only using the four functions with single pixel shifting in [150], the probability for selecting a pixel in the 5×5 window to construct edge detectors will be different.

For example, a left neighbour of a discriminated pixel, which is the closest to the discriminated pixel in the left direction, has the same probability to be selected as a right neighbour (the closest pixel to the discriminated pixel in the right direction). However, the closest pixel in the left direction has obviously higher probability to be selected than the second closest pixel in the same direction because the second closest pixel needs a combination of two functions (two single pixel shifting operators). However, there are no proofs claiming that the second closest pixel is obviously less importance than the closest pixel for constructing edge detectors. The new search operator $s_{n,m}$ assumes that the all pixels in a window have the same probability to be selected. Also, compared with the approach in [150] in the view of representation of a GP tree, the new search operator $s_{n,m}$ needs fewer nodes to find pixels that are not close to the discriminated pixel.

11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55

(a) before calling $s_{-1,0}$

11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55

(b) after calling $s_{-1,0}$

Figure 3.1: Example two-dimensional intensity matrix and its result after calling $s_{-1,0}$.

Figure 3.1 (a) shows an example of a small two-dimensional matrix (for a ramp edge), and (b) is its result after calling a shifting function $s_{-1,0}$. Using $s_{-1,0}$, the intensities of the fixed pixels might be changed. Note that the last column of the shifted result is filled by the nearest element in the matrix. Via using shifting functions to implicitly search neighbours, neighbours of each discriminated pixel can be combined for constructing edge features with common operators.

It is possible for this GP system to generate some existing edge detectors. For instance, the 2×2 window Robert detector (see Equation (3.3)) [57] is represented by the GP edge detector GE_{Robert} which is given by Equation (3.4). In order to employ the neighbours of each discriminated pixel (including each discriminated pixel itself) used in the Robert detector, functions $s_{1,1}$, $s_{1,0}$ and $s_{0,1}$ are used to select the pixels around each discriminated pixel. Figure 3.2 presents a tree representation of the 2×2 window

Figure 3.2: The 2×2 Robert detector constructed with GP. Nodes “ $\sqrt{\quad}$ ” and “SQ” are functions *sqrt* and *square*, respectively.

Robert filter based on full image I .

$$R_{Robert,x} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \otimes I \quad (3.1)$$

$$R_{Robert,y} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \otimes I \quad (3.2)$$

$$R_{Robert} = \sqrt{R_{Robert,x}^2 + R_{Robert,y}^2} \quad (3.3)$$

$$GE_{Robert} = \sqrt{(I - s_{-1,-1}(I))^2 + (s_{-1,0}(I) - s_{0,-1}(I))^2} \quad (3.4)$$

3.3.2 Searching Single Blocks

In general, it is a difficult task for GP to evolve a good edge detector using only single pixels to remove influence of noise and textures. The edge detector employs the intensity level of each single pixel, and it is sensitive to the pixel intensities. For instance, a pixel, which is not an edge point and is not affected by noise or texture, is easily marked as a non-edge point by an edge detector, such as the Sobel edge detector [57]. After adding noise

to the pixel, the edge detector might detect it as an edge point. In order to suppress noise and textures, information for an edge from a local area needs to be used. In general, a local area for indicating a pixel as an edge point includes a set of its neighbours.

If only $s_{n,m}$ is used to filter noise and textures, filters as subtrees need to be constructed, and then a program is constructed by these subtrees. However, the sizes of GP trees including filters are normally very large. If a large size tree is allowed in the GP system, the search space (tree size) will be exponentially increased, which leads to an increase in the computational cost. For example, the size of a full binary tree from depth $dp - 1$ to dp is increased by $O(2^{dp-1})$. Additionally, some existing filters cannot be constructed by the simple GP system, such as a median filter [23].

To suppress textures and reject noise, existing work employs a set of pixels. The dissimilarity of two blocks of pixels were indicated by statistical approaches [111] (see details in Section 2.1.2 on page 24). A surround suppression technique utilised blocks of pixels' intermediate results (gradients) to remove some texture responses [66]. To simulate this idea, new approaches to using a set of pixels or their intermediate results need to be developed so that the GP system has some ability to search blocks of pixels to reject noise and suppress textures.

A new search operator $block_{t,l,w,d}$ is designed to find blocks of pixels to extract edge features so that the GP system can effectively construct edge detectors with some ability to filter noise and textures. The new search operator $block_{t,l,w,d}$ includes approaches (t) to transforming a block of pixels to a single variable (such as the mean of intensities of all pixels in the block), the block size parameters l (the length of the block) and w (the width of the block), and the directional position d (where the block is located around a discriminated pixel). The argument of the search operator is a two-dimensional matrix, which is image I or an intermediate result from a subtree $sub(I)$. Figures 3.3 (a) and (b) describe two examples for the blocks of pixels specified by the search operator. Here, a block of pixels

Figure 3.3: Two example blocks of pixels specified by search operator $block_{t,l,w,d}$.

("bbbb") is specified relative to the discriminated pixel ("c"). In Figure 3.3 (a), the parameters for the block are $l = 4$, $w = 1$ and $d = \text{"right"}$; and in Figure 3.3 (b), the parameters for the block are $l = 2$, $w = 2$ and $d = \text{"up"}$.

The parameter t is used to transform a set of pixels to a single variable. The purpose of using a block of pixels is to indicate some special characteristics from the set. Mean and standard deviation are commonly used to summarise a group of data. Therefore, mean ($t = 0$) and standard deviation ($t = 1$) are employed in the search operator $block_{t,l,w,d}$.

The search operator $block_{t,l,w,d}$ is used to find a set of pixels around a discriminated pixel, and it has no ability to move to a new position. From the comparison between $s_{n,m}$ and $block_{t,l,w,d}$, the latter only searches a limited area for constructing edge detectors. Without $s_{n,m}$, the pixels only found by the search operator $block_{t,l,w,d}$ are dependent on its parameters l , w and d . However, $s_{n,m}$ can search pixels which are far away from the relevant discriminated pixel, and the ability to find pixels is dependent on not only its parameters n and m , but also the maximum depth of a GP tree.

In contrast to the existing approaches to utilising blocks of pixels, the search operator $block_{t,l,w,d}$ provides four distinguished characteristics. Firstly, an unfixed size block (used to find a set of pixels) is different from the common way using a fixed block [38, 123, 140]. It is possible that GP can find a suitable size block to construct edge detectors, which decreases the com-

11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55

Figure 3.4: Example of searching neighbours by $block_{t,l,w,d}$.

putational cost but does not affect the ability to detect edges. Secondly, flexible positions to find a block of pixels are used after calling the search operator $s_{n,m}$. In a ramp edge or a stair edge (see details in Figure 2.2 on page 19), the closest neighbours might affect the detection results because of its discontinuity. Using a combination of $s_{n,m}$ and $block_{t,l,w,d}$, such as $block_{t,l,w,d}(s_{n,m}(I))$, possibly avoids this influence. Thirdly, different directional positions are used together to construct edge detectors so that the directional calculation is avoided. Lastly, a flexible input in $block_{t,l,w,d}$ comes from original image I or an intermediate result after some processing on image I (subtree). After calling a subtree $sub(I)$ for filtering noise or textures, $block_{t,l,w,d}(sub(I))$ might help improve detection performance.

Figure 3.4 shows an example of searching blocks of pixels for discriminating the centre pixel (blue) in four different directions. Here, the neighbours with the same colour are in the same block. There are two major areas with intensities 11 and 55 respectively. Three columns with intensities 20, 30 and 40 are the critical region between both major areas. From existing methods, a comparison between the right block and the left block or between the right block and the bottom block is easy to find the ramp edge. For instance, the mean of pixel intensities from different blocks can be compared. The difference of the means of pixel intensities in the left block and the right block is $\frac{11*4+20*2}{6} - \frac{40+55*3}{4} = -37.25$. Here, out of the

right area is considered as the area with intensity 55. To discriminate the right neighbour (at the sixth column), all blocks move right by a pixel, and the right block has four "55" (three in the figure and one being out of the right area). The differences of the means for the right neighbours (the sixth, seventh and eighth columns) are -34.67 ($\frac{11*2+20*2+30*2}{6} - \frac{55*4}{4}$), -25.00 , and -13.33 , respectively. The differences of the means between the left block and the up block for the pixel and its relative right neighbours are -21.00 , -27.17 , -25.00 and -13.33 , respectively. Since the means from the left block and the up block are close, the relative response on the ramp edge is thick.

In Figure 3.4, if only using $s_{n,m}$, a high threshold 14 might be used to detect the boundary between the seventh and sixth columns based on the difference of two columns. The difference of the intensities between the seventh and sixth columns ($55 - 40$) is 15. However, if the intensities of the pixels at the last three columns are changed to 50 and the intensities of the pixels at the first three columns are changed to 10, the responses from the third columns to seventh columns are the same ($30 - 20 = 40 - 30 = 50 - 40 = 10$) if the difference of two columns is employed. Therefore, $s_{n,m}$ has difficulty to find edges after some change in the intensities of these pixels.

3.3.3 Searching Double Blocks

While $block_{t,l,w,d}$ can be used to search good edge features as discussed previously, the search space is huge. In order to reduce the search space of using blocks of pixels, two blocks with the same size are combined by common operators, and a new search operator is developed for combining two blocks of pixels. In edge detection, edge points are often located at boundaries between "different" areas. The detection mainly focuses on finding differences. Subtraction $-$ and division $\div$ are usually used for indicating difference from numerics. Therefore, we employ $-$ and $\div$ in the new search operator tb_{j,t,l,w,d_1,d_2} ($j = '-'$ or $j = '\div'$) using two blocks of

Figure 3.5: Examples of using two blocks of pixels.

pixels. Equations (3.5) and (3.6) give the definition of the search operator tb_{j,t,l,w,d_1,d_2} for $j = '-'$ and $j = '\div'$, respectively. Here, ε is a small positive constant, d_1 and d_2 are different directions, and $tb_{\div,t,l,w,d_1,d_2}$ requires that $block_{t,l,w,d_2}$ must be larger than or equal to 0.

$$tb_{-,t,l,w,d_1,d_2} = block_{t,l,w,d_1} - block_{t,l,w,d_2} \quad (3.5)$$

$$tb_{\div,t,l,w,d_1,d_2} = \frac{block_{t,l,w,d_1}}{block_{t,l,w,d_2} + \varepsilon} \quad (3.6)$$

Figures 3.5 (a) and (b) show combinations of two blocks of pixels used for tb_{j,t,l,w,d_1,d_2} . To discriminate the centre pixel, four combinations of two blocks are listed in Figure 3.5. Here, the same colour blocks are used in the same search operator tb_{j,t,l,w,d_1,d_2} . In this chapter, the combinations of d_1 and d_2 are limited to the four different combinations: left up and right up, left bottom and right bottom, left up and left bottom, and right up and right bottom.

Using tb_{j,t,l,w,d_1,d_2} is approximately considered as a small subset of using $block_{t,l,w,d}$. The aims of proposing tb_{j,t,l,w,d_1,d_2} are reducing the space for searching blocks of pixels and investigating which t in $block_{t,l,w,d}$ exists with high frequency in evolved edge detectors. Note that tb_{j,t,l,w,d_1,d_2} is not randomly selected from combinations of $block_{t,l,w,d}$. There are two elements of prior knowledge used in tb_{j,t,l,w,d_1,d_2} , namely the comparison

Figure 3.6: Example training image 23080 and its sampled result of size 125×125 pixels.

using difference and the approach (t) frequently used in the evolved edge detectors when $block_{t,l,w,d}$ is used. Additionally, when a random constant rnd is used as an argument in $block_{t,l,w,d}$ or tb_{j,t,l,w,d_1,d_2} , the return value is the same as rnd .

3.4 Experiment Settings

This section describes an image dataset and the settings for the GP system.

3.4.1 Image Dataset

To evolve edge detectors, the BSD image dataset [123], including natural images with ground truth provided, is chosen. To sample training images as the training data, the small training dataset S_{20} (see Section 2.4 on page 46) is employed. For each image, a single subimage of size 125×125 pixels is randomly extracted. Figure 3.6 shows training image 23080 and

Table 3.1: Search operators in the function set.

name	search operators
Set_s	$\{s_{n,m}\}$
$Set_{s,b}$	$\{s_{n,m}, block_{t,l,w,d}\}$
$Set_{s,b,tb}$	$\{s_{n,m}, block_{t,l,w,d}, tb_{j,t,l,w,d_1,d_2}\}$
Set_b	$\{block_{t,l,w,d}\}$
Set_{tb}	$\{tb_{j,t,l,w,d_1,d_2}\}$
$Set_{s,tb}$	$\{s_{n,m}, tb_{j,t,l,w,d_1,d_2}\}$

its randomly sampled subimage of size 125×125 pixels. The test dataset is the 100 BSD full sized (481×321) test images.

3.4.2 Sets of Terminals and Functions

In order to investigate the influence of the function set, different settings are used to choose search operators for the GP system. To simplify the description using a function set and a terminal set, Table 3.1 shows different settings for the search operators in the function set, and $\{+, -, *, \div, abs, sqrt, square\}$ are always kept in the function set. Note that all settings in Table 3.1 have two situations, namely including *IFLess* and without *IFLess*.

When a new function node is created, a function operator is randomly selected from the function set. In order to increase the probabilities of choosing operators $s_{n,m}$, $s_{0,1}$, $s_{1,0}$, $s_{0,-1}$ and $s_{-1,0}$ are directly added into the function set when $s_{n,m}$ is employed.

3.4.3 Parameter Settings

The parameter values for $block_{t,l,w,d}$ are: $t = 0$ for the mean or $t = 1$ for the standard deviation, l is from 3 to 7, w is from 1 to 7, and d is one of left,

right, up or down direction. From the initial experiment results, standard deviation has higher occurrences than mean in the evolved edge detectors, so only standard deviation is used in tb_{j,t,l,w,d_1,d_2} . The parameter values for GP are: population size 800; maximum depth (of a program) 10; maximum generation 200; and probabilities for mutation 0.15, crossover 0.80 and elitism (replication) 0.05. These values are chosen based on common settings and initial experiments. Each GP experiment for each setting is repeated for 30 independent runs.

3.5 Results and Discussions

This section firstly gives the results without *IFLess* in the function set, and then the results including *IFLess* in the function set. In order to compare this GP system with the common method using a moving window, a common system [197] uses all pixels in a 5×5 window as terminals, and employs all functions in the proposed GP system, except for the three search operators. Note that n and m in $s_{n,m}$ are from -2 to 2 . Using a single $s_{n,m}$ is equal to selecting a pixel from a 5×5 window. Therefore, the 5×5 window is chosen in the common system. The common system detects edge points based on raw pixels, not full images. In the common system, fixed neighbours are given as terminals. Like the Sobel edge detector, the common system moves the window pixel by pixel to detect edge points. $Set_{5 \times 5}$ is used to indicate the evolved edge detectors using the 5×5 window to detect pixels one by one.

3.5.1 Without *IFLess*

Overall Results

Table 3.2 gives the test performance of the means and standard deviations of $f_{0.5}$, the maximum $f_{0.5}$ of the evolved edge detectors, and the means and standard deviations of the test time on the 100 BSD test images. All

Table 3.2: Test performance (mean $\pm$ standard deviation) of the GP edge detectors without *IFLess* on the 100 BSD test images. Note that the time is for testing one BSD image.

	$f_{0.5}$	maximum ($f_{0.5}$)	time (seconds)
$Set_{5 \times 5}$	0.2563 ± 0.0046	0.2619	0.0403 ± 0.0140
Set_s	0.2632 ± 0.0098	0.2807	0.0213 ± 0.0093
Set_b	0.2844 ± 0.0164	0.3210	0.6137 ± 0.2363
Set_{tb}	0.2774 ± 0.0158	0.3116	0.2414 ± 0.0826
$Set_{s,b}$	0.2953 ± 0.0161	0.3170	0.2971 ± 0.1939
$Set_{s,tb}$	0.2895 ± 0.0103	0.3063	0.1031 ± 0.0566
$Set_{s,b,tb}$	0.2919 ± 0.0130	0.3148	0.2152 ± 0.1330

settings for search operators, except for $Set_{5 \times 5}$, come from Table 3.1. Here the test time on each image is on a single machine with CPU 3.1 GHz based on an implementation in C++. From Table 3.2, the highest mean of $f_{0.5}$ comes from the edge detectors evolved by GP using $Set_{s,b}$ (the search operators $s_{n,m}$ and $block_{t,l,w,d}$). The common method using the 5×5 moving window ($Set_{5 \times 5}$) has the lowest mean of $f_{0.5}$. From the comparison of the best evolved edge detector (maximum $f_{0.5}$) from each setting, using the search operator $block_{t,l,w,d}$ (Set_b) only has the best edge detector with $f_{0.5} = 0.3210$ in Table 3.2, which is increased by 23% ($\frac{0.3210-0.2619}{0.2619}$), compared with the best edge detector from $Set_{5 \times 5}$. The best edge detectors from GP using search operators with a single block of pixels and/or two blocks of pixels are higher than 0.3. However, the $f_{0.5}$ values of the best evolved edge detectors using single pixels are not higher than 0.3.

For the computational cost, the evolved edge detectors using single pixels (less than 0.1 second) are much faster than the evolved edge detectors including the search operators with blocks of pixels (their average being longer than 0.1 second). Compared with the evolved edge detectors

only using the search operator $block_{t,l,w,d}(Set_b)$, the evolved edge detectors using the search operator tb_{j,t,l,w,d_1,d_2} obviously reduce the computational cost of detecting images. The evolved edge detectors from $Set_{s,b,tb}$ take almost a third of the test time for the evolved edge detectors from Set_b . The evolved edge detectors from $Set_{s,tb}$ only take around 0.1 second. In $Set_{s,tb}$, 18 of the 30 evolved edge detectors take obviously less than 0.1 second to detect a BSD image. Therefore, the search operator using two blocks of pixels effectively reduces the computational cost, compared to those using a single block of pixels. Note that a GP edge detector executes from the bottom up, which leads to redundancies existing in the calculations, such as the same subtrees. This is a potential reason that the computational cost in the evolved edge detectors using blocks of pixels is heavy.

The standard deviations in these evolved edge detectors using blocks of pixels (on detecting time) are large. If using $block_{t,l,w,d}$, the detecting time for evolved edge detectors is varied. For example, the standard deviation from Set_b is large, and the detecting times of the evolved edge detectors are located in a wide range. Comparing Set_{tb} with Set_b , the evolved edge detectors using tb_{j,t,l,w,d_1,d_2} only is more stable than the evolved edge detectors using $block_{t,l,w,d}$ only. From the detecting times in Table 3.2, it is found that combining $block_{t,l,w,d}$ or tb_{j,t,l,w,d_1,d_2} with $s_{n,m}$ can improve the stability of the test times. For example, $Set_{s,b}$ and $Set_{s,tb}$ obviously decrease the standard deviations on detecting a BSD image.

Note that the standard deviations of $f_{0.5}$ in Table 3.2 are small, although the test times are varied. Since using $block_{t,l,w,d}$ is not stable for the detecting cost, all evolved edge detectors using $block_{t,l,w,d}$ cannot be considered as heavy computational cost detectors. In these experiments, some evolved edge detectors using $block_{t,l,w,d}$ take less than 0.1 second to detect a BSD test image. For instance, one evolved edge detector from $Set_{s,b}$ with $f_{0.5} = 0.2894$ only takes 0.04 seconds for detecting a BSD image.

Table 3.3: Recall and precision of the GP edge detectors without *IFLess* on the 100 BSD test images.

	recall	precision
$Set_{5 \times 5}$	0.4486 ± 0.0255	0.1797 ± 0.0056
Set_s	0.4460 ± 0.0355	0.1872 ± 0.0087
Set_b	0.4045 ± 0.0462	0.2226 ± 0.0253
Set_{tb}	0.3976 ± 0.0461	0.2165 ± 0.0238
$Set_{s,b}$	0.4150 ± 0.0486	0.2320 ± 0.0227
$Set_{s,tb}$	0.4156 ± 0.0331	0.2237 ± 0.0167
$Set_{s,b,tb}$	0.4147 ± 0.0505	0.2283 ± 0.0213

Recall and Precision

The $f_{0.5}$ values on the BSD test images are not high, because there is no allowed offset distance from a predicted edge point to its true edge point in $f_{0.5}$. Therefore, the precision is not high. Table 3.3 gives the means of recall and precision for these evolved edge detectors. From the table, the means of precision of the evolved edge detectors using blocks of pixels are higher than the evolved edge detectors using single pixels, but the former decreases their recall. Compared to the evolved edge detectors from using the moving window, the evolved edge detectors using the shifting function $s_{n,m}$ in Set_s increase by 4.17% in precision, but only decrease by 0.58% in recall. Note that the means of recall and precision are obtained from the evolved edge detectors using different settings, and that the means of $f_{0.5}$ are not equal to the result of $f_{0.5}$ combining the means of recall and precision.

There is an interesting observation regarding all evolved edge detectors with high recall and low precision. It is possible that the same $f_{0.5}$ values might have high precision and low recall. Detection results with high precision and low recall describes that only a few true edge points

were detected, but false alarms are not many either. From the recall values in Table 3.3, the largest difference in recall comes from $Set_{5 \times 5}$ and Set_{tb} (0.0510). There is similar ability of finding true edge points for the evolved edge detectors using single pixels or including blocks of pixels. The largest difference in precision is between $Set_{5 \times 5}$ and $Set_{s,b}$ (0.0523). Although the difference in precision for these evolved edge detectors is similar to the difference in recall, the influence on detected images is different. Since the number of true edge points is fixed, recall indicates the detected edges against the true edges but precision indicates the ability for accurately predicting pixels as true edge points. A slight improvement in precision possibly makes more true non-edge points distinguished from predicted edge points when precision is low. For the two extreme prediction results (all pixels predicted as edge points or non-edge points), $f_{0.5}$ is not the lowest for all pixels predicted as edge points (recall is 1.0 and precision is the percentage of true edge points in all pixels). However, $f_{0.5}$ is the lowest (0.0) when all pixels are predicted as non-edge points. In the early evolution stage, $f_{0.5}$ might be very low in each program. Therefore, programs with high recall survive more easily than the programs with low recall. The evolved edge detectors can detect most edge points, and are not trapped to only give responses on a few true edge points.

Statistical Comparisons

Figure 3.7 provides boxplots for the $f_{0.5}$ values of the evolved edge detectors from the different settings over the 30 replications. Here, the label of each box is the index of each setting. From these boxplots, most of the evolved edge detectors using search operators are better than the evolved edge detectors using the 5×5 moving window. Also, most of the evolved edge detectors using blocks of pixels are better than the evolved edge detectors using single pixels only. However, the range of the test performances ($f_{0.5}$) of the evolved edge detectors using blocks of pixels is large, which means that the robustness of GP using the search operators

Figure 3.7: Boxplots ($f_{0.5}$) for evolved edge detectors using different settings and without *IFLess*.

with blocks of pixels needs to be further improved. One evolved edge detector from $Set_{s,b}$ (outlier) has the test performance of $f_{0.5} = 0.2448$, which is worse than most of the edge detectors from $Set_{5 \times 5}$ and Set_s . Compared with the other settings using blocks of pixels, $Set_{s,b,tb}$ has shorter range of $f_{0.5}$ than Set_b and Set_{tb} , and a better evolved result than $Set_{s,b}$ in the worst case.

Table 3.4 gives p -values using two-sample t -tests for each pair of settings. Here, the first group comes from the relevant setting in the first column and the second group comes from the relevant setting in the first row; $\downarrow$ indicates that the first group is significantly worse than second group; and $\uparrow$ indicates that the first group is significantly better than the second group when using the significance level 0.05. From the table, the evolved edge detectors from Set_s are significantly better than the evolved edge detectors from $Set_{5 \times 5}$. It seems that this GP system using the search operator $s_{n,m}$ is better than the common approach to using the 5×5 moving window. From the p -values of comparing Set_s with the other settings (using blocks of pixels) respectively, the evolved edge detectors using blocks of pixels are significantly better than the evolved edge detectors using single pixels only. When the search operator using two blocks of pixels is em-

Table 3.4: Statistical p -values (two-samples t -tests) among constructed GP edge detectors from $Set_{5 \times 5}$, Set_s , Set_b , Set_{tb} , $Set_{s,b}$, $Set_{s,tb}$, $Set_{s,b,tb}$ on the 100 BSD test images.

	Set_s	Set_b	Set_{tb}	$Set_{s,b}$	$Set_{s,tb}$	$Set_{s,b,tb}$
$Set_{5 \times 5}$	0.0011 ↓	0.0000 ↓	0.0000 ↓	0.0000 ↓	0.0000 ↓	0.0000 ↓
Set_s		0.0000 ↓	0.0001 ↓	0.0000 ↓	0.0000 ↓	0.0000 ↓
Set_b			0.1060	0.0141 ↓	0.1639	0.0601
Set_{tb}				0.0001 ↓	0.0011 ↓	0.0004 ↓
$Set_{s,b}$					0.1106	0.3865
$Set_{s,tb}$						0.4404

ployed, the results from Set_{tb} are significantly worse than the results from $Set_{s,b}$, $Set_{s,tb}$ and $Set_{s,b,tb}$; the comparisons between $Set_{s,tb}$ and $Set_{s,b,tb}$ and between $Set_{s,tb}$ and $Set_{s,b}$ are not significantly different.

There is an interesting comparison between Set_b and Set_{tb} , and the relevant p -value from the two-sample t -test is 0.0601, which is very close to the significance level 0.05. If a paired t -test for the comparison between Set_b and Set_{tb} is utilised, the relevant p -value is 0.03592, which indicates that they are significantly different. Here, the assumption for the paired t -test is that each difference (in the statistical test) of the results from Set_b and Set_{tb} has the same condition (choosing the same random seed); and a two-sample t -test assumes that two groups are independent. From these comparisons, it is shown that using tb_{j,t,l,w,d_1,d_2} alone is not good enough to construct edge detectors.

From the comparison between Set_b and $Set_{s,b}$, the results from Set_b is significantly worse. It suggests that using the search operator $block_{t,l,w,d}$ only is not enough to construct good edge detectors. There is no significant difference of $f_{0.5}$ values between Set_{tb} and Set_b , but the edge detectors evolved from Set_b are faster than the detectors evolved from Set_{tb} to de-

tect edges. It is suggested that the search operator tb_{j,t,l,w,d_1,d_2} can improve efficiency for constructing edge detectors, in terms of detection speed and accuracy.

Apparently, combining the search operator using a single pixel and the search operators using blocks of pixels improves the detection performance ($f_{0.5}$) for the evolved edge detectors. A reason is that $block_{t,l,w,d}$ and tb_{j,t,l,w,d_1,d_2} can search new pixels after combining them with $s_{n,m}$. Via using $s_{n,m}$, intermediate results from $block_{t,l,w,d}$ or tb_{j,t,l,w,d_1,d_2} for each discriminated pixel and its neighbours can be further manipulated, which makes it possible to improve detection accuracy.

To further confirm the comparisons in Tables 3.4, the number of the evolved edge detectors from a setting better than another setting is analysed by a non-parametric statistical technique. The non-parametric statistical technique, namely the Kruskal-Wallis method [174] from an overall comparison, is employed. Table 3.5 reveals the test results using the overall significance 0.05 and Holm's method [75] for p -value adjustment. The multiple statistical tests show that there is no significant difference between Set_s and $Set_{5 \times 5}$. From the overall comparison, there are no significant differences between Set_b and $Set_{s,b}$, between Set_{tb} and $Set_{s,tb}$, and between Set_{tb} and $Set_{s,b,tb}$. The results from Set_{tb} are significantly worse than the result from $Set_{s,b}$. However, in terms of the computational cost for detecting images, the evolved edge detectors from Set_{tb} is not worse than the evolved edge detectors from $Set_{s,b}$.

To summarise, from the test performance comparison ($f_{0.5}$ and the time for detecting a BSD image), it seems that the search operator $block_{t,l,w,d}$ mainly works for detection accuracy improvement, but introduces heavy computational cost. Using the search operators $s_{n,m}$ and $block_{t,l,w,d}$ together makes detection performance improve and computational cost lower than using $block_{t,l,w,d}$ only. When adding the knowledge found from the evolved edge detectors in $Set_{s,b}$ to combine two blocks together as a new search operator (namely tb_{j,t,l,w,d_1,d_2}), the evolved edge detectors can fur-

Table 3.5: Non-parametric statistical comparison (Kruskal-Wallis statistic [174]) among constructed GP edge detectors from $Set_{5 \times 5}$, Set_s , Set_b , Set_{tb} , $Set_{s,b}$, $Set_{s,tb}$, $Set_{s,b,tb}$ on the 100 BSD test images.

	Set_s	Set_b	Set_{tb}	$Set_{s,b}$	$Set_{s,tb}$	$Set_{s,b,tb}$
$Set_{5 \times 5}$	—	↓	↓	↓	↓	↓
Set_s		↓	↓	↓	↓	↓
Set_b			—	—	—	—
Set_{tb}				↓	—	—
$Set_{s,b}$					—	—
$Set_{s,tb}$						—

ther reduce the computational cost for detecting images, and keeps almost the same detection performance ($f_{0.5}$).

3.5.2 Using *IFLess*

Table 3.6 gives the test performances ($f_{0.5}$ and test time) of GP edge detectors using *IFLess*. Here, the p -value is obtained from a comparison between the first group (the test performance $f_{0.5}$ when using *IFLess*) and the second group (the test performance $f_{0.5}$ when not using *IFLess*). Surprisingly, *IFLess* does *not* improve the test performance in each setting. The evolved edge detectors from $Set_{s,tb}$ using *IFLess* is significantly worse than the evolved edge detectors from $Set_{s,tb}$ without *IFLess*. From the best evolved edge detector for each setting, the best evolved edge detectors from $Set_{s,b}$ and $Set_{s,b,tb}$ are better than their best edge detectors without *IFLess*. Both edge detectors using *IFLess* have higher $f_{0.5}$ than all edge detectors without *IFLess*. Function *IFLess* enlarges the search space for constructing edge detectors, but it might increase the difficulty of finding good edge detectors in the GP system.

One obvious difference between using and not using *IFLess* is the

Table 3.6: Test performance (mean $\pm$ standard deviation) of GP edge detectors using *IFLess* on the 100 BSD test images.

	$f_{0.5}$	maximum ($f_{0.5}$)	time (seconds)	p -value
$Set_{5 \times 5}$	0.2546 ± 0.0049	0.2611	0.0998 ± 0.0639	0.1717
Set_s	0.2604 ± 0.0084	0.2739	0.0438 ± 0.0212	0.2497
Set_b	0.2884 ± 0.0174	0.3105	1.1263 ± 0.4270	0.3814
Set_{tb}	0.2750 ± 0.0167	0.3050	0.3163 ± 0.0948	0.5579
$Set_{s,b}$	0.2980 ± 0.0133	0.3217	0.5382 ± 0.3485	0.4878
$Set_{s,tb}$	0.2792 ± 0.0162	0.3025	0.1261 ± 0.0725	0.0055 $\downarrow$
$Set_{s,b,tb}$	0.2970 ± 0.0163	0.3242	0.3568 ± 0.2293	0.2007

computational cost for the evolved edge detectors. Using *IFLess* increases the time of detecting a BSD image. All evolved edge detectors using *IFLess*, except for the settings including tb_{j,t,l,w,d_1,d_2} , almost take twice as much time as the relevant evolved edge detectors without *IFLess*. A potential reason is that all branches (of *IFLess*) in the tree-based programs are called because of the execution from the bottom up. If the GP programs based on full images are transformed to programs based on moving windows, the computational cost for a node *IFLess* does not decrease obviously. Function *IFLess* has four arguments (A , B , C and D), only one of its arguments (C or D) does not need to be called, so it seems that using *IFLess* has heavier computational cost than the arithmetic operators.

An interesting observation is that the values of test time in the settings including tb_{j,t,l,w,d_1,d_2} do not increase too much (less than double) when *IFLess* is used. In particular, the increased time for $Set_{s,tb}$ is very short. It seems that *IFLess* has weaker influence on tb_{j,t,l,w,d_1,d_2} than the other search operators, in terms of the computational cost. A potential reason is that subtrees including tb_{j,t,l,w,d_1,d_2} with edge information might not be good to extract edge features when they are only used as the first two

arguments of *IFLess*.

Assuming that there are two different types of edges in one image, the first two arguments in *IFLess* could perhaps be used to discriminate the edge type for each pixel, and the last two arguments are two subtrees indicating edge information. However, the search space for the four arguments is too large, so it is very difficult to discover a subtree containing this structure. When its first argument is always larger or smaller than the second argument, such as $IFLess(2, 3, C, D)$, *IFLess* only equals to the third argument or the fourth argument. Therefore the influence from *IFLess* is not obvious. Whether *IFLess* is helpful to construct edge detectors depends on its ability to construct the four arguments including edge information. If existing edge knowledge is employed in the four arguments, the logical operator possibly works better for edge detector construction, which will be investigated in Chapter 7.

3.5.3 Comparing with Existing Edge Detectors

In order to compare the evolved edge detectors with some existing edge detectors, the Sobel edge detector and the Gaussian gradient are chosen since they have been widely used [14, 57, 140]. The edge responses on the BSD test images from the Sobel edge detector and the Gaussian gradient are normalised to the range from 0 to 1. Different thresholds $\frac{k}{51}$ ($k = 1, \dots, 50$) are used to find the maximum of $f_{0.5}$. Here, the maximum of $f_{0.5}$ for the Sobel edge detector is 0.2337, and maximum of $f_{0.5}$ for the Gaussian gradient is 0.2643. The parameters for the Gaussian gradient are: the window size with 7×7 and the scale with 1.0. The 7×7 window is selected because its maximum of $f_{0.5}$ is higher than the maximum value from 5×5 and 3×3 windows. The Sobel edge detector and the Gaussian gradient are deterministic, so there is only one value for each detector.

Table 3.7 shows the comparisons among the evolved edge detectors without *IFLess*, the Sobel edge detector, and the Gaussian gradient. Here,

Table 3.7: Comparison (p -values using one-sample t -tests) among the evolved edge detectors without *IFLess*, the Sobel edge detector, and the Gaussian gradient.

	Sobel	Gaussian gradient
$Set_{5 \times 5}$	0.0000 $\uparrow$	0.0000 $\downarrow$
Set_s	0.0000 $\uparrow$	0.4897
$Set_{s,b}$	0.0000 $\uparrow$	0.0000 $\uparrow$
$Set_{s,b,tb}$	0.0000 $\uparrow$	0.0001 $\uparrow$
Set_b	0.0000 $\uparrow$	0.0000 $\uparrow$
Set_{tb}	0.0000 $\uparrow$	0.0000 $\uparrow$

one-sample t -tests are used to compare the results of the evolved edge detectors without *IFLess* (from each setting) with the result of the Sobel edge detector or the Gaussian gradient. As can be seen, all GP evolved edge detectors are significantly better than the Sobel edge detector. Compared with the Gaussian gradient, the evolved edge detectors from $Set_{5 \times 5}$ are significantly worse, the evolved edge detectors from Set_s are not significantly different, but the other GP evolved edge detectors using the search operators are significantly better. Therefore, using search operators can effectively evolve good edge detectors.

When a filter uses a window to perform convolution with images and combines the convolution results from the horizontal and vertical directions, the test time for detecting a BSD image is 0.02 seconds using a 3×3 window (the Sobel edge detector), 0.05 seconds for using a 5×5 window, and 0.07 seconds for using a 7×7 window (the Gaussian gradient). From Table 3.2, the GP evolved edge detectors using $s_{n,m}$ have similar detecting time to the Sobel edge detector. The average time for the GP evolved edge detectors from $Set_{5 \times 5}$ is slightly shorter than a filter with a 5×5 window. When using single pixels to construct edge detectors, it seems that GP can

Figure 3.8: Three example BSD test images and their ground truth (GT).

effectively reduce the computational cost of detecting images.

3.5.4 Example Detected Images

Using Single Pixels

Figure 3.8 shows three example images from the BSD test dataset. Here, “GT” indicates the ground truth. These three images include different edge information. Image 42049 has clear objects and its background has slightly gradual change at the four corners. Image 69020 includes regular textures (grass and animal skin), a very low contrast area between the background and the back of the animal, and noise in the background. Image 385039 also includes textures, but the texture on the wall gives much stronger responses than the grass texture.

Figure 3.9 shows the results of the three example images detected by the Sobel edge detector and the Gaussian gradient. The binary edge results detected by the Sobel edge detector are obtained by the threshold 0.1764 ($f_{0.5} = 0.2337$), and the Gaussian gradient uses the threshold 0.1961 ($f_{0.5} = 0.2643$) for the relevant binary detected edges. Both thresholds are selected

Figure 3.9: Three example BSD test images detected by the Sobel edge detector and the Gaussian Gradient (GG).

from the maximum $f_{0.5}$ with $\frac{k}{51}$.

From these three detected images, the Sobel edge detector finds most of the true edge points, but it is strongly affected by the textures and noise, such as the textures in images 69020 and 385039. The Gaussian gradient filters some noise in images 69020 and 385039, such as their grass background. For image 69020, the Gaussian gradient has a higher precision than the Sobel edge detector. The three example detected images show that edge detectors directly using single pixels are good at detecting clear boundaries (high contrast and few textures), but have problems to detect edges affected by textures and noise.

Figure 3.10 shows the three images detected by the best GP evolved edge detectors from $Set_{5 \times 5}$ and Set_s . Similar to the Sobel edge detector, the GP evolved edge detector from Set_s finds most of true edge points in the three images. The GP evolved edge detector is also affected by noise and textures. However, compared with the results from the Sobel edge detector, the edge detector from Set_s reduces the influence of the textures and noise in image 69020. Compared with the Gaussian gradient, the edge

Figure 3.10: Three example test images detected by the best edge detectors from $Set_{5 \times 5}$ and Set_s , respectively.

detector from Set_s has slightly weaker ability to filter noise, such as the background in image 385039. From the comparison between the edge detectors from $Set_{5 \times 5}$ and Set_s , the three detected images from $Set_{5 \times 5}$ have more falsely predicted edge points, even for the edges of the clear object (bird) in image 42049. There are more falsely predicted edge points in the body of the bird (from $Set_{5 \times 5}$) than the detected result from Set_s . An interesting observation is that the detected edge at the top of the roof in image 385039 (top right) from the Sobel edge detector and the Gaussian gradient is broken, but the GP evolved edge detectors from both $Set_{5 \times 5}$ and Set_s find the full edge.

Using Blocks of Pixels Only

Figure 3.11 gives the three example images detected by the best GP evolved edge detectors from Set_b and Set_{tb} , respectively. Obviously different from the detection results of the edge detectors using single pixels only, the results from the two GP evolved edge detectors using blocks of pixels have

Figure 3.11: Three example test images detected by the best edge detectors from Set_b and Set_{tb} , respectively.

good suppression on textures. For image 42049, the two edge detectors with blocks of pixels keep a similar performance to the edge detectors using single pixels. For image 69020, the textures are filtered out by the two GP evolved edge detectors with blocks of pixels. For image 385039, the evolved edge detector from Set_b reduces the influence of the wall texture, and the GP evolved edge detector from Set_{tb} almost suppresses the wall texture. Therefore, using blocks of pixels can effectively work on texture suppression.

However, there are two problems in the two GP evolved edge detectors. The first problem is that some detected edges are broken. For instance, the detected edges at the legs of the animal in image 69020 are broken. A potential reason is that the suppression of responses on textures is so heavy that some responses on true edges are filtered. The other problem is that abnormal detection occurs at some pixels. Taking the detected image 42049 as an example, both evolved edge detectors have false positives in the background. These false positives might be caused by the standard deviation on blocks of pixels. Further processing can address these two

Figure 3.12: Three example test images detected by the best edge detectors from $Set_{s,b}$, $Set_{s,tb}$, and $Set_{s,b,tb}$, respectively.

problems, such as linking edges, and removing stand-alone edges.

Additionally, the thick edges detected by the evolved edge detectors can be thinned by normal thinning operators [106], so the thickness of the detected edges is not addressed in this chapter.

Combinations of Search Operators

Figure 3.12 shows the three images detected by the best edge detectors using single pixels and blocks of pixels ($Set_{s,b}$, $Set_{s,tb}$ and $Set_{s,b,tb}$). Similar to the detected results from Set_b and Set_{tb} , the detected results from the combinations of the three search operators are not strongly affected by textures. However, there are too many false positives existing in images 69020 and 385039, especially for the results from $Set_{s,tb}$ and $Set_{s,b,tb}$. In

terms of fewer false spots in the detected results from $Set_{s,b}$ than $Set_{s,tb}$ and $Set_{s,b,tb}$, the best edge detector from $Set_{s,b}$ is slightly better than the best edge detectors from $Set_{s,tb}$ and $Set_{s,b,tb}$.

It seems that all of the GP evolved edge detectors using blocks of pixels suppress textures, but at the same time have the problem of abnormally detecting false positives. Compared with the edge detectors using single pixels only, the edge detectors using blocks of pixels do not decrease obviously in finding true edge points, but obviously remove lots of falsely predicted edge points.

3.5.5 Summary

From the test performances (test time on a BSD test image and $f_{0.5}$), there are four interesting observations from the experiment results. Firstly, the common GP method using a 5×5 moving window has higher $f_{0.5}$ than the Sobel edge detector. Secondly, the GP system using $s_{n,m}$ is better than using the 5×5 moving window to evolve edge detectors, in terms of the test time and $f_{0.5}$. Thirdly, the GP system using the search operators based on a single block of pixels is better than using $s_{n,m}$ only to evolve edge detectors, in terms of $f_{0.5}$. However, the GP evolved edge detectors from the former including blocks of pixels take a longer time than the latter including single pixels only to detect the BSD images. Lastly, the edge detectors evolved by GP using the search operators based on double blocks of pixels are faster than the edge detectors evolved by GP using single blocks of pixels to detect the images, while not decreasing the test performance $f_{0.5}$.

3.6 Further Discussions

This section further discusses computational cost in this GP system using different search operators, the influence of the maximum generation, and the blocks actually used in the search operators. The analysis of the GP

Table 3.8: Approximate means of training time (hours) for GP evolving edge detectors.

	without <i>IFLess</i>	using <i>IFLess</i>
$Set_{5 \times 5}$	4	9
Set_s	2	4
Set_b	51	87
Set_{tb}	20	25
$Set_{s,b}$	26	42
$Set_{s,tb}$	9	12
$Set_{s,b,tb}$	20	32

evolved edge detectors utilises basic statistical terms, such as counting occurrences, means and standard deviations. These terms are expected to explain the behaviours of the GP evolved edge detectors so that there is workload reduction in comparison with manually analysing the evolved complicated programs.

3.6.1 Computational Cost

Training Time

Table 3.8 gives the means of training time (hours) of GP evolving edge detectors with the 20 subimages based on 30 runs for each setting. When *IFLess* is used, the GP system with Set_b has a very heavy computational cost, and the mean training time (87 hours) is much longer than the other settings. For evolving edge detectors with single pixels, GP using Set_s is much faster than using $Set_{5 \times 5}$. After introducing tb_{j,t,l,w,d_1,d_2} , the computational cost for using blocks of pixels is obviously decreased, and it is not strongly affected by the function *IFLess*.

Table 3.9: Occurrences of the functions and terminals in the evolved edge detectors without *IFLess*.

nodes		$Set_{5 \times 5}$	Set_s	Set_b	Set_{tb}	$Set_{s,b}$	$Set_{s,tb}$	$Set_{s,b,tb}$
total	mean	127.4	80.4	113.4	134.1	63.8	84.1	66.1
	s.d.	45.3	25.5	31.6	41.9	23.1	24.5	24.6
<i>functions</i>	mean	76.2	58.9	75.0	86.7	46.9	60.0	48.6
	s.d.	27.8	18.9	20.0	26.6	16.3	17.7	16.4
$s_{n,m}$	mean	—	29.2	—	—	18.2	24.6	19.6
	s.d.	—	10.0	—	—	6.4	7.4	6.7
$block_{t,l,w,d}$	mean	—	—	21.0	—	8.4	—	5.2
	s.d.	—	—	7.7	—	5.1	—	2.5
tb_{j,t,l,w,d_1,d_2}	mean	—	—	—	20.2	—	7.1	2.5
	s.d.	—	—	—	6.1	—	3.3	2.3
I	mean	—	12.7	21.4	27.6	10.1	13.9	11.1
	s.d.	—	5.0	7.0	9.7	5.2	4.6	5.6
i	mean	31.6	—	—	—	—	—	—
	s.d.	12.2	—	—	—	—	—	—
rnd	mean	19.6	8.9	17.0	19.8	6.9	10.2	6.4
	s.d.	8.7	4.1	6.4	8.6	3.7	4.3	4.6

Occurrences of Functions and Terminals without *IFLess*

In order to further analyse the computational cost in the 30 evolved edge detectors in each setting, the numbers of occurrences of the functions and terminals existing in the evolved edge detectors are compared in different settings.

Table 3.9 provides the means and standard deviations (s.d.) of the number of the occurrences of the functions and terminals in the 30 evolved edge detectors for each setting. Here i indicates a pixel in the 5×5 window, and it is only used in $Set_{5 \times 5}$; “—” indicates that the relevant function

or terminal is not included in the function set or terminal set; and "Total" is the size of an evolved program. There are seven interesting observations. Firstly, there are many function nodes existing in the evolved edge detectors from $Set_{5 \times 5}$, Set_b and Set_{tb} . Secondly, there are small numbers of functions in the evolved edge detectors from the settings including $s_{n,m}$, such as 46.9 ± 16.3 functions in the evolved edge detectors from $Set_{s,b}$. Thirdly, a smaller number of functions does not mean faster detecting responses. For instance, the evolved edge detectors from $Set_{s,b}$ take longer than the evolved edge detectors from Set_s and $Set_{s,tb}$ to detect images. Fourthly, large numbers of terminals (image I or pixel i , and random constants rnd) exist in the evolved edge detectors from $Set_{5 \times 5}$, Set_b and Set_{tb} , such as 27.6 ± 9.7 images I and 19.8 ± 8.6 random constants rnd in Set_{tb} . Fifthly, the number of occurrences of the search operators using $block_{t,l,w,d}$ and/or tb_{j,t,l,w,d_1,d_2} is obviously decreased when $s_{n,m}$ is used. For example, the average number of search operators using blocks of pixels is around 20 in Set_b and Set_{tb} , and the average number of these operators is around 8 in $Set_{s,b}$, $Set_{s,tb}$ and $Set_{s,b,tb}$. Sixthly, the number of random constants rnd in a setting with $s_{n,m}$ is smaller than its occurrence number in a setting without $s_{n,m}$. Lastly, a large number of the search operators $s_{n,m}$ exist in the evolved edge detectors from $Set_{s,b}$, $Set_{s,tb}$ and $Set_{s,b,tb}$, being slightly smaller than its number of occurrences in Set_s .

From the comparisons among Set_s , $Set_{s,b}$, $Set_{s,tb}$ and $Set_{s,b,tb}$, the number of function $s_{n,m}$ keeps in a small range when $s_{n,m}$ is utilised to construct edge detectors. It seems that $s_{n,m}$ is not strongly affected by the other two search operators. Also, from the comparison of the number of occurrences of functions in the evolved edge detectors using $s_{n,m}$ and without $s_{n,m}$, it suggests that $s_{n,m}$ is important to keep program size in a suitable range and not too large.

The average size of an evolved edge detector from $Set_{5 \times 5}$ is much larger than the average size of an evolved edge detector from Set_s . The ratio of the size of the evolved programs from $Set_{5 \times 5}$ (1.58) to Set_s is close

to the ratio of the mean of the time to detect a BSD image from $Set_{5 \times 5}$ to Set_s (1.89), which suggests that the program size complexity could be used to indicate its computational complexity when the edge detectors only use single pixels. However, when evolved edge detectors include the search operators $block_{t,l,w,d}$ and/or tb_{j,t,l,w,d_1,d_2} , the number of the functions in the evolved edge detectors cannot indicate the computational cost for detecting images. For example, the size of the evolved programs from $Set_{s,tb}$ is 84.1 ± 24.5 , smaller than the size of the evolved programs from $Set_{s,b,tb}$. However, the mean of the test time from the former is shorter than the latter to detect a BSD image. The reason for the difference is that the block size in each search operator $block_{t,l,w,d}$ or tb_{j,t,l,w,d_1,d_2} might be different, and the computational cost in these edge detectors mainly comes from the calculations on blocks of pixels.

Note that the performance on computational cost is not strongly connected to the performance on detection accuracy $f_{0.5}$. An example evolved edge detector from $Set_{s,tb}$ without *IFLess* has $f_{0.5} = 0.2893$, and only takes 0.04 seconds for detecting a BSD image. In the relevant program, the number of occurrences of tb_{j,t,l,w,d_1,d_2} is only three and the number of occurrences of $s_{n,m}$ is 17. The test performance $f_{0.5}$ is close to the mean in $Set_{s,tb}$, and the numbers of occurrences of tb_{j,t,l,w,d_1,d_2} and $s_{n,m}$ in the example edge detector are lower than the relevant means. It seems that the test performance on complexity and detection accuracy ($f_{0.5}$) of a program might not be a trade-off when the program has several search operators. However, if directly restricting a small number of search operators tb_{j,t,l,w,d_1,d_2} and $block_{t,l,w,d}$, the difficulty for finding good edge detectors is increased. How to find good edge detectors with low complexity is very challenging in the GP system, and needs to be further investigated.

Occurrences of Functions and Terminals using *IFLess*

The numbers of occurrences of the functions and terminals existing in the evolved edge detectors are given in Table 3.10 when the GP system em-

Table 3.10: Occurrences of the functions and terminals in the evolved edge detectors using *IFLess*.

nodes		$Set_{5 \times 5}$	Set_s	Set_b	Set_{tb}	$Set_{s,b}$	$Set_{s,tb}$	$Set_{s,b,tb}$
total	mean	341.2	176.6	277.1	254.7	131.7	135.3	130.4
	s.d.	223.2	80.7	131.9	101.7	55	68.5	54.5
<i>functions</i>	mean	178.1	116.4	157.4	140.8	88.5	87.2	88.0
	s.d.	120.3	50.9	72.9	54.1	37.2	43.2	37.7
$s_{n,m}$	mean	—	55.00	—	—	33.3	32.2	33.1
	s.d.	—	23.0	—	—	16.4	17.7	17.5
$block_{t,l,w,d}$	mean	—	—	32.2	—	14.0	—	9.0
	s.d.	—	—	9.7	—	8.5	—	5.3
tb_{j,t,l,w,d_1,d_2}	mean	—	—	—	27.5	—	10.2	4.2
	s.d.	—	—	—	8.1	—	6.3	4.5
<i>IFLess</i>	mean	21.5	7.3	16.5	16.1	5.2	5.9	5.3
	s.d.	15.7	5.1	8.9	9.2	3.8	4.8	4.1
I	mean	—	31.5	62.6	61.5	26.0	26.3	24.7
	s.d.	—	16.8	30.7	26.8	11.3	14.8	10.6
i	mean	86.6	—	—	—	—	—	—
	s.d.	54.6	—	—	—	—	—	—
rnd	mean	76.5	28.8	57.0	52.3	17.3	21.8	17.7
	s.d.	50.3	16.2	31.7	24.7	10.3	13.2	10.9

plays function *IFLess*. The numbers of occurrences of functions and terminals for the evolved edge detectors using *IFLess* are much larger than those without using *IFLess* in each setting. From Table 3.2 and Table 3.6, the computational cost for evolved edge detectors using *IFLess* are higher than the evolved edge detectors without *IFLess*. From the number of occurrences of functions in the evolved edge detectors using *IFLess*, the

high computational cost is shown for detecting images. Although the test time cannot be indicated by the number of occurrences of the search operators, the comparison between the same search operators in the same setting still indicates approximate test times for detecting images.

An interesting observation is that there are a few *IFLess* functions (around six) in the evolved edge detectors when the function set includes $s_{n,m}$. From Table 3.6, function *IFLess* does not improve the detection accuracy. It is possible that *IFLess* in all found edge detectors has very limited contribution for constructing edge features, compared with function $s_{n,m}$.

When each argument of *IFLess* including I is similar to a subtree without *IFLess*, the approximately expected number of occurrences of terminal I in the evolved edge detectors should be four times that of the evolved edge detectors without *IFLess*. The number of occurrences of terminal I or i between using *IFLess* (in Table 3.10) is raised to three times as against without *IFLess* (in Table 3.9).

Another interesting observation is that the number of occurrences of random constants rnd is steeply raised, but the number of occurrences of tb_{j,t,l,w,d_1,d_2} is not increased too much. From the analysis of computational cost, it is not a good suggestion to select *IFLess* in the function set.

3.6.2 Number of Generations

In order to check whether the evolved edge detectors are over-trained, the best solutions from generation 120 in each setting are tested. Since the best solutions at half of the maximum generation 200 might not be close to the final solutions, a generation larger than the half, namely 120, is chosen. Table 3.11 gives the test performance on the evolved edge detectors without *IFLess*. The p -values are from the two-sample t -tests of the comparisons between the best solutions at generation 120 and generation 200 without *IFLess*. From these p -values, it is found that these best solutions at generation 120 are not significantly different from the best solutions at

Table 3.11: Test performance (mean $\pm$ standard deviation) of GP edge detectors using the maximum generation 120 and without *IFLess* on the 100 BSD test images.

	$f_{0.5}$	maximum ($f_{0.5}$)	times (seconds)	p -value
$Set_{5 \times 5}$	0.2554 ± 0.0055	0.2634	0.0383 ± 0.0125	0.4839
Set_s	0.2597 ± 0.0145	0.2824	0.0218 ± 0.0090	0.2877
Set_b	0.2781 ± 0.0167	0.3110	0.5297 ± 0.2274	0.1470
Set_{tb}	0.2752 ± 0.0148	0.2954	0.2040 ± 0.0772	0.2205
$Set_{s,b}$	0.2963 ± 0.0159	0.3241	0.2795 ± 0.1590	0.8048
$Set_{s,tb}$	0.2865 ± 0.0102	0.3033	0.0887 ± 0.0496	0.2715
$Set_{s,b,tb}$	0.2881 ± 0.0130	0.3109	0.2364 ± 0.1506	0.2658

generation 200. Only $Set_{s,b}$ decreases the mean of $f_{0.5}$ a bit for the best solutions from generation 120 to 200. Compared with the maximum $f_{0.5}$ of the evolved edge detectors at the generation 200 in each setting, the maximum $f_{0.5}$ at generation 120 (maximum) is very close.

An interesting observation here is the computational cost. Except for Set_s and $Set_{s,b,tb}$, the means and standard deviations of the test times for the evolved edge detectors (at generation 120) to detect BSD images are smaller than the means and standard deviations of the test times for the evolved edge detectors at generation 200. From the analysis of the occurrences of the functions, it seems that the program size for the best solutions in all settings, except for Set_s and $Set_{s,b,tb}$, is increased while its fitness has slight increase (being close to a fixed value).

Table 3.12 gives the numbers of occurrences of the functions and terminals in the evolved edge detectors at generation 120. Comparing with those numbers in Table 3.9, these numbers in Table 3.12 indicate the similar change occurrences related to the test time for the best solutions between the generations 120 and 200. For example, the test times for Set_s

Table 3.12: Occurrences of the functions and terminals in the evolved edge detectors without *IFLess* and maximum generation 120.

nodes		$Set_{5 \times 5}$	Set_s	Set_b	Set_{tb}	$Set_{s,b}$	$Set_{s,tb}$	$Set_{s,b,tb}$
total	mean	122.5	81.9	100.6	119.1	61.0	76.3	67.8
	s.d.	45.4	20.7	33.6	38.9	23.9	20.2	23.5
<i>functions</i>	mean	73.8	59.2	66.7	77.6	45.6	54.5	50.0
	s.d.	28.3	14.7	21.3	25.4	17.5	14.3	16.4
$s_{n,m}$	mean	—	29.3	—	—	19.3	22.5	18.3
	s.d.	—	7.7	—	—	8.2	6.2	6.2
$block_{t,l,w,d}$	mean	—	—	18.6	—	7.8	—	5.5
	s.d.	—	—	7.7	—	3.2	—	4.1
tb_{j,t,l,w,d_1,d_2}	mean	—	—	—	17.4	—	6.3	2.7
	s.d.	—	—	—	5.9	—	3.0	2.7
I	mean	—	12.9	19.6	23.7	10.4	12.5	10.5
	s.d.	—	5.0	7.5	7.3	5.0	3.8	4.7
i	mean	29.5	—	—	—	—	—	—
	s.d.	10.8	—	—	—	—	—	—
rnd	mean	19.2	8.9	14.3	17.8	5.0	9.1	7.6
	s.d.	9.1	4.1	6.8	9.0	3.2	4.5	4.0

using the maximum generation 200 are very similar to the times for the best solutions at generation 120, and the relevant numbers of occurrences of the functions and terminals are very close in Table 3.9 and Table 3.12. The number of the total functions in the best solutions at generation 200 is 58.9 ± 18.9 , and the relevant number at generation 120 is 59.2 ± 14.7 . The test time on a BSD image for the best solutions at generation 200 is 0.0213 ± 0.0093 , and the relevant test time at generation 120 is 0.0218 ± 0.0090 . For $Set_{5 \times 5}$, Set_b , Set_{tb} , $Set_{s,b}$ and $Set_{s,tb}$, there are fewer occurrences of the functions and terminals in the best solutions at generation 120, compared with

the relevant numbers at generation 200. Also, there are shorter test times on a BSD image for the best solutions at generation 120, compared with the best solutions at generation 200. $Set_{s,b,tb}$ has more occurrences of the functions and terminals in the best solutions at generation 120 than generation 200, and longer test time on a BSD image. Therefore, analysis of occurrences of the functions and the terminals shows that the program size for the best solutions is increased in all settings, except for Set_s and $Set_{s,b,tb}$.

From the comparisons of test performances on detection accuracy ($f_{0.5}$) and test time on a BSD image, there might be slight over-training in $Set_{s,b}$. However, the best evolved program grows in size when the current best fitness is close to a constant in five of the seven settings. Also, all the standard deviations of the test time in Table 3.12, except for $Set_{s,b,tb}$, are smaller than relevant standard deviations in Table 3.9. It seems that all the best candidate solutions have similar computational cost when their fitness values are very close to a convergent value. Since the best program grows in size in the convergence stage or near to it, a restriction on the program size is suggested to be integrated into the fitness evaluation. In the convergence stage, the comparisons between the generations 120 and 200 show that the program size and the detection accuracy is not a trade-off after several search operators exist in the program.

From the analysis of the different phenomena from the best solutions at generations 120 and 200, it is not good to use the difference of the fitness values of individuals to discriminate whether an individual is better than others. A very slight improvement (detection accuracy) for the best solution might lead to a large increase of the program size. Obviously, the best program at generation 120 is better than the best program at generation 200 for detecting images. However, how to evaluate individuals in the late evolving stage is complicated. Statistical techniques might be helpful to discriminate whether an individual is significantly better than others in this stage.

Clearly, $Set_{s,b,tb}$ is different from the other settings in the changes between generations 120 and 200. The number of the functions of the best solutions in $Set_{s,b,tb}$ at generation 120 is slightly larger than the relevant number at generation 200. The main difference comes from the numbers of function $s_{n,m}$. It is possible that there are lots of similar combinations of the three search operators (in terms of $f_{0.5}$), but some combinations including $s_{n,m}$ are slightly better than others. After several generations, more $s_{n,m}$ are selected in the current best solution. Since the search operator $s_{n,m}$ is faster than the other operators, the test time is not decreased obviously.

3.6.3 Blocks Used by Search Operators

The blocks used in the search operators $block_{t,l,w,d}$ and tb_{j,t,l,w,d_1,d_2} are investigated from their shape and size, and transformation approach t .

Block Size

Figure 3.13 reveals that the total occurrences of combinations $w \times l$ in the evolved edge detectors when *IFless* is not used. Since each search operator tb_{j,t,l,w,d_1,d_2} includes two blocks with the same size, the number of occurrences of blocks $w \times l$ from Set_{tb} is obviously larger than the others. The largest number of occurrences (94) in Set_{tb} comes from 2×4 , and it is obviously larger than others. It seems that a very large block or a very small block might not be a good selection in the search operator. In Set_b , the size with the highest occurrences (43) is 3×3 , but some combinations have more than 30 occurrences, such as 2×3 and 4×3 .

From an overview, it is hard to find obviously interesting patterns from these different block sizes. Using these occurrences and these block sizes, the overall average sizes for these blocks used in the evolved edge detectors from Set_b , Set_{sb} , $Set_{s,b}$, $Set_{s,tb}$ and $Set_{s,b,tb}$ are 18.7 ± 14.1 , 22.3 ± 13.3 , 20.5 ± 13.1 , 25.9 ± 14.8 and 22.0 ± 14.0 , respectively. The expected size from the original blocks based on uniform distribution (considered as the

Figure 3.13: Occurrences of combinations ($w \times l$ in the evolved edge detectors).

same occurrence for each block size) is 24.0 ± 15.1 . Except for $Set_{s,tb}$, the estimated sample average size for using a block is slightly decreased. The estimated sample mean (block size) in $Set_{s,tb}$ is slightly larger than the original expected value. How strong the block size is to affect the detection performance needs further investigation.

The estimated sample mean (block size) in $Set_{s,tb}$ is larger than the others, but the evolved edge detectors are faster than the latter. From Table 3.9, the number of occurrences of random constants rnd is larger than the numbers in $Set_{s,b}$ and $Set_{s,b,tb}$. A potential reason is that tb_{j,t,l,w,d_1,d_2} has a random constant rnd as its argument.

Table 3.13: Total occurrences of approaches used in blocks (in the 30 evolved edge detectors without *IFLess*).

		Set_b	Set_{tb}	$Set_{s,b}$	$Set_{s,tb}$	$Set_{s,b,tb}$
$block_{t,l,w,d}$	Mean	293	—	92	—	54
	SD	336	—	159	—	102
tb_{j,t,l,w,d_1,d_2}	Minus	—	307	—	120	26
	Division	—	299	—	92	49

Approaches to Transforming a Set of Pixels to a Single Variable

Table 3.13 lists the occurrences of the approaches used in the search operators $block_{t,l,w,d}$ and tb_{j,t,l,w,d_1,d_2} . Here "SD" is the standard deviation of the intensities of the pixels in a block. To distinguish, "s.d" is used for the numbers of occurrences of the functions and terminals, "SD" is an approach to transforming a set of pixels to a single variable. It is found that the standard deviation has obviously higher occurrences than the mean in the search operator $block_{t,l,w,d}$. From the images detected by the evolved edge detectors using blocks of pixels, a potential reason is that the different areas, especially including textures, have different standard deviations.

However, it is hard to claim which combination method (subtraction – or division $\div$) is better to transform two blocks of pixels as a single variable in edge feature extraction. From the total of the occurrences of subtraction (453) and division (440) in Set_{tb} , $Set_{s,tb}$ and $Set_{s,b,tb}$, it seems that they might not have a major difference.

3.7 Chapter Summary

The goal of this chapter was to investigate the use of GP to construct low-level edge detectors using the three search operators when the prior

knowledge is ground truth only. Via using full images as input, the three search operators are developed to search neighbours. The experiment on the BSD image dataset showed that the goal was successfully achieved.

Section 3.2 proposed a general GP system for evolving edge detectors based on full images. In order to find neighbourhoods for detecting pixels as edge points or non-edge points, three search operators based on single pixels and blocks of pixels in Section 3.3 were employed in the function set.

The results of the experiments on the 100 BSD test images in Section 3.5 show that the evolved edge detectors using blocks of pixels are significantly better than the evolved edge detectors using single pixels only, in terms of detection accuracy. For the test time of detecting a BSD image, the evolved edge detectors using single pixels are faster than the evolved edge detectors using blocks of pixels. Besides, the evolved edge detectors using the single pixel search operator on full images are faster than the evolved edge detectors using a moving window. The detected images show that the evolved edge detectors using blocks of pixels can suppress textures, which is a main problem for the edge detectors with single pixels.

After proposing the search operator combining two blocks of pixels, the computational cost for the evolved edge detectors was reduced, and the detection accuracy did not decrease. Therefore, reducing the search space on blocks of pixels is efficient to improve the performance of the evolved detectors, in terms of the computational cost.

A discontinuous function *IFLess* (combining *IF* and $<$) has been employed in the function set, and it is found that the evolved program size is increased obviously, but their detection accuracies are not improved. It may be difficult for the GP system to use *IFLess* to find better edge detectors than without *IFLess* because of the large search space existing in using *IFLess*.

Also, the evolved edge detectors using single pixels have significantly better detection accuracy than the Sobel edge detector, but is not signifi-

cantly better than the Gaussian gradient. The evolved edge detectors using blocks of pixels have significantly better detection accuracy than the Gaussian gradient.

From the analysis of the evolved edge detectors, the frequencies of occurrences of the functions and terminals are helpful to understand the computational cost and find important factors from the search operators. However, the analysis here is still simple, and more work needs to be done in the future. For example, it would be interesting to automatically simplify evolved edge detectors and extract combinations of neighbours (in order to find their neighbourhood).

The best solution has been investigated at two different generations. The relationship between the size and fitness of the best program has been discussed. When the best program does not have obvious improvement of its fitness, a restriction on the program size is needed. Also, the criterion for discriminating whether a program is better than another program should consider statistical test techniques.

From the analysis of the block sizes used in the search operators in Section 3.6, the search operator using two blocks of pixels has obvious high occurrences for choosing two 2×4 blocks when only the search operator is used in the GP system. From the distribution of the numbers of occurrences of the selected block sizes, it is difficult to find a suitable range of the block size for constructing good edge detectors.

Since the fitness function $f_{0.5}$ is based on all pixels across images, the detection accuracy for each image is not clear. In the following chapter, the evaluation on detected images from different views (pixels or images) will be investigated. Also, different evaluation criteria apart from detection accuracy and localisation will be looked into.

Chapter 4

New Fitness Functions for Low-level Feature Construction

4.1 Introduction

The fitness function in GP is very important in guiding the evolution of edge detectors for natural images. *F*-measure [123] and FOM [152] (see Section 2.1.4 on page 36) are two popular methods for evaluating the performance of edge detectors, but there are very few reports in the literature using the two methods from the *pixel* view and the *image* view. Here, a fitness function from the pixel view evaluates the performance on detected results based on all pixels across all different images, which only includes a single stage. A fitness function from the image view is a two stage process. In the first stage, the performance on each detected full image is evaluated. The second stage combines the performance of detected full images together using a function, such as the arithmetic mean.

F-measure mainly focuses on the accuracy of finding true edge points (as a proportion of the total number of true edge points) and the accuracy of predicting true edge points (in all numbers of predictions). Generally, the evaluation comes from the pixel view. Performance on each full image is not clear because there is no information on full images. Another prob-

lem is that F -measure fails to present the similarity between ground truth and predictions, such as a true boundary of one object and the detected shapes.

When the tolerance distance (offset) from a predicted edge point to a true edge point is allowed, matching predicted edge points to true edge points is a typical *assignment problem* [60]. In the assignment problem, each predicted edge point can only be assigned to a single true edge point and vice versa, which is called *one-to-one assignment*. If a predicted edge point cannot find a relative true edge point within the tolerance distance, the predicted point is a false detection. The complexity of the best algorithm for solving the assignment problem in the literature is $O(N_I \log(N_I))$ [60, 123], where N_I is the problem size. For a single image i , N_I is the number of all pixels. If the tolerance distance is considered in F -measure, the computational cost of using F -measure as a fitness function in the training stage will be very heavy.

In order to reduce the computational cost of matching predictions with ground truth, relaxing the restriction of one-to-one assignment has been used in FOM. FOM allows to match multiple predicted edge points to a single true edge point. In FOM, a predicted edge point is matched to the nearest true edge point, which has much lower computational cost than one-to-one assignment. However, FOM has some drawbacks from *type II* errors (false negatives, see Table 2.1 on page 35) [9]. FOM is seldom used as an evaluation criterion in the learning stage, but it is still chosen for comparing the performance of edge detectors [164]. A potential reason of not using FOM in the learning stage is that a common learning algorithm usually uses a set of sampled pixels to train edge detectors, and there is not much information from full detected images. The proposed GP system in the previous chapter is based on full images, so it is easy to consider information from full training images. If the tolerance distance is considered in the training stage, it is necessary to investigate the use of FOM and its variants in the learning stage.

In order to address drawbacks in FOM, *type II* errors should be considered in the fitness function so that there is a balance between *type I* errors (false alarms) and *type II* errors. An existing modification of FOM [149] focuses on the matching direction. This FOM variant only uses distances from ground truth to predicted edges and includes a factor about false (unmatched) edge points. The FOM variant was only used to test artificial images. Whether it can be used for natural images needs to be validated. Since it mainly addressed the matching direction from ground truth to predicted edge points, it might not be good at addressing the *type I* errors. It is necessary to investigate both matching directions between predictions and ground truth.

In addition to the evaluation view (from pixels or images) and the detection localisation, the observations of a GP program need to be considered when they are transformed to soft edge maps. Here, observations are the spread of GP program outputs. For instance, a program $x + 2$ has observation 3 when x is equal to 1. If the program is used to discriminate whether x is larger than -2 , the answer is “yes” when its observation is larger than the fixed threshold 0.

In the GP system, each pixel corresponds to an observation. Each input image has a set of observations, which is the output of a GP edge detector. When a fixed threshold is used to discriminate one pixel as an edge point or a non-edge point, the distance from the observation to the threshold is not considered. However, if most observations are very close to the threshold and only a few of the observations are far away from the threshold, they are not good for obtaining soft edge maps because the observations around the threshold belong to the same grayscale level if a linear transformation is applied. In order to obtain good soft edge maps, a transformation of the observations of GP edge detectors needs to be investigated. When a fitness function considers the distance from observations to a fixed threshold, whether a binary edge map using different thresholds on a transformation of the observations of a GP edge detector is changed

sharply needs to be investigated. Also, whether the fixed threshold is the best value for the evolved edge detectors discriminating pixels as edge points should be investigated.

4.1.1 Chapter Goal

The goal of this chapter is to investigate fitness functions in GP for automatically searching pixels to construct low-level edge detectors based on different evaluation criteria. First of all, the evaluation criteria will be investigated from the pixel view and the image view. *F*-measure as detection accuracy and FOM as localisation accuracy are employed. To evaluate the performance of an edge detector, evaluation methods are developed from the pixel view and the image view. Secondly, matching directions in FOM will be investigated for performance improvement. Lastly, evaluation criteria based on the spread of observations of evolved edge detectors around a fixed threshold will be investigated so that good soft edge maps can be obtained from the evolved edge detectors. Specifically, the following research objectives will be investigated.

- Whether there are differences for the evaluation methods between the pixel view and the image view.
- Which technique, the *F*-measure or FOM, is better to evaluate an edge detector's performance.
- Whether a combination of both matching directions in FOM can improve detection performance.
- Whether the fitness functions based on spread of observations of evolved edge detectors can be used to obtain better soft edge maps than the fitness function without restriction on the spread.

4.1.2 Chapter Organisation

In the remainder of this chapter, the second, third and fourth sections investigate fitness functions based on three different criteria. The second section introduces evaluations from the pixel view and the image view. The third section addresses the problem of matching directions between predictions and ground truth used in evaluation. The fourth section proposes evaluations based on the spread of observations of evolved edge detectors around a fixed threshold. The fifth section gives the experiment settings. The sixth section presents the experiment results with discussions. The seventh section provides further discussions. The last section gives a summary of this chapter.

4.2 Evaluations from Pixel and Image Views

To indicate the similarity between the edge maps detected by GP edge detectors and the relevant ground truth, the fitness functions using detection accuracy and localisation will be investigated in this section from the pixel view and the image view. F -measure and FOM will be used to develop new fitness functions.

4.2.1 F -measure

F -measure has been popularly applied to edge detection performance evaluation based on balancing recall and precision [39, 38, 53, 81, 123]. Therefore, F -measure is employed to develop a fitness function in the GP system. However, F -measure normally uses all pixels across all different images to do the evaluation; the accuracy of individual full images is not clear from it. Thus, we investigate how to use F -measure to develop fitness functions from both (pixel and image) views.

When the accuracies of detected images are indicated by F -measure, a group of accuracies from each detected image can be indicated by different

means, such as the arithmetic mean and geometric mean. Different means present a summary (central tendency) for a group of numbers. In the image view for F -measure, $f_{i,0.5}$ (see Equation (4.1)) presents the detection accuracy for image i with $f_{0.5}$, where recall P_{r_i} and precision P_{p_i} are for all pixels in image i . The arithmetic mean and geometric mean are chosen to evaluate the detection accuracy from the image view. The fitness function $f_{F_{avg}}$ using the arithmetic mean is defined in Equation (4.2), where N_{img} is the number of individual full images. The geometric mean can indicate some details for each image, and the fitness function $f_{F_{gm}}$ using the geometric mean is defined in Equation (4.3).

$$f_{i,0.5} = \frac{2P_{r_i}P_{p_i}}{P_{r_i} + P_{p_i}} \quad (4.1)$$

$$f_{F_{avg}} = \frac{1}{N_{img}} \sum_{i=1}^{N_{img}} f_{i,0.5} \quad (4.2)$$

$$f_{F_{gm}} = \left(\prod_{i=1}^{N_{img}} f_{i,0.5} \right)^{\frac{1}{N_{img}}} \quad (4.3)$$

Since $f_{i,0.5}$ ranges from 0 to 1, $f_{F_{gm}}$ is typically smaller than $f_{F_{avg}}$ for the same set of images. However, when the sum of all image $f_{i,0.5}$ values is fixed, $f_{F_{avg}}$ fails to indicate any difference from these images. Suppose there are four training images and three edge detectors D_1 , D_2 and D_3 with the same $f_{0.5}$ value for them. The accuracies of detected images from D_1 , D_2 and D_3 are shown in Figure 4.1. Here, the horizontal axis represents images, the vertical axis represents $f_{i,0.5}$ values, and image 4 has a larger number of true edge points than images 1, 2, and 3. Although the detectors D_1 , D_2 and D_3 have the same detection performance based on the overall results of the four images, the detection performances of the three detectors on each image are very different. If the performance is based on the number of $f_{i,0.5}$ values larger than constant f_c , the detector D_2 is better than D_1 , which can be indicated by $f_{F_{avg}}$. If D_2 has the same sum of the four $f_{i,0.5}$ values as D_3 , the detector D_3 might have much higher gen-

Figure 4.1: Detection accuracies $f_{i,0.5}$ values for four different images of three detectors.

eral ability to detect edges than the detector D_2 , which can be indicated by $f_{F_{gm}}$, not $f_{F_{avg}}$.

A similar evaluation to $f_{F_{avg}}$ and $f_{F_{gm}}$ is the square root of sum of squares $f_{F_{qm}}$, defined in Equation (4.4). To investigate the influence of the square, a new fitness function $f_{F_{qf}}$ is defined in Equation (4.5). $f_{F_{qf}}$ is based on the pixel view.

$$f_{F_{qm}} = \sqrt{\frac{1}{N_{img}} \sum_{i=1}^{N_{img}} f_{i,0.5}^2} \quad (4.4)$$

$$f_{F_{qf}} = \frac{2P_r^2 P_p^2}{P_r^2 + P_p^2} \quad (4.5)$$

In summary, $f_{0.5}$ and $f_{F_{qf}}$ describe the global detection performance based on overall pixels across all images in the training set, but $f_{F_{qm}}$, $f_{F_{avg}}$ and $f_{F_{gm}}$ summarise the detection performance with the accuracies of individual full images.

4.2.2 FOM

Since the potential drawbacks in FOM might not appear in the evaluation of detection results, FOM is employed as a fitness function f_{FOM} (see Equation (2.14) on page 37) to check whether it is suitable for evolving reasonable detectors for natural images.

From the image view, several fitness functions with FOM are investigated. Here, f_{FOM_i} is the FOM of image i , and it is defined in Equation (4.6), where N_{T_i} is the number of true edge points in image i , N_{P_i} is the number of predicted edge points in image i , Set_{P_i} is the set of predicted edge points in image i , and α_d and dt_1 are the same as in f_{FOM} .

$$f_{FOM_i} = \frac{1}{\max\{N_{T_i}, N_{P_i}\}} \sum_{j \in Set_{P_i}} \frac{1}{1 + \alpha_d dt_1^2(j)} \quad (4.6)$$

The fitness function using the arithmetic mean $f_{FOM_{avg}}$ is defined in Equation (4.7), the fitness function using the geometric mean $f_{FOM_{gm}}$ is defined in Equation (4.8), and the fitness function using the square root of sum of squares is defined in Equation (4.9).

$$f_{FOM_{avg}} = \frac{1}{N_{img}} \sum_{i=1}^{N_{img}} f_{FOM_i} \quad (4.7)$$

$$f_{FOM_{gm}} = \left(\prod_{i=1}^{N_{img}} f_{FOM_i} \right)^{\frac{1}{N_{img}}} \quad (4.8)$$

$$f_{FOM_{qm}} = \sqrt{\frac{1}{N_{img}} \sum_{i=1}^{N_{img}} f_{FOM_i}^2} \quad (4.9)$$

4.3 Matching Directions in FOM

The standard FOM employs the matching direction from predictions to ground truth, which is considered as a *precision variant* because of evaluating the accuracy of predictions. The precision variant requires the number of predicted edge points to be close to the number of true edge points. The matching direction from ground truth to predictions can be considered as a *recall variant* because of evaluating the ability to find true edge points. The recall variant expects as many predictions as possible for a high value, but the precision variant expects that good detection should have predicted edge points not many more than true edge points.

Figure 4.2: The same FOM value with different detection results.

FOM evaluates edge detectors in terms of detection localisation, but there are some drawbacks [9]. For instance, Figure 4.2 shows two different detected edge maps with the same FOM value. However the detected result in Figure 4.2 (a) has overlap in only one true line and misses the top line. We can see that FOM may not reflect some *type II* errors.

To investigate the influence of both matching directions, an existing FOM variant using the single matching direction from ground truth to predictions is employed as a fitness function. Also, a proposed combination of both directions is utilised as a new fitness function.

4.3.1 Fitness Function with a Single Direction Matching

In order to improve the sensitivity to *type II* errors, a modification of FOM will focus on the matching direction. An existing FOM variant only uses distances from ground truth to predicted edges, and includes a factor about false (unmatched) edge points [149]. This FOM variant was only used to test artificial images in [149], but it is directly used as a new fitness function ($f_{FOM,gt}$) in this chapter to test natural images. $f_{FOM,gt}$ is defined in Equation (4.10), where N_{FM} is the number of false edge points, β is a weight for penalising N_{FM} , and $dt_2(i)$ is the distance from a true edge point i to the nearest predicted edge point. Note that the matching directions in dt_1 and dt_2 are different. In [149], it was suggested that β be set to

1. Here α_d is a weight factor on the distance. In $f_{FOM,gt}$, α_d and β are used to balance *type I* errors and *type II* errors, but the relationship between α_d and β is *not* clear. The second part in $f_{FOM,gt}$ is sensitive to β . When β is large, the detection results will mainly focus on recall (the number of true positives).

$$f_{FOM,gt} = \left(\frac{1}{N_T} \sum_{i=1}^{N_T} \frac{1}{1 + \alpha_d dt_2^2(i)} \right) \left(\frac{1}{1 + \frac{\beta N_{FM}}{N_T}} \right) \quad (4.10)$$

Fitness function $f_{FOM,gt}$ matches a true edge point to the nearest predicted (unmatched) edge point, then marks the nearest predicted edge point as matched edge point. Each true edge point is only allowed to match a predicted edge point. The predicted edge points left unmatched are considered as false matching points. After the matching operation, $f_{FOM,gt}$ finds unmatched predicted edge points. Therefore, $f_{FOM,gt}$ has more calculation than the standard FOM.

4.3.2 A Proposed Combination of FOM and *F*-measure

A new FOM variant, inspired by the Hausdorff distance [79], is developed here by taking into account both distances between the ground truth and the predicted edges (both $dt_1(i)$ and $dt_2(i)$). In order to fairly balance the measurement between *type I* errors and *type II* errors, $f_{FOM,\alpha}$ is designed based on the balance between $dt_1(i)$ and $dt_2(i)$ with the weight factor α used in *F*-measure. Function $f_{FOM,\alpha}$ is defined by Equation (4.13), where, $f_{FOM,pre}$ (see Equation (4.11)) and $f_{FOM,rec}$ (see Equation (4.12)) are precision and recall by considering both matching directions between predictions and ground truth. Function $f_{FOM,\alpha}$ considers both directions and merges them together with the weight factor α .

$$f_{FOM,pre} = \frac{1}{N_T} \sum_{i=1}^{N_T} \frac{1}{1 + \alpha_d dt_2^2(i)} \quad (4.11)$$

$$f_{FOM,rec} = \frac{1}{N_P} \sum_{i=1}^{N_P} \frac{1}{1 + \alpha_d dt_1^2(i)} \quad (4.12)$$

$$f_{FOM,\alpha} = \frac{f_{FOM,rec} f_{FOM,pre}}{\alpha f_{FOM,rec} + (1 - \alpha) f_{FOM,pre}} \quad (4.13)$$

An existing FOM variant considered the errors from the differences of both matching directions [9], and its definition $f_{FOM,\delta}$ is given in Equation (4.14), where N_I is the total number of pixels. Compared with $f_{FOM,\delta}$, $f_{FOM,\alpha}$ only considers the edge points (the true edge points and predicted edge points) rather than all pixels of images, which means that the computational cost for $f_{FOM,\alpha}$ is lower than $f_{FOM,\delta}$ (using all pixels in an image) because the number of (true and predicted) edge points in one image is usually much lower than the total number of pixels in the image. Since $f_{FOM,\delta}$ includes both matching directions for each pixel, it is not good to be used as a fitness function in a GP system because of its heavy computational cost.

$$f_{FOM,\delta} = \sum_{j=0}^{N_I} ((dt_1(j) - dt_2(j))^2) \quad (4.14)$$

4.3.3 Comparisons Among Different FOM Variants

Figure 4.3 gives three examples of different predictions against ground truth. Figure 4.3 (a) is the ground truth including two lines, (b) “One Edge” only finds one edge with thick detection, (c) “Missing” misses two true edge points, and (d) “Offset” has one pixel distance offset from the detected result to the ground truth. As can be seen, the detected result only including one line is worse than the detected result missing two true edge points. If there is a tolerance for one pixel offset from predicted lines to true lines, the detected result with one pixel offset is better than the detected result only including one line. In general, it is hard to claim which detected result in Figure 4.3 (c) or (d) is better. From the line topology, Figure 4.3 (d) is better than Figure 4.3 (c). However, from the detection accuracy, Figure 4.3 (c) is better. Due to the pros and cons, a good evaluation function should not have very large difference between the situation in Figure 4.3 (c) and (d).

Figure 4.3: Three different example detected results in a 7×7 window.

To compare different FOM variants, the factor α_d is set to $\frac{1}{9}$ and the allowed offset is restricted in the range of the connected neighbour position, namely in a 3×3 window. The standard FOM and three FOM variants ($f_{FOM,gt}$, $f_{FOM,\alpha}$ and $f_{FOM,\delta}$) are used to measure the three detected results. Table 4.1 gives their evaluation values on the three predictions. Note that the ranges of f_{FOM} , $f_{FOM,gt}$ and $f_{FOM,0.5}$ ($f_{FOM,\alpha}$ with $\alpha = 0.5$) are from 0 to 1, and higher values mean better detection results. Lower values in $f_{FOM,\delta}$ mean better detection results, and the minimum value 0.0 is for the perfect matching.

For f_{FOM} , the detected result with only one edge is higher than the detected results with missing true edge points and one pixel distance offset, which wrongly indicates the measurements of the three detected results.

Table 4.1: Measurements for three example detected edges by f_{FOM} , $f_{FOM,gt}$, $f_{FOM,\alpha}$ and $f_{FOM,\delta}$.

	One Edge	Missing	Offset
f_{FOM}	0.9500	0.6667	0.7500
$f_{FOM,gt}$	0.4242	0.9667	0.9000
$f_{FOM,0.5}$	0.7622	0.9831	0.9000
$f_{FOM,\delta}$	12.7136	0.0468	11.4500

Comparing the differences among the three detected results in terms of f_{FOM} , the difference between the result missing edge points and the results with one pixel distance offset is not large.

In terms of $f_{FOM,gt}$, the result only including one edge has the lowest value, and the difference between the result with missing edge points and the result with one pixel distance offset is small, which means that $f_{FOM,gt}$ suitably indicates the two detected results. In the early stage of evolution, evolved edge detectors might not have good detection results. The detected result only including one line has very low $f_{FOM,gt}$, so it might not be good to use $f_{FOM,gt}$ to select potentially good edge detectors in the early stage.

Compared with $f_{FOM,gt}$, the proposed combination $f_{FOM,0.5}$ presents similar evaluations on the results with missing edge points and offset. However, $f_{FOM,0.5}$ does not give a very low value for the detected result only including one line. This is because $f_{FOM,0.5}$ balances both matching directions. The detected result only including one line has high precision if the overlap is allowed. $f_{FOM,0.5}$ shows the ability to select potentially good edge detectors in the early evolving stage.

For $f_{FOM,\delta}$, the result with offset is measured with a too high error value, which is close to the result only including one line. The similarity between the ground truth and predicted edge points is required to be too

high in terms of $f_{FOM,\delta}$. When all predicted edge points of a detected result have offset from true edge points, the function $f_{FOM,\delta}$ does not accept it as a good detection. Taking into account the heavy computational cost of $f_{FOM,\delta}$ and not good offset tolerance, $f_{FOM,\delta}$ is not used as a fitness function in the GP system.

4.4 Observation Spread

In general, the observations of an evolved edge detector with threshold 0 are only considered for their signs (positives or negatives). Positive values are used to discriminate the relevant pixels as edge points. When the observations are transformed to grayscale levels, images detected by the evolved edge detector are described as soft edge maps.

Threshold techniques are used to filter non-edge points from a soft edge map. Figure 4.4 shows one detected image from the BSD dataset [123] by the Sobel detector [57]. Figure 4.4(a) is the soft edge map based on the grayscale level range from 0 to 255; (b), (c) and (d) are the corresponding hard edge maps using the thresholds 1, 3 and 22 respectively. Figure 4.4(b) includes noise. In Figure 4.4(d), the edge map removes some details of the shapes of objects. Figure 4.4(c) includes the boundaries from the background and object shapes.

4.4.1 Transformation

The outputs of a GP edge detector usually only focus on the binary edge map (B); and the distances from its observations (o_i) to threshold c_{th} are not considered. Actually, it is possible that many observations are close to c_{th} , so B will change sharply when c_{th} changes a little. When a linear transformation LT (see Equation (4.15)) is used to obtain the soft edge map s_{lt} from o , there might be problems with the grayscale levels for the soft edge map s_{lt} and the separation of observations around c_{th} . In LT ,

Figure 4.4: Different edge maps for detected edges in a BSD test image by the Sobel detector.

o_{min} and o_{max} are the minimum and maximum values in o . The range of o ($o_{max} - o_{min}$) might be very large. Many pixels may be located in low grayscale levels ($o_i \ll o_{max}$) or high levels ($o_i \gg o_{min}$), so that many pixels marked as edge points or non-edge points with respect to c_{th} could be mapped into the same level.

$$s_{lt} = LT(o) = \frac{o - o_{min}}{o_{max} - o_{min}} \quad (4.15)$$

Figure 4.5 gives two observation examples on the true edge points and true non-edge points. Figure 4.5 (a) has larger range of observations than Figure 4.5 (b). Most observations in Figures 4.5 (a) and (b) are very close to threshold 0. Since Figure 4.5 (a) has a large spread, the transformation LT possibly maps these observations into the same grayscale level. Also the soft edge map transformed from Figure 4.5 (a) is sparse. Since Figure 4.5 (b) has a small spread, the observations around threshold 0 are easy to map to different grayscale levels. Due to the small range of the observations, the soft edge map obtained from LT contains many grayscale levels.

Figure 4.5: Different observation spreads for true edge points and non-edge points.

When c_{th} uses different values (from 0 to 1) to threshold $LT(o)$ to obtain different edge maps B , these edge maps are generally very different. When c_{th} is very low, B will have high recall and low precision. When c_{th} is high, B will have low recall and high precision. However, the redundancy from high recall and low precision is too high so that the suitable information is hard to find, and the confidence from low recall and high precision is not enough to extract suitable edge features. Therefore, a detection result with high recall and low precision, or low precision and high recall, is not normally helpful to find edges, and it is not worth considering for further processing.

To obtain good soft edge maps from a GP edge detector, an S-shaped (see Equation (4.16)) nonlinear transformation NT is employed to transform o so that the influence of the extrema of o are weakened. The benefit of the transformation is that the change around c_{th} is smooth, and these observations around c_{th} in a soft edge map could be separated to different levels.

$$s_{nt} = NT(o) = \frac{1}{1 + e^{-o}} \quad (4.16)$$

4.4.2 Fitness Function on Observation Spread

The standard F -measure is employed to evaluate GP edge detectors, and α is set to 0.5, following [123]. Therefore $F(f_{0.5})$ can be also described as in Equation (4.17), where N_{TP} is the total number of pixels on the edges correctly detected, N_T is the total number of pixels on the true edges and N_P is the number of pixels detected as edge points. More details about these indications can be seen from Table 2.1 in Section 2.1.4 (on page 35).

$$F = f_{0.5} = \frac{2N_{TP}}{N_T + N_P} \quad (4.17)$$

Fitness Function using a Restriction of Short Distances and Large Range

A new fitness function based on F considers penalisation on short distances from o_i to c_{th} , e.g., when the distance from o_i to c_{th} is smaller than a tolerance value c_{dt} . A true predicted edge point requires that its observation is larger than c_{dt} . Also, a prediction is considered as a predicted point for a pixel when the distance from its observation to threshold c_{th} is larger than $-c_{dt}$. The range of observations of an evolved edge detector must be in the range from c_{dmin} to c_{dmax} . If an observation of an evolved edge detector is not in the range, the evolved edge detector is heavily penalised. The new fitness function f_{F_d} is shown in Equation (4.22). Here, $N_{P,d}$ (see Equation. (4.20)) replaces the changed N_P , and $N_{P,d,i}$ (see Equation. (4.18)) is used to modify the precision so that a penalty for o_i around threshold c_{th} is added in f_{F_d} . Similar to $N_{P,d}$, $N_{TP,d}$ (see Equation. (4.21)) is modified for the true edge points.

$$N_{P,d,i} = \begin{cases} 1 & \text{if } o_i \geq c_{th} - c_{dt} \\ 0 & \text{otherwise} \end{cases} \quad (4.18)$$

$$N_{TP,d,i} = \begin{cases} 1 & \text{if } o_i \geq c_{th} + c_{dt} \text{ and } t_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.19)$$

$$N_{P,d} = \sum_i N_{P,d,i} \quad (4.20)$$

$$N_{TP,d} = \sum_i N_{TP,d,i} \quad (4.21)$$

$$f_{F_d} = \begin{cases} 0 & \text{if } o_{max} \geq c_{dmax} \quad \text{or} \quad o_{min} \leq c_{dmin} \\ \frac{2N_{TP,d}}{N_T + N_{P,d}} & \text{otherwise} \end{cases} \quad (4.22)$$

Fitness Function Based on Distance Spread

A new fitness function, which considers all distances from $NT(o)$ to c_{th} , is designed to keep $NT(o_i)$ as far away from c_{th} as possible, and the new fitness function f_{F_s} is shown in Equation (4.25), where $N_{P,s}$ is the modified N_P , $N_{P,s,i}$ is for the pixel i , and $w_{s,1}$, $w_{s,2}$, $w_{s,3}$ are positive weights. Here, $s_{nt,i}$ (see Equation (4.16), after the S-shaped transformation for the pixel i) is directly included into the precision $N_{P,s,i}$. The reason for using the S-shaped transformation is that good detectors only focus on minimising $N_{P,s}$ if the distance from o_i to c_{th} is directly used and the range of o is large.

$$N_{P,s,i} = \begin{cases} 1 + w_{s,1}s_{nt,i} & \text{if } NT(o_i) > c_{th} \quad \text{and} \quad t_i = 0 \\ w_{s,2}s_{nt,i} & \text{if } NT(o_i) \leq c_{th} \quad \text{and} \quad t_i \neq 0 \\ \frac{w_{s,3}+1}{1+w_{s,3}s_{nt,i}} & \text{if } NT(o_i) > c_{th} \quad \text{and} \quad t_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.23)$$

$$N_{P,s} = \sum_i N_{P,s,i} \quad (4.24)$$

$$f_{F_s} = \frac{2N_{TP}}{N_T + N_{P,s}} \quad (4.25)$$

Fitness Function Based on AUC

When edge detection is considered as a binary classification problem, the class labels (edge points and non-edge points) are unbalanced (expect many more non-edge points than edge points in an image). The soft edge maps from GP detectors are evaluated by different threshold levels. The area under the curve (AUC) based on multi-threshold techniques has been used

for unbalanced binary classification performance evaluation [18]. Therefore, a new fitness function based on AUC (used in [18]) is employed for edge detection. The new fitness function $f_{F_{cr}}$ is shown in Equation (4.26), and it is based on the correlation ratio measure. Note that μ_0 and μ_1 are the means of the observations from non-edge points ($c = 0$) and edge points ($c = 1$), μ is the mean of all observations, $\sum_i$ contains the all observations, and $Th(\mu_0, \mu_1)$ (see Equation (4.27)) is used to indicate whether the means of two categories (edge points and non-edge points) are on different sides of threshold 0.

$$f_{F_{cr}} = \frac{1}{2} \sqrt{\frac{\sum_{c=0}^1 (\mu_c - \mu)^2}{\sum_i (o_i - \mu)^2}} + \frac{Th(\mu_0, \mu_1)}{2} \quad (4.26)$$

$$Th(\mu_0, \mu_1) = \begin{cases} 1 & \text{if } \mu_0 \leq 0 \text{ and } \mu_1 > 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.27)$$

4.5 Experiment Settings

This section describes the image datasets used in the experiments. Also, the parameter settings of the experiments are given.

4.5.1 Image Datasets

To evaluate the evolved edge detectors, two image datasets including natural images with ground truth provided are chosen. The first image dataset is the BSD dataset [123]. In order to check the generalisation ability of the GP edge detectors, the ETHZ shape classes [47] are only used to test the performances of the GP edge detectors evolved from the BSD dataset.

To sample training images as the training data, the small training dataset S_{20} (see Section 2.4 on page 46) is employed. Given the computational cost c_{edge} for evaluating a GP edge detector on an image, the probability for elitism $P_{elitism}$, the maximum generation N_{gen} and the population size N_{pop} , the computational cost in the training stage is approximately equal to $(1 - P_{elitism})N_{gen}N_{pop}c_{edge}N_{img}$. Generally, $P_{elitism}$, N_{gen} and N_{pop} are set by

Figure 4.6: Some example subimages from 20 training images in BSD and the corresponding ground truth.

a reasonable range, so the computational cost is mainly dependent on the number of images N_{img} . If the original training image dataset is directly used to evolve edge detectors, it is very expensive to evaluate a candidate edge detector. Therefore, in order to obtain an efficient training dataset for evolving GP edge detectors, N_{img} images from the BSD training dataset set are selected as training data. In [38], a subset of the 200 training images ($\frac{1}{6}$ of each of the first 100 images) was used to train edge detectors and the test performance is similar to the highest reported performance of the boundary detection in the literature. Therefore, a small subset of the training images (S_{20} in Section 3.4) is used for the GP experiments and five subimages are randomly sampled from each image with the condition of containing edge points. The training dataset contains N_{img} ($= 20 \times 5$) images of size 51×51 pixels. Figure 4.6 shows some subimages randomly sampled from the 20 training images.

4.5.2 Sets of Terminals and Functions

Section 3.4 (page 66) provides different settings for evolving low-level edge detectors. From the results of the experiments in Section 3.5, Set_s and $Set_{s,b}$ from Table 3.1 are employed in this chapter for the investigation of the fitness functions (from the pixel view and the image view). The search operators in Set_s are based on a single pixel, and the search operators in $Set_{s,b}$ are based on a single pixel or a single block of pixels. Set_s is also utilised to investigate the fitness functions based on localisation accuracy and observation spread.

4.5.3 Fitness Functions

The fitness functions used in this chapter are based on the pixel view and the image view, the matching direction in localisation, and the relationship between observations and the fixed threshold used in the GP system. These fitness functions are listed in Table 4.2. Here, dt_1 is the matching direction from predictions to ground truth, dt_2 is the matching direction from ground truth to predictions, and $f_{FOM,0.5}$ is $f_{FOM,\alpha}$ when α is set to 0.5. In “observation”, “free” indicates that the distance from an observation to the fixed threshold is not considered in the fitness function.

4.5.4 Parameter Settings

Figure 3.13 in Section 3.6.3 (on page 96) showed the difficulty of finding suitable combinations of the block size in $Set_{s,b}$. In order to decrease the computational cost in the training stage, the range of the block width w in $block_{t,l,w,d}$ is changed to from 1 to 3. The training images are 100 sub-images of 51×51 pixels, which is different from the 20 sub-images of 125×125 pixels in Section 3.4. The total pixels (260100) in the 100 sub-images is smaller than the total pixels (312500) in the 20 sub-images. A smaller population size 600 is used in this chapter, and the maximum

Table 4.2: Fitness functions used in this chapter.

criterion		method	functions
view	pixel	F -measure	$F, f_{F_{qf}}$
		FOM	f_{FOM}
	image	F -measure	$f_{F_{avg}}, f_{F_{gm}}, f_{F_{qm}}$
		FOM	$f_{FOM_{avg}}, f_{FOM_{gm}}, f_{FOM_{qm}}$
localisation	dt_1	FOM	f_{FOM}
	dt_2		$f_{FOM,gt}$
	$dt_1 + dt_2$		$f_{FOM,0.5}$
observation	free	F -measure	F
	short distance		f_{F_d}
	distance spread		$f_{F_s}, f_{F_{cr}}$

generation is increased to 250. Also, in order to find potentially good edge detectors within the 250 generations, the probability for mutation is increased to 0.35, and the probability for crossover is decreased to 0.60. These values are chosen based on common settings and empirical search via initial experiments.

4.6 Results and Discussions

This section firstly describes the results on evaluations from the pixel view and the image view. Secondly, the comparison between F -measure and FOM is provided. Thirdly, the test performance on the ETHZ image dataset is used to compare evolved edge detectors with existing edge detectors. Fourthly, the detected images from F -measure and FOM are visually described. Fifthly, the results from the fitness functions on matching directions in FOM are presented. Sixthly, the results from the fitness functions on observation spread are given.

Table 4.3: Means and standard deviations of test performance F , $f_{F_{avg}}$ and $f_{F_{gm}}$ values of GP edge detectors constructed by Set_s using F variants for the BSD 100 test images.

fitness function	test performance		
	F	$f_{F_{avg}}$	$f_{F_{gm}}$
F	0.2575 ± 0.0089	0.2731 ± 0.0100	0.2569 ± 0.0093
$f_{F_{avg}}$	0.2454 ± 0.0099	0.2631 ± 0.0099	0.2456 ± 0.0095
$f_{F_{gm}}$	0.2382 ± 0.0138	0.2558 ± 0.0145	0.2388 ± 0.0140
$f_{F_{qf}}$	0.2555 ± 0.0104	0.2685 ± 0.0113	0.2525 ± 0.0108
$f_{F_{qm}}$	0.2464 ± 0.0091	0.2643 ± 0.0100	0.2466 ± 0.0096

4.6.1 Evaluations on Pixel vs Image Views

Set_s Based on F -measure

Table 4.3 gives the means and standard deviations of test performance F , $f_{F_{avg}}$ and $f_{F_{gm}}$ values of GP edge detectors from the fitness functions F , $f_{F_{avg}}$, $f_{F_{gm}}$, $f_{F_{qf}}$ and $f_{F_{qm}}$ with Set_s for the BSD test image dataset. Since the fitness functions are from the pixel view and the image view, the test performance employs several fitness functions from both views as the performance evaluation methods. The test performance evaluation methods have F from the pixel view, and $f_{F_{avg}}$ and $f_{F_{gm}}$ from the image view. Comparing the means of test performance F , $f_{F_{avg}}$ and $f_{F_{gm}}$ values of the results evolved from the five fitness functions, the difference among these means is not remarkable. The best results come from the fitness function F in terms of the three test performance evaluation approaches F , $f_{F_{avg}}$ and $f_{F_{gm}}$.

In order to compare the differences in these results, a multiple comparison based on t -tests uses Holm's method [75] for p -value adjustment, and overall significance level 0.05. The comparison results are shown in

Table 4.4: Comparisons among GP edge detectors constructed by Set_s using F , $f_{F_{avg}}$, $f_{F_{gm}}$, $f_{F_{qf}}$ and $f_{F_{qm}}$ for the BSD 100 test images.

pair	test performance		
	F	$f_{F_{avg}}$	$f_{F_{gm}}$
$(F, f_{F_{avg}})$	↑	↑	↑
$(F, f_{F_{gm}})$	↑	↑	↑
$(F, f_{F_{qf}})$	—	—	—
$(F, f_{F_{qm}})$	↑	↑	↑
$(f_{F_{avg}}, f_{F_{gm}})$	—	—	—
$(f_{F_{avg}}, f_{F_{qf}})$	↓	—	—
$(f_{F_{avg}}, f_{F_{qm}})$	—	—	—
$(f_{F_{gm}}, f_{F_{qf}})$	↓	↓	↓
$(f_{F_{gm}}, f_{F_{qm}})$	↓	—	—
$(f_{F_{qf}}, f_{F_{qm}})$	↑	—	—

Table 4.4. Here, “↑” indicates that the results from the first fitness function are significantly better than the results from the second fitness function, “↓” indicates that the first one is significantly worse than the second one, and “—” indicates there is no significant difference between the two fitness functions.

From the pairwise comparisons, there are several interesting observations. Firstly, the fitness functions from the pixel view (F and $f_{F_{qf}}$) are significantly better than the fitness functions from the image view when the test performance evaluation method is based on overall pixels. Secondly, the results from fitness function F are significantly better than the results from all other fitness functions from the image view no matter whether the test performance evaluation method uses F , $f_{F_{avg}}$ or $f_{F_{gm}}$. Thirdly, using the fitness function with square ($f_{F_{qf}}$ or $f_{F_{qm}}$) does not significantly affect the results, e.g., F with $f_{F_{qf}}$ (P_r^2 and P_p^2), and $f_{F_{avg}}$ with $f_{F_{qm}}$ (f_i^2). Fourthly,

Figure 4.7: Scatter plot for detection results $f_{i,0.5}$ (mean of 30 GP edge detector in each fitness function for each image) based on F with Set_s .

$f_{F_{gm}}$ has no significantly better results than the others.

Figure 4.7 (a) shows the correlation coefficients between F (the horizontal axis) and the other fitness functions (the vertical axis) in Table 4.3 based on the average of f_i values (30 GP edge detectors) of each image in the BSD test dataset. Here, the horizontal axis indicates results from the fitness function F , the numbers in the legend are the correlation coefficients, and “slope= 1.0” indicates a line with slope 1.0. To clearly show the relationship between the detection results from F and the other fitness functions based on F -measure, Figures 4.7 (b)–(e) give the separate distributions for the relevant detection results. From these figures, the detection results from $f_{F_{qf}}$ are very close to the results from F . Some results from $f_{F_{qm}}$ and $f_{F_{avg}}$ are not so close to the results from F . One image has better results by the edge detectors from $f_{F_{gm}}$ than the edge detectors from F , but for most of the 100 images, the results from $f_{F_{gm}}$ are worse than the results

Figure 4.8: Scatter plot for detection results $f_{i,0.5}$ (all 30 GP edge detectors) based on F with Set_s .

from F , in terms of test performance f_i .

Figure 4.8 (a) shows the correlation coefficients for the BSD test images detected by all edge detectors from F and the other fitness functions based on F . Figures 4.8 (b)–(e) present the separate distributions between F and $f_{F_{qf}}$, F and $f_{F_{qm}}$, F and $f_{F_{avg}}$, and F and $f_{F_{gm}}$. Similar to the relevant distributions based on average performance (see Figure 4.8), some of the detection results from $f_{F_{gm}}$ are obviously different from the results from F .

Figure 4.9 (a) presents the test detection performance (f_{FOM_i}) distributions between the fitness function F and the other fitness functions (based on F -measure). Figures 4.9 (b)–(e) give the separate distributions for these fitness functions. Based on the evaluation using f_{FOM_i} on each test image, it is easy to find that the detection performances on a few individual images from $f_{F_{gm}}$ are better than the detection results from F , although $f_{F_{gm}}$

Figure 4.9: Scatter plot for detection results f_{FOM_i} (all 30 GP edge detectors) based on F with Set_s .

is worse than F (the correlation coefficient 0.9369) from the image view of all 100 test images.

$Set_{s,b}$ Based on F -measure

Table 4.5 gives the means and standard deviations of test performance F , $f_{F_{avg}}$ and $f_{F_{gm}}$ values of GP edge detectors from the fitness functions F , $f_{F_{avg}}$, $f_{F_{gm}}$, $f_{F_{qf}}$ and $f_{F_{qm}}$ with $Set_{s,b}$ for the BSD test image dataset. Similar to Table 4.3, the difference among these means of the three test performance methods from the five fitness functions are not large. The five fitness functions also keep the same rank by the three test performance evaluation approaches, compared with Table 4.3. It seems that the test performance evaluations from the pixel view and the image view are similar when the dataset is large. In addition, all the highest means of test performance F , $f_{F_{avg}}$ and $f_{F_{gm}}$ values come from the fitness functions $f_{F_{qf}}$,

Table 4.5: Means and standard deviations of test performance F , $f_{F_{avg}}$ and $f_{F_{gm}}$ values of GP edge detectors constructed by $Set_{s,b}$ using F variants for the BSD 100 test images.

fitness function	test performance		
	F	$f_{F_{avg}}$	$f_{F_{gm}}$
F	0.2870 ± 0.0140	0.2994 ± 0.0105	0.2832 ± 0.0108
$f_{F_{avg}}$	0.2693 ± 0.0202	0.2868 ± 0.0168	0.2691 ± 0.0173
$f_{F_{gm}}$	0.2677 ± 0.0166	0.2828 ± 0.0138	0.2660 ± 0.0140
$f_{F_{qf}}$	0.2895 ± 0.0143	0.3002 ± 0.0127	0.2843 ± 0.0126
$f_{F_{qm}}$	0.2667 ± 0.0133	0.2863 ± 0.0112	0.2681 ± 0.0116

which is different from the highest means from the fitness function F in Table 4.3.

Table 4.6 gives the multiple comparison results for the five fitness functions. Different from the comparison results in Table 4.4, there are no different comparison results among the three test performance methods for each pair in Table 4.6. The fitness functions on overall pixels are significantly better than the fitness functions on individual images, and there are no significant differences among the fitness functions $f_{F_{avg}}$, $f_{F_{gm}}$ and $f_{F_{qm}}$.

Figure 4.10 gives the distributions of the average (30 replications in each fitness function) of test performance $f_{i,0.5}$ between F and $f_{F_{qf}}$, F and $f_{F_{qm}}$, F and $f_{F_{avg}}$, and F and $f_{F_{gm}}$. The detection results from $f_{F_{qm}}$, $f_{F_{avg}}$ and $f_{F_{gm}}$ have very close correlation coefficients with F . Only a few images have better detection results from the fitness functions based on individual images than the fitness functions based on overall pixels. The correlation coefficient $f_{F_{gm}}$ against F in $Set_{s,b}$ is increased, compared with its value in Set_s . When the search operator $block_{t,l,w,d}$ is used to improve testing performances, the difference of the detection result between F and $f_{F_{gm}}$ decreases.

Table 4.6: Comparisons among GP edge detectors constructed by $Set_{s,b}$ using F , $f_{F_{avg}}$, $f_{F_{gm}}$, $f_{F_{qf}}$ and $f_{F_{qm}}$ for the BSD 100 test images.

pair	test performance		
	F	$f_{F_{avg}}$	$f_{F_{gm}}$
(F, F_{avg})	↑	↑	↑
(F, F_{gm})	↑	↑	↑
(F, F_{qf})	—	—	—
(F, F_{qm})	↑	↑	↑
(F_{avg}, F_{gm})	—	—	—
(F_{avg}, F_{qf})	↓	↓	↓
(F_{avg}, F_{qm})	—	—	—
(F_{gm}, F_{qf})	↓	↓	↓
(F_{gm}, F_{qm})	—	—	—
(F_{qf}, F_{qm})	↑	↑	↑

Figure 4.10: Scatter plot for detection results $f_{i,0.5}$ (mean of 30 GP edge detectors) based on F with $Set_{s,b}$.

Set_s Based on FOM

Table 4.7 gives the means and standard deviations of test performance F , f_{FOM} and $f_{FOM_{avg}}$ values of the results from the fitness functions f_{FOM} ,

Table 4.7: Means and standard deviations of test performance F and f_{FOM} values of GP edge detectors constructed by Set_s using FOM variants for the BSD 100 test images.

fitness function	test performance		
	F	f_{FOM}	$f_{FOM_{avg}}$
f_{FOM}	0.2397 ± 0.0235	0.4447 ± 0.0181	0.4444 ± 0.0171
$f_{FOM_{avg}}$	0.2486 ± 0.0092	0.4200 ± 0.0093	0.4424 ± 0.0098
$f_{FOM_{gm}}$	0.2332 ± 0.0118	0.4021 ± 0.0158	0.4208 ± 0.0130
$f_{FOM_{qm}}$	0.2465 ± 0.0098	0.4196 ± 0.0086	0.4422 ± 0.0087

$f_{FOM_{avg}}$, $f_{FOM_{gm}}$ and $f_{FOM_{qm}}$. Since $f_{F_{gm}}$ and $f_{F_{avg}}$ have the same indications for the performance evaluations in Table 4.3 and Table 4.5, $f_{FOM_{gm}}$ is not used for the test performance evaluation. To present the detection accuracy, F is added in here. Similar to Table 4.3, the fitness function f_{FOM} is the best for evolving GP edge detectors in the four fitness functions, in terms of f_{FOM} and $f_{FOM_{avg}}$. The means of $f_{FOM_{avg}}$ values from the four fitness functions are closer than the means of f_{FOM} values on the BSD test images. Also, the test performance approaches for the fitness functions f_{FOM} , $f_{FOM_{avg}}$, $f_{FOM_{gm}}$ and $f_{FOM_{qm}}$ are the same rank in the evaluations using f_{FOM} and $f_{FOM_{avg}}$. However, the ranks by the test performance evaluation approaches f_{FOM} and $f_{FOM_{avg}}$ are different from the rank by the evaluation approach F . The highest f_{FOM} is the results from the fitness function f_{FOM} itself, but the maximum of the average of F values comes from the fitness function $f_{FOM_{avg}}$.

Table 4.8 gives the multiple comparison results for the four fitness functions. From the test performance evaluation approach f_{FOM} , the fitness function f_{FOM} is significantly better than the other three fitness functions. However, if $f_{FOM_{avg}}$ is used to evaluate edge detectors' performance, the standard FOM as the fitness function is only significantly better than the

Table 4.8: Comparisons among GP edge detectors constructed by Set_s using f_{FOM} , $f_{FOM_{avg}}$, $f_{FOM_{gm}}$ and $f_{FOM_{qm}}$ for the BSD 100 test images.

pair	test performance		
	F	f_{FOM}	$f_{FOM_{avg}}$
$(f_{FOM}, f_{FOM_{avg}})$	—	↑	—
$(f_{FOM}, f_{FOM_{gm}})$	—	↑	↑
$(f_{FOM}, f_{FOM_{qm}})$	—	↑	—
$(f_{FOM_{avg}}, f_{FOM_{gm}})$	↑	↑	↑
$(f_{FOM_{avg}}, f_{FOM_{qm}})$	—	—	—
$(f_{FOM_{gm}}, f_{FOM_{qm}})$	↓	↓	↓

fitness function $f_{FOM_{gm}}$. This is different from the comparison results when F -measure is used to evaluate the performance of edge detectors evolved by the fitness functions based on F -measure from the pixel view and the image view. Also, when the fitness functions based on FOM are used from the pixel view and the image view and the test performance is based on F , the standard FOM as the fitness function has no significantly different results, compared with the fitness functions based on individual images.

Figure 4.11 gives the distributions based on f_{FOM_i} for all test images detected by the fitness functions $f_{FOM_{qm}}$, $f_{FOM_{avg}}$ and $f_{FOM_{gm}}$, against the results from the fitness function f_{FOM} . The correlation coefficients for these fitness functions are lower than 0.9, however, the correlation coefficients from the relevant fitness functions based on F in Figure 4.7 and Figure 4.9 are higher than 0.9. It seems that the fitness functions using FOM more easily affect the detection results based on the pixel view and the image view than F -measure.

Figure 4.11: Scatter plot for detection results f_{FOM_i} (mean of 30 replications) based on f_{FOM} with Set_s .

$Set_{s,b}$ Based on FOM

Table 4.9 gives the means and standard deviations of test performance F , f_{FOM} and $f_{FOM_{avg}}$ values of the results from the fitness functions f_{FOM} , $f_{FOM_{avg}}$, $f_{FOM_{gm}}$ and $f_{FOM_{qm}}$. Different from Table 4.7, the rank (in terms of f_{FOM}) for the four fitness functions is different with the evaluation using $f_{FOM_{avg}}$. The best test performance f_{FOM} is from the standard FOM as the fitness function, but the maximum of the average of $f_{FOM_{avg}}$ of the test results comes from the fitness function $f_{FOM_{avg}}$. Similar to Table 4.7, the maximum of the average of test performance F values of the test results is from the fitness function $f_{FOM_{avg}}$.

Table 4.10 gives the multiple comparison results for the four fitness functions. The only similarity of these comparisons to the former three multiple comparisons (in Table 4.4, Table 4.6 and Table 4.8) is that the fitness function based on overall pixels has the significantly best performance when the fitness function itself is used as the test performance evaluation approach. The interesting observation in Table 4.10 is that there are no significant differences for most of all pairs. When F or $f_{FOM_{avg}}$ is used as the test performance evaluation method, there are no significant differences among the four fitness functions.

Table 4.9: Means and standard deviations of F and f_{FOM} values of GP edge detectors constructed by $Set_{s,b}$ using FOM variants for the BSD 100 test images.

fitness function	test performance		
	F	f_{FOM}	$f_{FOM_{avg}}$
f_{FOM}	0.2730 ± 0.0196	0.4949 ± 0.0314	0.4717 ± 0.0186
$f_{FOM_{avg}}$	0.2842 ± 0.0150	0.4649 ± 0.0238	0.4753 ± 0.0176
$f_{FOM_{gm}}$	0.2829 ± 0.0111	0.4667 ± 0.0198	0.4717 ± 0.0148
$f_{FOM_{qm}}$	0.2726 ± 0.0232	0.4475 ± 0.0327	0.4653 ± 0.0235

Table 4.10: Comparisons among GP edge detectors constructed by $Set_{s,b}$ using f_{FOM} , $f_{FOM_{avg}}$, $f_{FOM_{gm}}$ and $f_{FOM_{qm}}$ for the BSD 100 test images.

pair	test performance		
	F	f_{FOM}	$f_{FOM_{avg}}$
$(f_{FOM}, f_{FOM_{avg}})$	—	↑	—
$(f_{FOM}, f_{FOM_{gm}})$	—	↑	—
$(f_{FOM}, f_{FOM_{qm}})$	—	↑	—
$(f_{FOM_{avg}}, f_{FOM_{gm}})$	—	—	—
$(f_{FOM_{avg}}, f_{FOM_{qm}})$	—	—	—
$(f_{FOM_{gm}}, f_{FOM_{qm}})$	—	—	—

Figure 4.12 shows the distributions based on f_{FOM_i} for all test images detected by the fitness functions $f_{FOM_{qm}}$, $f_{FOM_{avg}}$, and $f_{FOM_{gm}}$ with $Set_{s,b}$, against the results from the fitness function f_{FOM} . The correlation coefficients for fitness functions $f_{FOM_{qm}}$ and $f_{FOM_{avg}}$ with $Set_{s,b}$ decrease a bit, compared with the relevant correlation coefficients for both fitness functions with Set_s . It seems that the difference of the detection results between f_{FOM} and its variants becomes larger when the search operator

Figure 4.12: Scatter plot for detection results f_{FOM_i} (mean of 30 replications) based on f_{FOM} with $Set_{s,b}$.

$block_{t,l,w,d}$ is used to improve detection performances (using f_{FOM} as the measurement).

Summary

There are several interesting phenomena between the evaluations on overall pixels and individual full images. Firstly, the fitness function from the pixel view has the best test performance when the test performance evaluation is based on the pixel view. Secondly, the test performance for the fitness functions using F -measure does not show an obvious difference between the evaluations from the pixel view and the image view. Thirdly, the test performance using f_{FOM} is affected by the view (pixel or image). The results indicate that F -measure as a fitness function is better for the evaluation from the pixel view than from the image view. The relative influence of FOM will be discussed in the next subsections.

4.6.2 Fitness Functions F vs FOM

To compare all GP edge detectors from different fitness functions with the Sobel edge detector, the maximum values of F and f_{FOM} of the results

Table 4.11: Comparisons between GP edge detectors and the Sobel edge detector, for the BSD 100 test images.

Pair	Set_s (test performance)				$Set_{s,b}$ (test performance)			
	F	$f_{F_{avg}}$	f_{FOM}	$f_{FOM_{avg}}$	F	$f_{F_{avg}}$	f_{FOM}	$f_{FOM_{avg}}$
(F, Sobel)	↑	↑	↓	—	↑	↑	—	↑
$(f_{F_{avg}}, \text{Sobel})$	↑	↑	↓	—	↑	↑	↓	↑
$(f_{F_{gm}}, \text{Sobel})$	—	—	↓	↓	↑	↑	↓	—
$(f_{F_{qm}}, \text{Sobel})$	↑	↑	↓	—	↑	↑	↓	—
$(f_{F_{qf}}, \text{Sobel})$	↑	↑	↓	—	↑	↑	—	↑
(f_{FOM}, Sobel)	—	—	↓	—	↑	↑	↑	↑
$(f_{FOM_{avg}}, \text{Sobel})$	↑	↑	↓	—	↑	↑	—	↑
$(f_{FOM_{gm}}, \text{Sobel})$	—	—	↓	↓	↑	↑	—	↑
$(f_{FOM_{qm}}, \text{Sobel})$	↑	↑	↓	—	↑	↑	—	↑

from the Sobel edge detector are obtained by 50 thresholds for the 100 BSD test images. Here, the maximum F of the Sobel edge detector is 0.2337, its maximum $f_{F_{avg}}$ (using a fixed threshold for all test images) is 0.2478, its maximum f_{FOM} is 0.4642 and its maximum $f_{FOM_{avg}}$ is 0.4380. Table 4.11 gives the comparison results from Set_s and $Set_{s,b}$ based on the test performance F , $f_{F_{avg}}$, f_{FOM} and $f_{FOM_{avg}}$.

From the test performance F , almost all GP evolved edge detectors are significantly better than the Sobel edge detector, except for the edge detectors from the fitness functions $f_{F_{gm}}$, f_{FOM} and $f_{FOM_{gm}}$, using Set_s . It seems that the geometric mean is not a good fitness function for evaluating trained edge detectors when the training data is large. In addition, all GP evolved edge detectors from $Set_{s,b}$ are significantly better than the Sobel edge detector, in terms of the test performance F and $f_{F_{avg}}$. It also shows that $block_{t,l,w,d}$ can efficiently construct edge detectors.

Based on the test performance f_{FOM} , the Sobel edge detector outper-

forms all of the edge detectors from Set_s and some of the edge detectors from $Set_{s,b}$. Only the edge detectors from the fitness function f_{FOM} with $Set_{s,b}$ are significantly better than the Sobel edge detector. It seems that the GP edge detectors do not perform well from the test performance f_{FOM} when the fitness function is based on F -measure.

From the test performance $f_{F_{avg}}$ and $f_{FOM_{avg}}$, the comparison results in Table 4.11 reveal that $f_{F_{avg}}$ is similar to F , but $f_{FOM_{avg}}$ is obviously different from f_{FOM} . The comparisons results for Set_s based on test performance $f_{FOM_{avg}}$ show that all GP evolved edge detectors from these fitness functions are not significantly different from the Sobel edge detector, except for the edge detectors from the fitness functions $f_{F_{gm}}$ and $f_{FOM_{gm}}$. The geometric mean as the fitness function has the worst detection results than the Sobel edge detector, in terms of the test performance f_{FOM} . All the results from these fitness functions with $Set_{s,b}$ are significantly better than the Sobel edge detector in terms of the test performance $f_{FOM_{avg}}$, except for the results from the fitness functions $f_{F_{gm}}$ and $f_{F_{qm}}$. It seems that the evaluation approaches using FOM from the pixel view and the image view are obviously different, but F -measure is not strongly affected by the view from overall pixels or individual images.

Further, to investigate whether f_{FOM} is fair to evaluate the test performance, one-to-one assignment is employed to check the test performance. In order to fairly compare F with f_{FOM} , the results from the full set of the terminals and functions $Set_{s,b}$ are evaluated by F after one-to-one assignment. Table 4.12 gives the means and standard deviations of F values of the GP edge detectors from the fitness functions F and f_{FOM} . The maximum F based on the BSD test image dataset for the Sobel, Marr [122] and Canny [28] edge detectors are 0.48, 0.50 and 0.58, respectively [4].

Table 4.13 gives the multiple comparisons for these edge detectors. From the comparison results, the GP evolved edge detectors from the fitness function F are significantly better than the edge detectors from the fitness function f_{FOM} . Even though the GP edge detectors from the fitness

Table 4.12: Test performance F after using one-to-one assignment on the GP edge detectors constructed by $Set_{s,b}$ with fitness function F or f_{FOM} for the BSD 100 test images.

fitness function	test performance F	
	mean	standard deviation
F	0.5077	0.0156
f_{FOM}	0.4882	0.0366

Table 4.13: Comparisons among GP edge detectors constructed by $Set_{s,b}$, the Sobel edge detector, the Marr edge detector [122] and the Canny edge detector for BSD 100 test images after one-to-one assignment.

pair	test performance F
(F, f_{FOM})	↑
(F, Sobel)	↑
(F, Marr)	—
(F, Canny)	↓
(f_{FOM}, Sobel)	—
(f_{FOM}, Marr)	—
(f_{FOM}, Canny)	↓

function F do not contain pre-processing and post-processing techniques, they are significantly better than the Sobel and Marr edge detectors. When f_{FOM} is used, the evolved edge detectors are not significantly different from the Sobel and Marr edge detectors. However, these GP edge detectors are significantly worse than the Canny detector based on the evaluation approach F after using one-to-one assignment. A potential reason is that the Canny edge detector includes filtering noise, non-maximum suppression and hysteresis thresholding techniques.

Table 4.14: Test performance (F) for detection results on the ETHZ dataset.

class	Canny	Berkeley	CT	GP (F)	GP(f_{FOM})
apple logos	0.05	0.15	0.21	0.20 ± 0.02	0.16 ± 0.03
bottles	0.11	0.28	0.29	0.19 ± 0.02	0.17 ± 0.03
giraffes	0.10	0.32	0.26	0.17 ± 0.01	0.16 ± 0.02
mugs	0.15	0.32	0.30	0.21 ± 0.01	0.18 ± 0.02
swans	0.10	0.27	0.38	0.17 ± 0.01	0.16 ± 0.02
Average	0.10	0.27	0.29	0.19	0.17

From the comparison result between F and f_{FOM} , it is found that F is significantly better than f_{FOM} to evolve edge detectors for the BSD image dataset.

4.6.3 Testing on the ETHZ Dataset

To check the generalisation ability of these GP edge detectors, the ETHZ dataset is used as another set of test images. Table 4.14 gives the test performances for the Canny and GP edge detectors evolved by the fitness functions F and f_{FOM} with $Set_{s,br}$ and the boundary detectors Berkeley [123] and Component Tree (CT) [39]. Except for the results of the GP edge detectors, all test results come from [39] (only giving F). Here, “Average” indicates the average of F values for the five classes (apple logos, etc.). From the average results, these GP edge detectors from F and f_{FOM} have higher means than the Canny edge detector, but these means are lower than the two boundary detectors.

Table 4.15 gives the comparison results based on different classes in the image dataset, where “ F ” indicates the edge detectors come from F , and “ f_{FOM} ” indicates the edge detectors come from f_{FOM} . Firstly, the GP edge detectors from F and f_{FOM} are significantly better than the Canny edge detector based on the test performance evaluation approach F . Secondly,

Table 4.15: Comparisons among GP edge detectors constructed by $Set_{s,b}$ from fitness functions F and f_{FOM} , the Canny edge detector, the Berkeley and Component Tree (CT) boundary detectors for the ETHZ image dataset.

pair	apple logos	bottles	giraffes	mugs	swans
(F, f_{FOM})	↑	—	—	↑	↑
(F, Canny)	↑	↑	↑	↑	↑
$(F, \text{Berkeley})$	↑	↓	↓	↓	↓
(F, CT)	↓	↓	↓	↓	↓
(f_{FOM}, Canny)	↑	↑	↑	↑	↑
$(f_{FOM}, \text{Berkeley})$	—	↓	↓	↓	↓
(f_{FOM}, CT)	↓	↓	↓	↓	↓

F has three subimage datasets with significantly better detection results than f_{FOM} , and the other two have no significant difference. Therefore, the edge detectors from the fitness function F are better than the edge detectors from f_{FOM} based on the test performance evaluation approach F . Thirdly, although the GP edge detectors are low-level edge detectors, they provide one good result (the apple logos) which is significantly better than the Berkeley boundary detector. However, these GP edge detectors are significantly worse than the CT boundary detector.

4.6.4 Example Detected Images From F and f_{FOM}

Figure 4.13 shows three example detected BSD images by the best edge detector from the fitness functions F (test performance $F = 0.3131$) and f_{FOM} (test performance $F = 0.3078$) with $Set_{s,b}$. Most of the edges in the three images are detected, so the GP edge detectors have very good ability to find true edge points. Compared to the best edge detector from f_{FOM} , the best edge detector from F finds more true edge points, e.g., the left

Figure 4.13: Three example BSD images detected by the best edge detectors from F and f_{FOM} with $Set_{s,b}$.

part of the boat in the image from the first column, the edges from the reflection in the water in the image from the middle column, and the middle line at the right bottom of the image from the last column. Note that some true edges detected by the edge detector from F , but missed by the edge detector from f_{FOM} , have low graylevels because these edges are only considered by a few human observers. Although the edge detector from F has good ability to detect edges, it is more affected by some textures than the edge detector from f_{FOM} , e.g., the textures from the left bottom of the first

Figure 4.14: Three example ETHZ images detected by the best edge detectors from F and f_{FOM} with $Set_{s,b}$.

image, the middle of the second image, and the grass in the third image.

Figure 4.14 gives three example detected images from the ETHZ dataset by the two best edge detectors from F and f_{FOM} . It shows the same characteristics of detection results from the GP edge detectors in the three images, e.g., the edges in the water in image black3. The left of the road in image road is found by the edge detector from F , but it is poorly detected by the edge detector from f_{FOM} . Also, noise in the three images more heavily affect the edge detector from F than the edge detector from f_{FOM} .

Therefore, it seems that the edge detector from F has good ability to find true edge points, but it is affected by some textures, and the edge detector from f_{FOM} has a good ability to suppress textures but misses some true edge points.

To summarise, F is better than f_{FOM} as the fitness function for evolving edge detectors based on the BSD test image dataset and the ETHZ image dataset. When the tolerance distance from predictions to ground truth is not allowed, the results on the ETHZ image dataset show that GP edge detectors are better than the Canny edge detector. From the comparisons between the edge detectors evolved from f_{FOM} and the Sobel edge detector, and between the edge detectors evolved from $f_{FOM_{avg}}$ and the Sobel edge detector, $f_{FOM_{avg}}$ is better than f_{FOM} to evolve edge detectors, in terms of the test performance F and $f_{FOM_{avg}}$.

4.6.5 Results on Matching Directions in FOM

Overall Results

Table 4.16 gives the test performance of the evolved edge detectors from $f_{FOM,gt}$ and $f_{FOM,0.5}$. Here, paired-sample t -tests with significance level 0.05 are used to compare the results from $f_{FOM,gt}$ or $f_{FOM,0.5}$ with the results from the standard FOM. To evaluate the test performance of these evolved edge detectors, the test approaches choose the standard FOM and the average of FOM. Since only one matching direction is used in the evaluation functions, the detection accuracy is also evaluated by F .

In terms of the three test evaluation functions F , f_{FOM} and $f_{FOM_{avg}}$, the fitness function $f_{FOM,gt}$ has significantly worse performance than the standard FOM. However, the proposed fitness function $f_{FOM,0.5}$ is significantly better than the standard FOM, in terms of test performance F . When the test performance uses f_{FOM} or $f_{FOM_{avg}}$, there is no significant difference between $f_{FOM,gt}$ and the standard FOM. If comparing $f_{FOM,0.5}$ with F , there is no significant difference, in terms of test performance F . How-

Table 4.16: Test performance on different matching directions (compared with standard FOM).

fitness function	test performance		
	F	f_{FOM}	$f_{FOM_{avg}}$
$f_{FOM,gt}$	$0.1985 \pm 0.0168 \downarrow$	$0.3882 \pm 0.0146 \downarrow$	$0.3985 \pm 0.1287 \downarrow$
$f_{FOM,0.5}$	$0.2547 \pm 0.0136 \uparrow$	0.4455 ± 0.0196	0.4517 ± 0.0124

ever, when f_{FOM} or $f_{FOM_{avg}}$ is used as the test approach, the detected results from $f_{FOM,0.5}$ are significantly better than the detected results from F when a paired t -test with significance level 0.05 is used. It seems that the combination of F and FOM improves the detection performance.

Note that $f_{FOM,gt}$ uses the matching direction from ground truth to predictions. Although f_{FOM} and $f_{FOM_{avg}}$ use the matching direction from predictions to ground truth, F balances recall and precision on detection accuracy. From the three test approaches, it seems that $f_{FOM,gt}$ is significantly worse than the standard FOM for evolving edge detectors in this experiment. A potential reason for $f_{FOM,gt}$ having bad test performance is that the $f_{FOM,gt}$ finds it hard to select potentially good edge detectors in the early training stage, as described in Section 4.3.3 (on page 111).

Detected Images

Figure 4.15 shows three example images detected by the best edge detector from $f_{FOM,gt}$ and $f_{FOM,0.5}$ respectively. The detected images from $f_{FOM,gt}$ have thinner edges than the results of the other edge detectors from the fitness functions using the standard F -measure or FOM. However, these results are strongly affected by noise. The detected results indicate that $f_{FOM,gt}$ restricts the matching from ground truth to predictions (only allowing one ground truth edge point to match one predicted edge point) to

Figure 4.15: Three example BSD test images detected by the best edge detector from $f_{FOM,gt}$ and $f_{FOM,0.5}$ with Set_s , respectively.

obtain thin edges, but it is affected by noise because there is no matching direction from predictions to ground truth.

The detected images from $f_{FOM,0.5}$ are also affected by noise, but they have weaker influence from noise than the results from $f_{FOM,gt}$. Figure 4.16 shows the three images detected by the best edge detector from F and f_{FOM} respectively. Comparing the visual results in Figure 4.16, some detected edges from $f_{FOM,0.5}$ (in Figure 4.15) are thinner than the relevant detected edges from F and f_{FOM} , such as the boundaries at the middle

Figure 4.16: Three example BSD test images detected by the best edge detector from F and f_{FOM} with Set_s , respectively.

of the person in image 101087. Similar to the evolved edge detector from F , the textures strongly affect the edge detectors from f_{FOM} and $f_{FOM,0.5}$ to detect image 145086. The influence for the best edge detector from $f_{FOM,0.5}$ is weaker than the edge detectors from F and f_{FOM} . In image 78004, the best edge detector from F misses some edges on the boat, but the edge detector from $f_{FOM,0.5}$ finds them. It seems that $f_{FOM,0.5}$ can be used to evolve edge detectors for finding edge details (the matching direction from ground truth to predictions) and reduce influence of noise and textures (the matching direction from predictions to ground truth), and it is better than F and f_{FOM} to evolve edge detectors.

4.6.6 Results on Observation Spread

The evolved edge detectors from the fitness functions based on observation spread are investigated. Test performance using the fixed threshold 0, the linear transformation, the S-shaped transformation and a bandpass transformation are presented and discussed.

Table 4.17: Test performance (F and $f_{F_{avg}}$) on fixed threshold 0 for fitness functions $f_{F_{cr}}$, f_{F_d} and f_{F_s} .

	F	p -value	$f_{F_{avg}}$	p -value
$f_{F_{cr}}$	0.2151 ± 0.0361	0.0000 ↓	0.2230 ± 0.0401	0.0000 ↓
f_{F_d}	0.2502 ± 0.0174	0.0601	0.2649 ± 0.0200	0.0611
f_{F_s}	0.2541 ± 0.0163	0.3759	0.2684 ± 0.0178	0.2594

Fixed Threshold 0

Table 4.17 provides the means and standard deviations of test performance F and $f_{F_{avg}}$ for the evolved edge detectors from $f_{F_{cr}}$, f_{F_d} and f_{F_s} . Since the initial population are the same for the four fitness functions (F , $f_{F_{cr}}$, f_{F_d} and f_{F_s}), the differences of the pairs with the same initialisation are employed. The p -values are from the paired-sample t -tests (comparing to the test performance of the evolved edge detectors from F , see Table 4.3).

Although the correlation ratio indicates dispersion for binary targets [18], the detection accuracy for the evolved edge detectors with threshold 0 is significantly worse than the fitness function F . However, the two modified fitness functions f_{F_d} and f_{F_s} do not significantly reduce the test performance F and $f_{F_{avg}}$. A reason is that $f_{F_{cr}}$ mainly focuses on the dispersion of observations between edge points and non-edge points and the dispersion of all observations. Fitness functions f_{F_d} and f_{F_s} require not only the observations' position, but also the detection accuracy F . From the p -values 0.0601 and 0.3759, it seems that the restriction of the range of observations of evolved edge detectors has stronger influence than the S-shaped transformation for evolving edge detectors.

Figure 4.17 presents boxplots for the evolved edge detectors (with threshold 0) from F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} , in terms of test performance F and $f_{F_{avg}}$ respectively. From the localisation of F and $f_{F_{avg}}$ values for these evolved edge detectors, F , f_{F_d} and f_{F_s} are more stable than $f_{F_{cr}}$ to evolve edge de-

Figure 4.17: Boxplots for test performance F and $f_{F_{avg}}$ of the evolved edge detectors (with threshold 0) from fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} .

tectors with threshold 0 to obtain binary edge maps. The performance of the evolved edge detectors from $f_{F_{cr}}$ is not stable, and some have bad performance (low values). A reason is that $f_{F_{cr}}$ does not mainly work on the accuracy when threshold 0 is used. Since fitness functions F , f_{F_d} and f_{F_s} employ threshold 0, their test performance of using threshold 0 is stable.

The best edge detector (in terms of test performance F) comes from f_{F_d} , but it is considered as an outlier in the 30 GP edge detectors. In terms of test performance $f_{F_{avg}}$, the best edge detector also comes from f_{F_d} . f_{F_d} and f_{F_s} have one result with very low test performance F and $f_{F_{avg}}$, which is considered as an outlier in the 30 GP edge detectors. It is possible that the restriction on observations might increase the difficulty of finding good edge detectors.

Since the test performance of the evolved edge detectors from $f_{F_{cr}}$ is not normally distributed, a non-parametric statistic technique, the Kruskal-Wallis method [174], is employed for the comparisons among these fitness functions, in terms of the number of the results in the first group being higher than the second group. The p -value adjustment uses Holm's method [75] and the overall significance level is 0.05. Table 4.18 gives the statistical test results. Here, “—” indicates there is no significant difference.

Table 4.18: Kruskal-Wallis statistic [174] of pairs (test performance F and $f_{F_{avg}}$) on the results of the GP edge detectors with threshold 0.

	$f_{F_{cr}}$	f_{F_d}	f_{F_s}
F	($\downarrow, \downarrow$)	($-, -$)	($-, -$)
$f_{F_{cr}}$		($\uparrow, \uparrow$)	($\uparrow, \uparrow$)
f_{F_d}			($-, -$)

In a pair (A, B) , A indicates the evaluation using F , and B indicates the evaluation using $f_{F_{avg}}$. The comparisons show that the results from $f_{F_{cr}}$ are significantly worse than the results from F , f_{F_d} and f_{F_s} . There are no significant differences among the results from F , f_{F_d} and f_{F_s} .

In summary, from the comparisons in Figure 4.17 and Table 4.18, restrictions on the observations of evolved edge detectors using threshold 0 may not affect the performance on the detected binary edges. The correlation ratio is not good to evolve edge detectors with threshold 0 for obtaining binary edge maps on the BSD test image dataset.

Soft Edge Maps from Linear Transformation

With the purpose of evaluating the soft edge maps obtained from the evolved edge detectors, different thresholds are used to obtain binary edge maps and the performance for these evolved edge detectors are based on the best binary edge maps. From the overall view, threshold c_{th} is employed to discriminate pixels on the soft edge maps as edge points or non-edge points. After using threshold c_{th} on the soft edge maps of an edge detector, the relevant binary edge maps are obtained. The performance of the edge detector with threshold c_{th} is $F_{c_{th}}$ (see Equation (4.28)). Here, $P_{rec, c_{th}}$ and $P_{pre, c_{th}}$ are the relevant recall and precision when using threshold c_{th} . F_{max} (see Equation (4.29)) is considered as the test performance evaluation method from the pixel view. $F_{max, avg}$ is considered as the test

Table 4.19: Test performance $F_{max,avg}$ and F_{max} of the GP evolved edge detectors after using LT .

	F_{max}	p -value	$F_{max,avg}$	p -value
F	0.1867 ± 0.0619	0.0000 ↓	0.1917 ± 0.0730	0.0000 ↓
$f_{F_{cr}}$	0.1830 ± 0.0392	0.0000 ↓	0.1864 ± 0.0435	0.0000 ↓
f_{F_d}	0.1992 ± 0.0440	0.0000 ↓	0.2101 ± 0.0473	0.0000 ↓
f_{F_s}	0.1481 ± 0.0552	0.0000 ↓	0.1520 ± 0.0664	0.0000 ↓

performance evaluation method from the image view. $F_{max,avg}$ is defined in Equation (4.30), where $f_{F_{c_{th},avg}}$ is $f_{F_{avg}}$ when threshold c_{th} is used.

$$F_{c_{th}} = \frac{P_{rec,c_{th}} P_{pre,c_{th}}}{\alpha P_{rec,c_{th}} + (1 - \alpha) P_{pre,c_{th}}} \quad (4.28)$$

$$F_{max} = \max_{c_{th} = \frac{k}{51}, k=1,2,\dots,50} \{F_{c_{th}}\} \quad (4.29)$$

$$F_{max,avg} = \max_{c_{th} = \frac{k}{51}, k=1,2,\dots,50} \{f_{F_{c_{th},avg}}\} \quad (4.30)$$

Table 4.19 gives the means and standard deviations of test performance $F_{max,avg}$ and F_{max} of the evolved edge detectors after using LT to obtain soft edge maps. The p -value is from the paired-sample t -test between the soft edge maps and the binary edge maps (threshold 0). Each GP edge detector has a soft edge map and a binary edge map for each test image with the same observations. The soft edge maps and the binary edge maps are dependent. Therefore, the paired-sample t -tests are employed via using the differences of the paired-samples (the soft edge maps and the binary edge maps from the same observations).

For fitness function F (without considering observation spread), the test performance $F_{max,avg}$ and F_{max} values of the evolved edge detectors after using LT have the largest standard deviations among the results from the other fitness functions in Table 4.19. It indicates that their soft edge maps from LT are very unstable. When the range of observations of

evolved edge detectors are restricted, the evolved edge detectors improve the robustness for the soft edge maps by using LT . The fitness function f_{F_s} does not have good ability to evolve edge detectors for obtaining good soft edge maps with LT . A potential reason is that f_{F_s} evaluates the observations of GP evolved edge detectors after the S-shaped transformation, not from the raw space of the observations. From these p -values, the linear transformation LT cannot improve detection performance, and significantly decreases the test performance of the evolved edge detectors. A reason is that LT makes some observations from both sides of threshold 0 map into the same grayscale level.

Figure 4.18 shows the boxplots for the GP edge detectors evolved from F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} , in terms of F_{max} and $F_{max,avg}$ respectively. From the test performance distributions, some edge detectors evolved from F have very bad test performance, being close to 0, which means that the observations of those edge detectors around threshold 0 have failed to distinguish edges after using LT . Some of the edge detectors evolved from f_{F_s} also have very bad performance. The edge detectors evolved from $f_{F_{cr}}$ and f_{F_d} do not have very bad performance (being close to 0). Therefore, without the restriction of the raw range of the observations of evolved edge detectors, the soft edge maps transformed by LT might fail to distinguish edge points and non-edge points.

To further investigate the number of results from one fitness function better than the results from another fitness function, a non-parametric test technique is employed. Table 4.20 gives the Kruskal-Wallis statistical test results for the soft edge maps of the evolved edge detectors from the four fitness functions after using LT . From the comparison results, the edge detectors evolved from f_{F_d} have significantly better soft edge maps than the edge detectors evolved from f_{F_s} when LT is utilised. Although a few of the edge detectors evolved from F have very bad performance, there are no significant differences among F , $f_{F_{cr}}$ and f_{F_d} . It seems that the restriction of the range of the observations of evolved edge detectors only

Figure 4.18: Boxplots for F_{max} and $F_{max,avg}$ of the GP evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using LT .

Table 4.20: Kruskal-Wallis statistic [174] of pairs (F_{max} and $F_{max,avg}$) on the soft edge maps of the GP evolved edge detectors after using LT .

	$f_{F_{cr}}$	f_{F_d}	f_{F_s}
F	(-, -)	(-, -)	(↑, -)
$f_{F_{cr}}$		(-, -)	(-, -)
f_{F_d}			(↑, ↑)

improve the robustness on test performance for obtaining soft edge maps by LT , but cannot improve detection accuracy. When a fixed threshold is used in the training stage, the fixed threshold might be used to obtain the best test performance on test images.

Figure 4.19 plots precision and recall on different thresholds for the average, the best and the worst of the edge detectors evolved from F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using LT . From the average (30 replications) in Figure 4.19 (a), the edge detectors evolved from $f_{F_{cr}}$ have the worst performance on soft edge maps when LT is used. The edge detectors evolved from f_{F_d} and f_{F_s} have slightly better precision on the soft edge maps than the edge detectors evolved from F when they have the same recall.

Figure 4.19: Precision and recall for the GP evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using LT .

Figure 4.19 (b) describes precision and recall on the soft edge maps of the best edge detectors from the four fitness functions respectively. Fitness function f_{F_d} has similar test performance to the fitness function F . In the low recall area, the best edge detectors from f_{F_d} and F are slightly better than the best edge detectors from $f_{F_{cr}}$ and f_{F_s} . Over the whole range, these four edge detectors have similar test performance on precision and recall.

Figure 4.19 (c) presents precision and recall on the soft edge maps of the worst edge detectors from the four fitness functions respectively. The recall values of the worst edge detectors from F and f_{F_s} at different thresholds are close to 0, which means that only a few edge points are mapped to soft edges when LT is used. Therefore, LT fails to map the observations of the edge detectors from F and f_{F_s} to soft edges. Although the performance of the evolved edge detectors from f_{F_d} and $f_{F_{cr}}$ decreases, these edge detectors can still obtain some soft edges.

Figure 4.20 visually gives the soft edge maps on image 78004 for the best test performance ($F_{max,avg}$) in the first row and the worst test performance ($F_{max,avg}$) in the second row from the evolved edge detectors. Note that, the grayscales of the soft edge maps are not easy to read, they are inverted for easy reading, and the darkest pixel (graylevel 0) has the highest

Figure 4.20: Example soft edge maps on image 78004 from the best (the first row) and worst (the second row) edge detectors from F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using LT .

response. Comparing the soft edge maps from the first row, the evolved edge detectors from F and f_{F_d} have higher contrast responses on object boundaries than the edge detectors from $f_{F_{cr}}$ and f_{F_s} . For the worse edge detectors, the results from F and f_{F_s} only contain very few edge points. The detected results from $f_{F_{cr}}$ and f_{F_d} have very low contrast, and it is hard to distinguish close graylevels in a normal print version. The detected image from $f_{F_{cr}}$ in the second row includes a few of edge points and a large number of the non-edge points. The detected image from f_{F_d} in the second row has close graylevels for most of the pixels.

To summarise, the linear transform LT is *not* good to transform the observations of the evolved edge detectors with fixed threshold 0. When the range of observations are restricted in the training stage, the robustness of the evolved edge detectors using LT to obtain soft edge maps is improved. Since a very large range of the observations of evolved edge detectors is allowed in F and f_{F_s} , an evolved edge detector with a large and sparse space for its observations has very bad performance on its soft edge maps

Table 4.21: Test performance F_{max} and $F_{max,avg}$ of the evolved edge detectors after using NT .

	F_{max}	p -value	$F_{max,avg}$	p -value
F	0.2514 ± 0.03124	0.2428	0.2653 ± 0.0353	0.1800
$f_{F_{cr}}$	0.2181 ± 0.03633	0.5096	0.2261 ± 0.0407	0.5420
f_{F_d}	0.2503 ± 0.0174	0.0000 $\uparrow$	0.2650 ± 0.0200	0.0000 $\uparrow$
f_{F_s}	0.2545 ± 0.0148	0.2809	0.2690 ± 0.0153	0.3029

by using LT .

Soft Edge Maps from S-shaped Transformation

Table 4.21 provides the test performance F_{max} and $F_{max,avg}$ on the soft edge maps by applying NT . Here, p -values are obtained from the paired-sample t -tests. Different from using LT , the soft edge maps by using the S-shaped transformation have similar or significantly better results than the binary edge maps by using threshold 0. Although the means of F_{max} and $F_{max,avg}$ from f_{F_d} after using NT increase by 0.001, they are significantly better than the relevant results with threshold 0 from the paired-sample t -tests. This indicates that an evolved edge detector from f_{F_d} improves the detection performance a bit after using NT , compared with the relevant result with fixed threshold 0. Since f_{F_s} is based on the S-shaped transformation of the observations of an evolved edge detector, the test performance using NT is stable, and it is better than using LT .

Figure 4.21 shows the boxplots for F_{max} and $F_{max,avg}$ of the evolved edge detectors from fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using NT . In contrast with Figure 4.18, using NT to obtain soft edge maps is more stable than using LT . From an overall, the soft edge maps of the evolved edge detectors from the fitness functions f_{F_d} and f_{F_s} are located in a small range when NT is used. Since the S-shaped function allows a large and

Figure 4.21: Boxplots for F_{max} and $F_{max,avg}$ of the evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using NT .

sparse range of observations of evolved edge detectors, the transformation NT does not decrease the performance. Only f_{F_d} using NT significantly improves the detection performance than using threshold 0. However, the improvement is too small.

For F , one edge detector has bad performance on its soft edge maps after using NT . A reason is that the some observations might be mapped to the same graylevel by using NT if they are very close to threshold 0 from both sides. The transformation NT can avoid the problem caused by a large and sparse range of observations (see LT), but has a problem to distinguish observations around threshold 0. Since f_{F_d} and f_{F_s} have the restriction on the position of the observations of evolved edge detectors, being away from threshold 0, the test performance of the evolved edge detectors is stable when NT is used.

Additionally, the test performance on the evolved edge detectors from $f_{F_{cr}}$ is not stable after using NT . It seems that the non-linear transformation affects the correlation ratio. Comparing the test performance between using NT and using LT , the improvement on F_{max} and $F_{max,avg}$ for fitness function $f_{F_{cr}}$ is obviously lower than the other three fitness functions.

In order to compare these results based on the ranking of all evolved

Table 4.22: Kruskal-Wallis statistic [174] of pairs ($F_{max,avg}$ and F_{max}) on the soft edge maps of the GP evolved edge detectors after using NT .

	$f_{F_{cr}}$	f_{F_d}	f_{F_s}
F	$(\uparrow, \uparrow)$	$(-, -)$	$(-, -)$
$f_{F_{cr}}$		$(\downarrow, \downarrow)$	$(\downarrow, \downarrow)$
f_{F_d}			$(-, -)$

Figure 4.22: Precision and recall of the evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using NT .

edge detectors, Table 4.22 provides the Kruskal-Wallis statistical results on the soft edge maps after using NT . It is found that the results from $f_{F_{cr}}$ are significantly worse than the results from the other three fitness functions. There are no significant differences among the results from F , f_{F_d} and f_{F_s} . It seems that the transformation NT works well on most of the evolved edge detectors for obtaining good soft edge maps.

Figure 4.22 (a) reveals the average of the precision and recall values of the evolved edge detectors from F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using NT . Also, the recall and precision values of the best and the worst edge detectors from each fitness function are shown in Figures 4.22 (b) and (c).

Firstly, the results from $f_{F_{cr}}$ in the range with not high recall are slightly

irregularly located. Since the standard deviations (of all observations, observations for edge points and observations for non-edge points) are nonlinearly changed, the results from $f_{F_{cr}}$ after using NT might change irregularly for different thresholds c_{th} . Figure 4.22 (a) indicates that the results from $f_{F_{cr}}$ are worse than the results from the other three fitness functions.

Secondly, the results from f_{F_d} and f_{F_s} in Figure 4.22 (a) are located in a small range. The results from f_{F_s} are better (higher recall) than the results from F and f_{F_d} . The results from f_{F_d} are located in a very small range. A reason for this phenomenon is that the observations of the evolved edge detectors from f_{F_d} are restricted to be away from threshold 0. If using the S-shaped transform to the raw observations, the observations are almost all mapped to only a few graylevels.

Thirdly, the results from F in Figure 4.22 (a) are located in a wide range. Compared with the results from f_{F_d} , the results from F have higher precision but lower recall, or higher recall but lower precision. Also, the results of the best edge detector from F in Figure 4.22 (b) are located in a large range. The best edge detectors from the other three fitness functions have the recall and precision values within a very short range, which are approximately located in the range of the best edge detector from F .

Lastly, the worst edge detectors from F and $f_{F_{cr}}$ obtain soft edge maps with very bad performance by using NT . However, the worst edge detectors from f_{F_d} and f_{F_s} have high recall and not very low precision on their soft edge maps after using NT . From NT , it seems that the restrictions on the distance from observations to the threshold reduces the overlap of observations around threshold 0.

Figure 4.23 visually presents the soft edge maps on image 78004 detected by the edge detector (using NT) from F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} , respectively. The first row shows the results detected by the best edge detector from each fitness function, and the second row shows the results detected by the worst edge detector. Firstly, the results detected by the best edge detector from F has lower contrast than the relevant results from the other

Figure 4.23: Example soft edge maps on image 78004 after the best (the first row) and worst (the second row) edge detectors using NT .

three fitness functions. In the detected image from F in the first row, most of the non-edge points (not bright) are not mapped to very low grayscale, which means that the relevant observations are not far away from threshold 0. Secondly, the results detected by the worst edge detector from F hardly distinguish edge points and non-edge points from the human view. A reason for this visual result is that the observations close to threshold 0 are mapped to graylevels around 127. Thirdly, the results detected by the worst edge detectors from f_{F_d} and f_{F_s} (in the second row) have better visual soft edge maps than the detected image in the second row from F . It seems that the restriction on the distance from observations to threshold 0 is effective to avoid the difficulty of distinguishing edge points and non-edge points.

To summarise, NT can be effectively used to obtain soft edge maps for the evolved edge detectors and the relevant performance is the same or slightly better than the binary edge maps using threshold 0. Compared with LT , NT obviously increases the performance of transforming the observations of evolved edge detectors to soft edge maps.

Soft Edge Maps from Bandpass Transformation

From the results transformed by LT and NT , observations with a large and sparse space or being close to threshold 0 may not be good for obtaining soft edge maps when a continuous transformation is employed. In order to address the problem, a discontinuous transformation is proposed for obtaining soft edge maps. Based on the problems from mapping observations to soft edge maps, we propose a bandpass transformation [165] for obtaining soft edge maps. Equation (4.31) defines the bandpass transformation (BT), where bt_{min} , bt_{max} , c_{bt-} , c_{bt+} and c_{th} are used to control the band filter shape, and c_{bt_w} is used to control the response magnitude of an observation. Note that this transformation utilises threshold c_{th} . In this chapter, c_{th} in BT is fixed to 0. The range of this transformation is from 0 to 1. Parameters bt_{min} and bt_{max} are used to control the band width. Parameters c_{bt-} and c_{bt+} are used to adjust the difference between the observations from the left and right sides of threshold c_{th} . The image graylevel from 0 to 255 is separated into to sub-ranges for the observations on the left and right sides of threshold c_{th} . In this study, the bandpass width is approximately considered twice that of the grayscales, namely c_{bt_w} is set to 510. For the other parameters, c_{bt-} is set to 240, c_{bt+} is set to 266, bt_{min} is set to -240, and bt_{max} is set to 250.

$$s_{bt} = BT(o) = \begin{cases} 0.0 & \text{if } o < c_{bt_{min}} \\ \frac{c_{bt-} + o}{c_{bt_w}} & \text{if } o \leq c_{th} \text{ and } o \geq c_{bt_{min}} \\ \frac{c_{bt+} + o}{c_{bt_w}} & \text{if } o \leq bt_{max} \text{ and } o \geq c_{th} \\ 1.0 & \text{if } o > c_{bt_{max}} \end{cases} \quad (4.31)$$

Note that the bandpass transformation is not the same as a normal band filter. Actually, it combines two band filters and a high filter [165]. There are different transformations in the two band filters and the high filter. Since interesting observations mainly focus on the range from the combination of the two band filters, it is still called a bandpass transformation.

Table 4.23: Test performance F_{max} and $F_{max,avg}$ of the GP evolved edge detectors after using BT .

	F_{max}	p -value	$F_{max,avg}$	p -value
F	0.2601 ± 0.0085	$0.0002 \uparrow$	0.2750 ± 0.0099	$0.0002 \uparrow$
$f_{F_{cr}}$	0.2204 ± 0.0352	$0.0304 \uparrow$	0.2284 ± 0.0389	$0.0459 \uparrow$
f_{F_d}	0.2548 ± 0.0178	$0.0000 \uparrow$	0.2687 ± 0.0203	$0.0000 \uparrow$
f_{F_s}	0.2560 ± 0.0144	$0.0024 \uparrow$	0.2703 ± 0.0145	$0.0237 \uparrow$

Table 4.23 gives the test performance on soft edge maps after the evolved edge detectors employ NT . In contrast with the binary edge maps, all results on the soft edge maps are significantly better after using BT . However, the improvement is not obvious. It is possible that threshold 0 is the best or almost close to the best threshold for the observations of evolved edge detectors to obtain binary edge maps. After using BT , the bad performance on soft edge maps for a few of the evolved edge detectors from F is avoided.

Figure 4.24 gives the boxplots on F_{max} and $F_{max,avg}$ values of these evolved detectors. These boxplots in Figure 4.24 are similar to the relevant boxplots in Figure 4.17. Therefore, the soft edge maps keep the similar distributions of the test performance on binary edge maps using threshold 0.

Table 4.24 reveals the statistics on the comparisons among the soft edge maps for these evolved edge detectors using BT . The comparison results are the same as the comparisons on binary edge maps by using threshold 0 (Table 4.18) and the comparisons on soft edge maps by using NT (Table 4.22). Except for the linear transformation, the evolved edge detectors from $f_{F_{cr}}$ are significantly worse than the evolved edge detectors from F , f_{F_d} and f_{F_s} when threshold 0, NT or BT is used. Therefore, the correlation ratio might be not good for evolving edge detectors.

Figure 4.25 shows the recall and precision values for the average, the

Figure 4.24: Boxplots for $F_{max,avg}$ and F_{max} of the evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using BT .

Table 4.24: Kruskal-Wallis statistic [174] of pairs ($F_{max,avg}$ and F_{max}) on the soft edge maps of the GP evolved edge detectors after using BT .

	$f_{F_{cr}}$	f_{F_d}	f_{F_s}
F	$(\uparrow, \uparrow)$	$(-, -)$	$(-, -)$
$f_{F_{cr}}$		$(\downarrow, \downarrow)$	$(\downarrow, \downarrow)$
f_{F_d}			$(-, -)$

best and the worst of the evolved edge detectors from the four fitness functions respectively when BT is employed. Figure 4.25 (a) shows that $f_{F_{cr}}$ has the worst performances on the averages of recall and precision of the evolved edge detectors at difference thresholds. The recall and precision values of the evolved edge detectors from f_{F_d} and f_{F_s} are located in a range with high recall and not low precision. For F , only one point (from a threshold) is closely located in the range from f_{F_d} and f_{F_s} . For the other thresholds, the average of the evolved edge detectors has high recall but low precision, or high precision but low recall. Therefore, from the recall and precision values, the evolved edge detectors from f_{F_d} and f_{F_s} are better than the evolved edge detectors from F even though the transform BT

Figure 4.25: Precision and recall of the evolved edge detectors using fitness functions F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} after using BT .

can separate both sides of observations around threshold 0.

Figure 4.25 (b) shows the recall and precision values of the best edge detector from F , $f_{F_{cr}}$, f_{F_d} and f_{F_s} respectively. As can be seen, only one point from F is close to the range from f_{F_d} and f_{F_s} . It is possible that the bandpass width c_{btw} only works on observations close to threshold 0, and a large number of the observations of the evolved edge detector are mapped around the graylevel 127. For the worst performance in Figure 4.25 (c), one point from F has precision slightly lower than 0.16, which is better than other evolved edge detectors, in terms of F .

Figure 4.26 provides example image 78004 for the relevant soft edge maps after using BT . Since there is low contrast in the detected images from F , it is hard to distinguish edges and non-edges in a print version. Most pixels are mapped around graylevel 125. The other detected images are similar to the relevant detected results in Figure 4.22.

To summarise, the transformation BT can obtain soft edge maps which at least have the same test performance (F and $f_{F_{avg}}$) as the binary edge maps from threshold 0. Also, from BT , the restriction on the range of observations and the distance from observations to threshold 0 is helpful to obtain rich responses on edge points.

Figure 4.26: An example soft edge maps on image 78004 after the best (the first row) and worst (the second row) edge detectors using BT .

4.7 Further Discussions

This section provides further discussions on the evaluation view, F -measure and FOM, the thickness of detected edges, and parameters in the bandpass transformation.

4.7.1 Pixel vs Image Views

When the test performance evaluation methods are used based on overall pixels and individual images, two detection results with significant difference in the view of overall pixels are not significantly different in the view of individual images. Therefore, a fitness function based on overall pixels is better than fitness functions based on individual images. However, to compare two different edge detectors, a test performance evaluation method based on individual images should be also considered because the comparison in the pixel view may be different from the comparison in the image view. From the comparison results, it suggests that two sets of results from different fitness functions are likely to be significantly differ-

ent in the pixel view if they are significantly different in the image view.

From the same view (overall pixels or individual images), the fitness functions F and $f_{F_{qf}}$ are not significantly different, but fitness functions based on individual images reveal significantly different results, especially for the evaluation using the geometric mean. The fitness functions $f_{F_{gm}}$ and $f_{FOM_{gm}}$ do not give good test performance, compared with the other fitness functions based on individual images. A potential reason is that the training data is sufficient to cover most edge contents in the test images, but some edge contents in the training data have low occurrences in the test images so that these edge contents are learned and some normal edge contents might reduce the influence on the learned edge detectors when the geometric mean is used as the fitness function.

4.7.2 F -measure vs FOM

From the comparison results in Table 4.11, it is found that all GP edge detectors evolved by the fitness functions based on F -measure, except for $f_{F_{gm}}$ with Set_s , are significantly better than the Sobel edge detector based on the evaluation F without a tolerance distance for predicted edge points.

However, most of these GP edge detectors are worse than the Sobel edge detector in terms of f_{FOM} . From the definition of f_{FOM} , it is known that f_{FOM} is affected by the maximum value from N_T and N_P . Since N_T is much smaller than the number of the total pixels in all test images, FOM will obtain high value if N_P is also small. While N_T is not much smaller than the number of the total pixels in the training data because of the sampling technique, the trained edge detectors with high f_{FOM} will have a high ratio of N_P to the number of the total pixels in the training data. For the test images, the edge detectors will keep the high ratio of N_P to the number of the total pixels in the test data; but N_T is small, so that test performance in terms of f_{FOM} for the GP edge detectors is not good to indicate detection results.

Also, from the view of individual images, each image may not always only choose N_{T_i} or N_{P_i} as the maximum value so that FOM from the image view fails to present the right evaluation, and it leads to some f_{FOM_i} being larger than 1 if only using N_{T_i} or N_{P_i} . For example, f_{FOM} only chooses the sum of N_{T_i} . It is possible that N_{P_i} is larger than N_{T_i} in a detected image. Since the minimum value between N_{T_i} and N_{P_i} in the image is chosen, the relevant f_{FOM_i} is very likely larger than 1. Therefore, $f_{FOM_{avg}}$ is more reasonable than f_{FOM} , and the test performance based on $f_{FOM_{avg}}$ is much closer to F than f_{FOM} .

Regarding the drawbacks existing in f_{FOM} , the comparison based on f_{FOM} and F after one-to-one assignment reveals that f_{FOM} is not good to measure the detection results from F and f_{FOM} with $Set_{s,b}$. The average of f_{FOM} of the test performance on the GP edge detectors from the fitness function F with $Set_{s,b}$ is 0.4569, which is lower than the result from the fitness function f_{FOM} with $Set_{s,b}$ (0.4949). However, the evaluation based on F after one-to-one assignment shows that the GP edge detectors from F with $Set_{s,b}$ are significantly better than the GP edge detectors from f_{FOM} with $Set_{s,b}$ (in Table 4.13). Therefore, the evaluation based on F without offset is still better than f_{FOM} .

4.7.3 Thickness of Detected Edges

From the detected example images, the evolved edge detectors give thick responses on edges, which requires a post-processing technique such as non-maximum suppression [28] or a thinning operation [106]. A potential reason for the thickness is that the ground truth for the training images are combined from several human observations and some edges are more than one pixel width. Another potential reason, for the fitness functions based on F -measure, is that high recall exists in these edge detectors. An offset from a predicted edge point to the relevant true edge point is not allowed, which leads to low precision for most GP edge detectors. To

obtain high fitness, GP edge detectors prefer choosing high recall in early stages of evolution. The overlap for predicted edge points to a true edge point is allowed in the fitness functions based on FOM, which is another potential reason for thick responses from the GP edge detectors evolved by these fitness functions.

4.7.4 Parameters in Bandpass Transformation

The three transform functions are employed to analyse observations of GP evolved edge detectors. The transform BT has different parameters (see Equation (4.31)) to control the band width. Figure 4.25 (a) shows most of the observations of the evolved edge detectors are around threshold 0. In order to increase the contrast for the relevant graylevels of the observations around threshold 0, the band width $c_{bt_{rg}}$ is decreased to 100, c_{bt_-} is set to 40, c_{bt_+} is set to 60, bt_{min} is set to -40 , and bt_{max} is set to 40.

The test performance ($f_{F_{avg}}$) is 0.2739 ± 0.0099 , which is not significantly different from the results from $c_{bt_{rg}}$ using 510. Figure 4.27 shows the recall and precision values of average, the best and the worst of the evolved edge detectors from F . From the results, smaller $c_{bt_{rg}}$ makes the points with too low precision or too low recall disappear. In contrast with Figure 4.25, some points with low precision are moved to a range with higher precision.

Since the band width $c_{bt_{rg}}$ becomes small, the contrast between the graylevels mapped from observations around threshold 0 becomes high. Figure 4.28 shows the image 78004 detected by the best and worst edge detectors. Clearly, the edge map from the worst edge detector can be seen from the print version, but the edge map from the best edge detector decreases the contrast. Therefore, the parameter $c_{bt_{rg}}$ should not be small. To distinguish observations around threshold 0, the difference between c_{bt_-} and c_{bt_+} should be increased. From the change of parameters in BT , it also suggests that the range of observations should be addressed in the fitness

Figure 4.27: Scatter plots of recall and precision values of average, the best and the worse of the evolved edge detectors from F using $c_{bt_{rg}} = 100$.

(a) Best

(b) Worst

Figure 4.28: An example soft edge maps on image 78004 after the best and worst edge detectors using BT width $c_{bt_{rg}} = 100$.

function. Although the threshold is the best value or close to the value for evolved edge detectors to discriminate pixels, the response magnitudes on true edges might be not suitably distributed.

4.8 Chapter Summary

The goal of this chapter is to investigate different evaluation criteria for evaluating low-level edge detectors evolved by GP. This investigation was achieved by employing F -measure and FOM from the pixel view and the image view, the matching directions in FOM, and observation spread when

F -measure is used for obtaining soft edge maps.

The results indicate that F -measure as a fitness function is better for the evaluation based on overall pixels than individual images, but FOM is better for evaluation based on individual images than overall pixels when the training dataset is large. When the tolerance distance from a detected edge point to a true edge point is not allowed, the ETHZ image dataset shows that GP edge detectors are better than the Canny edge detector.

From the comparisons of the fitness functions based on overall pixels and individual images, the geometric mean is not good to evaluate trained edge detectors' fitness when the image dataset is large. When the results detected by the evolved edge detectors from different fitness functions are not significantly different based on the evaluation from the image view, they might be significantly different from the pixel view.

The optimal matching results for the edge detectors from the fitness functions F and f_{FOM} show that F is better than f_{FOM} to evolve edge detectors.

When evaluating both matching directions in FOM as a combination using the F -measure, the detection results show improved detection accuracy in terms of f_{FOM} . Compared with the results from F , the proposed combination has better performance when the test performance evaluation utilises f_{FOM} and $f_{FOM_{avg}}$.

When soft edge maps from evolved edge detectors are considered, the range of observations of evolved edge detectors should be addressed in the fitness function. Three different transformations indicate that the observations with small range and far away from a fixed threshold are good to obtain soft edge maps.

This chapter and Chapter 3 analysed how GP evolves edge detectors. The search operators used in the GP system implicitly select pixels to construct edge detectors. In order to further understand how GP evolves edge detectors, the next chapter will investigate pixels selected by the GP system.

Chapter 5

Pixels Implicitly Selected by GP

5.1 Introduction

Chapters 3 and 4 employed GP based on raw pixels from full images to evolve low-level edge detectors. The search operators $s_{n,m}$ based on single pixels implicitly search important pixels, which avoids manually setting a window size for constructing edge detectors. The GP evolved edge detectors show that $s_{n,m}$ is helpful to efficiently construct edge detectors with fast detection speed. In order to understand how well evolved GP programs work for specific tasks, further investigation on the evolved programs is needed.

Comprehensibility is one of major setbacks of GP in many areas [44, 96, 97]. Researchers have tried to understand the programs evolved by GP [6, 62, 71, 96]. GP (evolved) edge detectors are initially randomly generated, then automatically evolved by the evolutionary process guided by the fitness evaluation. To truly understand the structure of evolved edge detectors, very strong and wide expertise on edge detection is needed. It is a very challenging task to analyse GP edge detectors. Since most of the existing methods [57, 140] extract features based on explicitly given pixels, the analysis of pixels implicitly selected by GP helps us understand whether GP automatically selects a compact but sufficiently rich set of pix-

els to construct edge detectors.

There are two problems for analysing the pixels selected by GP. Firstly, it is not clear which pixels GP will implicitly select to construct edge detectors. From most of the existing edge detectors based on raw pixels, the employed pixels around their discriminated pixel are usually symmetric [57, 140]. The localisation of pixels implicitly selected by GP is unknown in advance and needs be investigated. Secondly, it is not clear whether pixels selected by GP are better than the pixels from a fixed window to construct edge detectors, such as linear edge filters and second-order edge filters. Here a linear edge filter is constructed by the sum of pixel intensities with weights, and a second-order edge filter is constructed by the sum of pixel intensities with weights and products of two pixel intensities (all combinations) with weights. It is very desirable to conduct investigation on the pixels selected by GP.

5.1.1 Chapter Goal

The goal of this chapter is to analyse which pixels are implicitly selected by GP based on full images. Since the search operators used in Chapter 3 work on pixels with very little prior knowledge, neighbourhoods for discriminating pixels as edge points or non-edge points are investigated. Specifically, the following research objectives will be investigated.

- What characteristics (e.g. relative pixel positions) exist in a set of raw pixels selected by GP.
- Whether a set of raw pixels selected by the GP edge detectors is better to construct linear and second-order filters than using all pixels in a fixed window.
- Whether these estimated filters are better than common existing low-level edge detectors, such as the Laplacian and Sobel edge detectors.

- Whether the GP edge detectors are better than the above estimated filters.

5.1.2 Chapter Organisation

In the remainder of this chapter, the second section investigates extracting pixels from GP evolved low-level edge detectors and how to construct linear and second-order edge filters. The third section describes the experiment settings. The fourth section presents the results of the experiments with discussions. The fifth section provides further discussions. The last section draws a summary of this chapter.

5.2 Analysing Pixels Selected by GP

There are three steps for analysing pixels selected by GP. The first step is to evolve edge detectors using GP. The second step is to extract pixels implicitly selected by GP. The last step is to construct edge filters using only the pixels extracted in the second step. Figure 5.1 shows the process flow to find neighbours of a discriminated pixel. After analysing a full image (terminal) and $s_{n,m}$ in the evolved edge detectors, a set of neighbours for a discriminated pixel will be obtained. In the Neighbourhood part of Figure 5.1, empty cells in the 5×5 window are considered to be redundant for constructing edge filters.

5.2.1 Settings for Evolving Edge Detectors

In order to quickly obtain evolved edge detectors, six images are selected from the BSD training image dataset set. Figure 5.2 shows these images. The six images are selected based on different edge content. It is considered that they are (reasonably) sufficient to evolve good edge detectors based on the initial experiments in Chapter 4. In order to reduce the com-

Figure 5.1: Process of finding pixels to construct edge filters.

putational cost, five different subimages of size 41×41 from each image are randomly sampled as training subimages.

The terminal and function sets used are the setting Set_s in Table 3.1 (on page 67); namely only the search operator $s_{n,m}$ with a single pixel is used for searching neighbours. The experiment settings are the same as the settings used in Section 3.4 (on page 66), except for the population size. Since the training data is small, a smaller popular size (500) is used for evolving edge detectors. We use 30 independent runs to generate 30 (GP evolved) edge detectors.

5.2.2 Methods for Extracting Pixels

Directly Extracting Pixels

The search operator $s_{n,m}$ implicitly finds the neighbours in GP evolved edge detectors. The three neighbours used in GE_{Robert} (see Equation (3.4) on page 60) can be expressed by $s_{-1,-1}(I)$, $s_{-1,0}(I)$ and $s_{0,-1}(I)$. Based on $s_{n,m}$ and image I , the selected pixels can be found. Figure 5.3 shows the pixels existing in the 30 GP edge detectors with at least 15 occurrences. The window presents pixels (neighbours) selected around the centre (discriminated) pixel from $s_{n,m}$. The number of occurrences of the selected neigh-

Figure 5.2: Six training images from BSD dataset.

bours are calculated based on the final terminal I . If a pixel only exists in a few GP edge detectors, it will be considered as giving limited contribution for detecting edges. The threshold for the pixel occurrences in these edge detectors is chosen a half of 30 (runs), namely at least 15. Note that a pixel might have several occurrences in a single GP edge detector. All pixels with at least 15 occurrences are located in the 5×5 window. Pixels with less than 15 occurrences are not shown, and a few pixels are located outside of the 5×5 window. Considering the centre pixel is the original position ($j = (0, 0)$), a two-dimensional position index (j) in the window contains the horizontal direction (right for the positive) and the vertical direction (down for the positive). The greatest number of occurrences (47) is from the discriminated pixel ($j = (0, 0)$), and the second greatest number of occurrences (36) is from the right neighbour ($j = (1, 0)$). The average number of occurrences of the discriminated pixel itself in the 30 GP edge detectors is more than 1.5 in each evolved detector.

Another interesting observation is that the pixels with high occurrences are diagonal pairs, such as $(1, 0)$ with $(0, 1)$, $(2, 0)$ with $(0, 2)$, $(0, -1)$ with

	17	20		
	25	47	36	29
		19	22	
		20		

Figure 5.3: Neighbours (at least 15 occurrences) selected from 30 GP edge detectors. Note that numbers are the frequency of occurrences of the neighbours.

$(-1, 0)$, $(1, 1)$ with $(-1, -1)$. Diagonal pairs have been investigated by applying diagonal derivatives for constructing edge features in our previous work [52] which shows that the edge features from the diagonal derivatives are better than the edge features from the horizontal and vertical derivatives, in terms of detection accuracy. Set_{nb1} will now be used to indicate the nine particular pixels in Figure 5.3 (including the discriminated pixel itself).

Extracting Pixels based on Orientations

Taking into account the similarity in the pixels around the same direction, a merging operation is proposed here to reduce the number of selected pixels and find a more rich set of pixels around a discriminated pixel in these GP edge detectors. In a function node of a tree-based program, the pixels selected from a subtree will be merged together if they are in the same direction. For instance, $(1, 1)$ and $(1, 0)$ are considered as the same direction (the direction only based on n or m from $s_{n,m}$, here n is chosen), and only $(1, 1)$ or $(1, 0)$ is a selected pixel. Figure 5.4 shows a simple example tree to extract neighbours by using this merging operation. In $s_{1,0}(I) + s_{1,1}(I)$, the node $+$ merges $\{(1, 0)\}$ from $s_{1,0}(I)$ and $\{(1, 1)\}$ from $s_{1,1}(I)$ to $\{(1, 0)\}$. The root node $-$ has $\{(1, 2), (2, 1)\}$ as the positions for the relevant selected

Figure 5.4: Example of the merging operation on a simple GP tree.

neighbours. This merging operation is not equal to the simple combination of the final selected pixels, because it is based on function nodes, not final terminals.

Figure 5.5 gives the selected pixels with at least 15 occurrences. Firstly, there are only 6 pixels selected. The number of selected pixels is less than the number of pixels in Set_{nb1} . Secondly, the number of selected pixels adjacent to the discriminated pixel is only 3, and these pixels are not connected to each other. Thirdly, two selected pixels in the positions (2, 3) and (3, 2) are not located in the 5×5 window. Lastly, these pixels have no diagonal pairs, compared with Set_{nb1} . Using this merging operation, the selected pixels are scattered within a local area. Set_{nb2} will now be employed to indicate these six particular pixels in Figure 5.5.

5.2.3 Filters Constructed from Selected Pixels

To analyse the pixels selected from the GP edge detectors, a technique is required to use these pixels to construct edge detectors to validate whether the set of the selected pixels is good for edge detection. Filter techniques are popular for extracting edge features when fast response speed in real-time applications is required. To analyse a set of pixels selected from GP, linear and non-linear (second-order) filters are built using only these selected pixels to perform edge detection. These filters are expected to allow

	22	63		171		
			181			
						37
					135	

Figure 5.5: Pixels (at least 15 occurrences) selected from 30 GP edge detectors with a merging operation. Note that numbers are the occurrences for the neighbours.

only one pass over the images so that they give responses as fast as possible and no extended memory is required, which is good for processing a sequence of images and in hardware design (considering limited memory and data buses) for the implementation of a filter with fast response speed.

Linear Filter

The linear filter LF is defined in Equation (5.1), where i is a discriminated pixel from image I , Set_{nb} is the used neighbours (including i), i_j is the pixel intensity (of neighbour j), β_j is the coefficient (weight) for i_j , which needs to be optimised, and b is the intercept, being similar to a threshold value. Note that j is a two-dimensional position index for these pixels in Equation (5.1), and b can be considered as β_0 . For example, $j = (0, -1)$ indicates the left neighbour of the discriminated pixel i .

$$LF(i) = \sum_{j \in Set_{nb}} \beta_j i_j + b \quad (5.1)$$

A linear filter in [2] includes β_j for the pixels in the 3×3 window ($Set_{nb} = \{(0, 0), (-1, 0), (-1, -1), (0, -1), (1, -1), (1, 0), (1, 1), (0, 1), (-1, 1)\}$). The sum of β_j must be 0 in the filter, and b must be 0. The threshold for the

filter needs to be solved by an adaptive algorithm, which takes some computational cost. In addition, the training image is only based on a binary image. However, images used in Equation (5.1) are based on grayscale levels, being more complicated than binary images. In order to rapidly detect edges in this chapter, the fixed threshold 0 is used for $LF(i)$, discriminating i as an edge point when $LF(i) > 0$.

Second-order Filter

With regard to approximating other existing edge detectors, a second-order filter SF is defined in Equation (5.2), where $\sum_{j,l \in Set_{nb}}$ indicates all combinations of neighbours i_j and i_l for discriminated pixel i , and $\beta_{j,l}$ is the coefficient for the combined factor $i_j i_l$. Since $i_j i_l$ is equal to $i_l i_j$, they are considered as only one combination and only $\beta_{j,l}$ is used for $i_l i_j$ and $i_j i_l$. Based on the limitation of only one pass over an image, the existing edge detectors, such as the Sobel detector can be expressed by the equation without using a square root.

$$SF(i) = \sum_{j \in Set_{nb}} \beta_j i_j + \sum_{j,l \in Set_{nb}} \beta_{j,l} i_j i_l + b \quad (5.2)$$

Parameter Estimation

A hybrid Particle Swarm Optimisation with Differential Evolution algorithm (PSO-DE) has shown strong ability to find global optima for non-linear optimisation problems [51, 54]. Therefore, PSO-DE is employed to estimate the parameters in the linear and second-order filters, The detection accuracy based on F is used as the objective function, so the estimated parameters are based on maximising the value of F . The details of this algorithm is beyond the scope of the objectives, and can be seen from [51, 54].

The objective function F only uses the response sign (edge points or non-edge points) of a filter, and cannot indicate the response magnitudes.

When the parameters β_j or $\beta_{j,l}$ change by a small amount, F might not be changed. To clearly indicate the changes of the estimated parameters, a correlation ratio variant from [18], $f_{F_{cr}}$ (see Equation (4.27) on page 119) is employed as a new objective function.

5.3 Experiment Settings

This section describes which pixels are used to construct edge filters.

5.3.1 Pixels Used to Construct Filters

From Equations (5.1) and (5.2), different sets of pixels (Set_{nb}) are used to construct edge filters. The pixels from 3×3 and 5×5 windows are used to compare with the pixels selected from GP. Note that two types of sets of pixels are used to optimise the linear and second-order filters. The first utilises a fixed area, such as the 3×3 and 5×5 windows. The other utilises a set of pixels which might be scattered around a discriminated pixel, and these pixels are selected by the GP edge detectors with high occurrences.

5.3.2 Baselines for the Comparisons

The Laplacian and Sobel edge detectors are selected as the baselines from the existing edge detectors. There are two reasons to only choose these two edge detectors. Firstly, the two detectors are popular low-level edge detectors and do not use pre-processing and post-processing [152]. Since pre-processing and post-processing techniques can be routinely and effectively applied to the outputs of both edge detectors, this chapter only focuses on the outputs themselves. The other low-level edge feature extraction methods, such as the Canny detector, can be considered as a combination of methods for extracting edge features and techniques for pre-processing and post-processing. A Canny detector can be approximated

by the combination of a Gaussian filter (pre-processing), a Sobel detector (feature extraction), and a non-maximum suppression (post-processing) [28]. Here, the GP edge detectors and estimated filters can use the non-maximum suppression technique as well. Secondly, the investigation is on low-level edge detectors constructed by raw pixels. For fair comparison, only edge detectors constructed by raw pixels are selected. The Laplacian edge detector is a linear filter based on raw pixels, and the Sobel edge detector can be approximated as a second-order edge detector based on raw pixels. To validate the choice of a compact but rich set of pixels selected by GP, the linear and second-order filters are employed. Therefore, these estimated filters based on the selected pixels are fairly compared with existing linear and second-order filters.

The selected pixels from GP edge detectors are different from the pixels used in the Laplacian and Sobel edge detectors. In order to check whether the pixels extracted from GP edge detectors are good to construct edge detectors, the pixels in the 3×3 and 5×5 windows are used to construct filters with the same approach. The filters based on the 3×3 and 5×5 windows are also used as baselines for the comparisons in this chapter.

5.4 Results and Discussions

This section provides the results from GP and estimated filters with discussions.

5.4.1 Images Detected by GP

Figure 5.6 gives four example detected binary images from the best GP edge detector, and the Sobel and Laplacian edge detectors with the best threshold value for each detector (based on the maximum F). The four example images come from the BSD test image dataset. These four images have different edge content such as a high contrast object in image

Figure 5.6: Four example test images from BSD dataset detected by the best GP, Sobel and Laplacian (“Lap”) edge detectors.

106024 and low contrast areas in image 299086. The threshold of the Sobel detector for the maximum F is 0.1765, and is 0.03922 for the Laplacian detector. From Figure 5.6, the Laplacian detector is sensitive to noise and has poor ability to find edges of the right part in image 45096. Compared with the Sobel detector, the GP edge detector has similar detection results for the four images, but the GP edge detector has better ability to detect distinct edges, e.g., the edges in the right part in the image 45096 and image 106024. For image 253055, the edges of the clouds are not clear and most

observers do not mark them as edges. The GP edge detector matches most of the observations, and does not give responses on the boundaries of the clouds. However, the GP edge detector fails to detect the right and bottom boundaries of the pyramid in image 299086 because the edges are not distinct. The Sobel edge detector has slightly better detection for these edges than the GP edge detector, but is affected by the sand texture noise. From the results, GP has good ability to find edges from the clear discontinuities, and suppresses the noise but possibly lost edges that are difficult to distinguish.

5.4.2 Optimised Linear Filters

Table 5.1 gives the averages and standard deviations of F values of the linear filters constructed by all pixels in 3×3 and 5×5 windows, Set_{nb1} (Figure 5.3) and Set_{nb2} (Figure 5.5) based on Equation (5.1) with objective functions F and $f_{F_{CR}}$. Here each setting for Set_{nb} is used to indicate the relevant filters. Figure 5.7 shows the performance of these linear filters using the means and standard deviations of their F values. Note that the training data (*randomly* sampled from the six images) is not the same in the 30 runs, so the difference for the estimated parameters still exists. From Table 5.1 and Figure 5.7, the test performance F for the parameters of these linear filters estimated by the objective function F is better than the objective function $f_{F_{CR}}$, but the linear filters estimated by $f_{F_{CR}}$ have more stable performances than the filters estimated by F .

Table 5.2 gives the comparisons of the test performance F among the estimated linear filters, the GP edge detectors and the Laplacian and Sobel edge detectors based on multiple one-sample t -tests with overall significance level 0.05 and p -value adjustment using Holm's method [75] over all comparisons. Here, " $\downarrow$ " indicates that the edge detector from the first column is significantly worse than the edge detector from the first row, " $\uparrow$ " for the significantly worse indication (the first column to the first row),

Table 5.1: Test performance (F) on 100 BSD test images for linear filters estimated by PSO based on objective functions F and $f_{F_{CR}}$.

Detector	F	$f_{F_{CR}}$
3×3	0.1786 ± 0.0029	0.1611 ± 0.0002
5×5	0.2022 ± 0.0025	0.1685 ± 0.0001
Set_{nb1}	0.1952 ± 0.0045	0.1743 ± 0.0002
Set_{nb2}	0.1797 ± 0.0016	0.1631 ± 0.0001

Figure 5.7: Test performance (F) averaged over the 100 BSD test images for linear filters estimated by PSO-DE based on objective functions F and $f_{F_{CR}}$.

and “—” for no significant difference. In each pair, the first comparison is based on the objective function F , and the second is based on the objective function $f_{F_{CR}}$. The comparisons between the GP edge detectors and existing edge detectors have been addressed in Chapters 3 and 4, this chapter mainly focuses on comparisons among the estimated filters.

There are six interesting comparisons from Table 5.2. Firstly, all estimated linear filters are significantly better than the 3×3 Laplacian edge detector. It seems that linear filters optimised by the hybrid PSO-DE based

Table 5.2: Comparisons (F, f_{FCR}) among linear filters, GP edge detectors (only from fitness function F), the Laplacian detector, and the Sobel detector.

	5×5	Set_{nb1}	Set_{nb2}	GP	Laplacian	Sobel
3×3	(↓, ↓)	(↓, ↓)	(-, -)	(↓, ↓)	(↑, ↑)	(↓, ↓)
5×5		(↑, ↓)	(↑, ↑)	(↓, ↓)	(↑, ↑)	(↓, ↓)
Set_{nb1}			(↑, ↑)	(↓, ↓)	(↑, ↑)	(↓, ↓)
Set_{nb2}				(↓, ↓)	(↑, ↑)	(↓, ↓)

on ground truth are better to detect edges than the Laplacian edge detector, in terms of test performance F . Secondly, the linear filters based on the 5×5 window have the best detection performance based on the average of F values for the 100 test images in the four estimated linear filters with the objective function F , but the linear filters constructed by Set_{nb1} have very close test performance to the 5×5 window linear filters according to the averages of F values in Table 5.1. Thirdly, all the estimated linear filters by the objective function f_{FCR} have close test performance, but the linear filters constructed by Set_{nb1} are significantly better than the other estimated linear filters. Fourthly, the estimated linear filters cannot compete with the Sobel and GP edge detectors. In general, the GP edge detectors are non-linear, so it seems that the non-linear edge detectors are better than the linear edge detectors. Fifthly, the estimated linear filters from Set_{nb1} have better test performance than the estimated linear filters from Set_{nb2} . Finally, the edge detectors constructed by Set_{nb1} are significantly better than the estimated 3×3 window linear edge detectors, and there are no significant differences between the linear edge detectors constructed by Set_{nb2} and the estimated 3×3 window linear edge detectors. Although Set_{nb1} and Set_{nb2} are not as good as the estimated 5×5 window edge detectors by the objective function F , these estimated linear edge detectors show

Table 5.3: Test performance (F) on 100 BSD test images for second-order filters estimated by PSO based on F and $f_{F_{CR}}$.

Detector	F	$f_{F_{CR}}$
3×3	0.2265 ± 0.0050	0.2204 ± 0.0028
5×5	0.2387 ± 0.0151	0.2380 ± 0.0025
Set_{nb1}	0.2390 ± 0.0073	0.2340 ± 0.0033
Set_{nb2}	0.2501 ± 0.0031	0.2401 ± 0.0015

Figure 5.8: Test performance (F) averaged over the 100 BSD test images for second-order filters estimated by PSO-DE based on F and $f_{F_{CR}}$.

that the pixels selected by GP contain rich information for constructing edge features based on the overall comparisons.

Optimised Second-order Filters

Table 5.3 gives the means and standard deviations of test performance F of the second-order filters constructed by all pixels in 3×3 and 5×5 windows, Set_{nb1} and Set_{nb2} based on Equation (5.2) with objective functions F and $f_{F_{CR}}$. Figure 5.8 shows the error-bars of these second-order filters based on the test performance using F . Different from the estimated linear filters,

Table 5.4: Comparisons (F, f_{FCR}) among second-order filters, GP edge detectors (only from fitness function F), the Laplacian detector, and the Sobel detector.

	5×5	Set_{nb1}	Set_{nb2}	GP	Laplacian	Sobel
3×3	(↓, ↓)	(↓, ↓)	(↓, ↓)	(↓, ↓)	(↑, ↑)	(−, ↓)
5×5		(−, −)	(↓, ↓)	(↓, ↓)	(↑, ↑)	(−, −)
Set_{nb1}			(↓, ↓)	(↓, ↓)	(↑, ↑)	(−, −)
Set_{nb2}				(−, ↓)	(↑, ↑)	(↑, ↑)

these second-order filters estimated by the objective functions F and f_{FCR} have close test performance (the average of values F). The second-order filters constructed with all pixels in Set_{nb2} have the highest average of F values. However, the standard deviations of the test performance of the second-order filters are higher than the linear filters.

Table 5.4 gives the comparisons of the test performance F among the estimated second-order filters, the GP edge detectors, and the Laplacian and Sobel edge detectors. There are several interesting comparisons in Table 5.4: firstly, the estimated second-order filters constructed with Set_{nb2} have the best test performance among the estimated second-order filters with the objective function F or f_{FCR} ; secondly, the estimated second-order filters constructed with Set_{nb1} can compete with the Sobel edge detector; thirdly, the estimated second-order filters constructed with Set_{nb2} based on the objective function F or f_{FCR} are significantly better than the Sobel detector and the estimated 5×5 window second-order filters; fourthly, the estimated second-order filters constructed with Set_{nb1} or Set_{nb2} are significantly better than the estimated 3×3 window second-order filters; and lastly, the estimated second-order filters with Set_{nb2} based on the objective function F can compete with the GP edge detectors.

From the comparisons, it seems that the pixels selected by GP with

high occurrences are compact but rich for constructing low-level edge detectors. Although the estimated 5×5 window second-order filters are significantly better than the estimated 3×3 window second-order filters, the estimated second-order filters with Set_{nb1} can compete with the estimated 5×5 window second-order filters. Since Set_{nb1} is included in the 5×5 window, it shows that there are redundancies existing in that area for constructing edge detectors. The number of pixels in Set_{nb1} is the same as the number of pixels in the 3×3 window. It seems that pixels in Set_{nb1} are better than the pixels in 3×3 window for constructing edge detectors. Although the number of pixels in Set_{nb2} is less than the number of pixels in the other filters, the estimated second-order edge detectors based on Set_{nb2} outperform the others, except for the GP edge detectors. Therefore, if the redundancy in the same direction is considered, the pixels selected by GP are more efficient and richer for constructing second-order filters than the pixels in Set_{nb1} . In addition, these pixels in Set_{nb2} are not all located in the 5×5 window.

5.4.3 Estimated Parameters for a Second-Order Filter

Since the second-order filters have similar test performance F to the GP evolved edge detectors, a second-order filter is given as an example. To present the best second-order filter constructed by Set_{nb2} with F , the i_j indices (j for pixel positions) in Equation (5.2) have $Set_{nb} = \{(-2, -1), (-1, -1), (0, 1), (1, -1), (2, 3), (3, 2)\}$. The estimated parameters (β_j by the position order of neighbours in the Set_{nb}) with four-decimal-point precision are shown in the second row of Table 5.5. The estimated parameters of the combined factors ($\beta_{j,l}$ by the position order in the Equation (5.2)) are shown from the third row to last row of Table 5.5. The overall test performance F is 0.2574.

Figure 5.9 shows the four corresponding detected images. These detected images are similar to the results detected by the GP edge detector

Table 5.5: Estimated parameters for an example second-order filter.

	$(-2, -1)$	$(-1, -1)$	$(0, 1)$	$(1, -1)$	$(2, 3)$	$(3, 2)$
$(-2, -1)$	-3.5443	-3.0891	-4.8875	-18.1688	-4.0319	-18.5054
$(-1, -1)$	-2.5557					
$(0, 1)$	20.0000	-3.9357				
$(1, -1)$	-4.2809	1.4129	-5.8742			
$(2, 3)$	-3.6624	0.9682	-9.7323	-9.9464		
$(3, 2)$	5.1471	13.1088	-7.9164	-7.4543	-9.7803	
$(3, 2)$	11.6844	-9.9062	6.3426	13.1130	1.7990	0.2317

Figure 5.9: Four example detected images by a second-order filter constructed with Set_{nb2} .

and the Sobel edge detector (see Figure 5.6). However, compared with the Sobel edge detector, the second-order filter here finds more edge points in images 45096 and 106024. For image 253055, the second-order filter does not give responses on the boundaries of the clouds, which is the same as the GP edge detector. For image 299086, the second-order filter detects some part of the edge of the right boundary of the pyramid, and rejects

more noise than the Sobel edge detector.

5.5 Further Discussions

This section discusses interesting phenomena occurring in this experiment, mainly focusing on the selected neighbours used to construct low-level edge detectors.

5.5.1 Objective Functions F and f_{FCR}

Robustness

To estimate the parameters of a linear or second-order filter, PSO-DE is employed. The work in reference [54] shows that this method is effective. However, the standard deviations of the test performance for all estimated filters from the objective function F are higher than the standard deviations of the test performance for the filters from the objective function f_{FCR} . The reason is that F is only based on the sign of the response. A slight change of a parameter in the filter affects the response magnitudes, but the sign of the response still stays the same. Therefore, the same F value in the training stage allows different values for the estimated parameters of a linear or second-order filter. A change of estimated parameters for a filter based on f_{FCR} brings a difference in f_{FCR} because f_{FCR} contains the information of the response magnitudes of the filter. The final solutions for these estimated parameters are almost the same.

Accuracy

From the test performance F of the estimated linear and second-order filters in Figures 5.7 and 5.8, the fitness function F is better than f_{FCR} . Since f_{FCR} indicates linear dispersion [48], when the two classes are difficult to separate by a threshold, f_{FCR} prefers the response magnitudes with a small

range, but loses some accuracy. However, when the filter has some ability to distinguish edge points from non-edge points, the accuracy in f_{FCR} becomes very important. That is why the second-order filters estimated by f_{FCR} have similar detection performance to the second-order filters estimated by F .

In summary, F is good for estimating the continuous parameters if only the detection accuracy is considered. However, f_{FCR} has more robust performance, and it can be used for estimating the parameters of a second-order filter, but it is not good at estimating the parameters of a linear filter due to low accuracy.

5.5.2 Linear and Second-order Filters

The estimated second-order filters are better than the estimated linear filters, but are more complicated. The estimated linear filters are better than the Laplacian edge detector, but are worse than the Sobel edge detector. A potential reason is that the linear relationship is not sufficient to suppress noise in the natural images. Another reason is that the estimated filters use a fixed threshold 0, but the performance for the Sobel edge detector is the maximum F on the test images from a set of thresholds after normalising the detection results. If the Sobel edge detector directly uses a fixed threshold based on training images, without normalising, the performance might decrease a bit. The best 5×5 window linear filter and the best linear filter constructed with all pixels in Set_{nbl} still have test performance close to the Sobel edge detector.

5.5.3 Number of Pixels Used to Construct Edge Detectors

The performance of the estimated filters has no obvious relationship with the size of the set of pixels for constructing them. Although the estimated 5×5 window filters are better than the estimated 3×3 window filters based on the construction of a linear filter or a second-order filter, the fil-

ters constructed by all pixels in Set_{nb2} are significantly better than the 3×3 window filters based on the linear construction or the second-order construction. A 3×3 window filter contains nine pixels, but a filter using Set_{nb2} only has six pixels.

Only considering the estimated linear filters, a set of pixels with a large size is good to construct linear filters, but there might be some relationships among these pixels. The estimated 5×5 window linear filters have 25 pixels and have the best test performance. However, the filters constructed with all pixels in Set_{nb1} are better than the 3×3 window linear filters. Since Set_{nb1} contains pixels (with diagonal pairs) from the 5×5 window, the linear filters constructed with Set_{nb1} are better than the linear filters constructed with Set_{nb2} , and they are close to the 5×5 window constructed linear filters. Although Set_{nb2} contains valuable pixels, the linear relationship among the pixels is not good to detect pixels as edge points or non-edge points. Therefore, when extracting neighbours Set_{nb} from GP edge detectors to construct linear edge detectors, the redundancy in a direction (not using the merging operation) should be kept.

Since a second-order filter has a stronger ability to disperse (split) two classes than a linear filter, the constructed second-order filters have better test performance. The 5×5 window second-order filters are better than the 3×3 window second-order filters, which indicates that all pixels in the 5×5 window are better to construct edge filters than all pixels in the 3×3 window. The second-order filters constructed with Set_{nb2} are better than the 5×5 window second-order filters, which indicates the pixels from Set_{nb2} are richer to construct edge detectors than the pixels in the 5×5 window. From the results based on Set_{nb1} and Set_{nb2} , a filter constructed with Set_{nb2} contains fewer pixels, but provides better test performances. This suggests that the directional redundancy in the pixels for constructing second-order filters can be removed. In addition, two pixels in Set_{nb2} are not located in the 5×5 window; it seems that the pixels slightly further away are also helpful for discriminating pixels.

Figure 5.10: The convergence for the estimated second-order filters with fitness function F in the training stage.

The number of the parameters in a second-order filter is now analysed to determine whether it affects the final solution. For a second-order filter, the number of parameters is $n_x + \frac{n_x(n_x+1)}{2}$, where n_x is the number of pixels to construct the filter, and the intercept b is not considered. Therefore, a 3×3 window second-order filter has 54 estimated parameters, 299 estimated parameters for a 5×5 window second-order filter, 54 estimated parameters for a second-order filter with Set_{nb1} , and 27 estimated parameters for a second-order filter with Set_{nb2} . Figure 5.10 shows the convergence for the four estimated second-order filters with the fitness function F . Here, the vertical axis is the average of F values of the best solutions (30 replications) in each generation. From Figure 5.10, the values of the objective functions from the these estimated filters converge to a constant or are very close to a constant. Since the hybrid PSO-DE has good ability to find good solutions, the number of these estimated parameters does not strongly affect the final solution. However, in the training stage, the estimated filters with Set_{nb1} are slightly better than Set_{nb2} . Whether there is over-fitting existing in the estimated filters with Set_{nb1} will be further work.

In summary, based on two different strategies to select pixels from GP edge detectors to construct low-level edge features, Set_{nb1} and Set_{nb2} indicate that GP automatically selects compact but rich sets of pixels for the edge detector construction. If the limited ability of dispersion for a linear filter is considered, pixels in the GP edge detectors should be directly chosen to construct linear edge features. To construct more complicated edge detectors based on raw pixels, the redundancy in the same direction should be removed and pixels with a limited distance are possibly useful to construct edge detectors. Therefore, it is suggested that the pixels in the GP edge detectors used to construct complicated filters should be chosen after using the proposed merging operation.

5.5.4 Pixels Selected by GP

Since the merging operation is employed at the function nodes, two different $s_{n,m}$ functions with the same direction in a subtree will be merged as one. In each function node, its children are considered as “terminals”, which is very different from the selection based on the real terminal (image I). In the merging operation, a “terminal” is not only I , but also a subtree containing $s_{n,m}$ is also considered as “terminal”. This is why Set_{nb1} and Set_{nb2} are different. The result for merging the Set_{nb1} is not equal to Set_{nb2} . Set_{nb1} is only dependent on the original terminal I , but Set_{nb2} is related to the function nodes and the original terminal I . After the merging operations, only a few pixels are selected and all these pixels are based on different directions or distances to the discriminated pixels. Therefore, the set of pixels is rich and has few redundancies. It is a potential reason that Set_{nb2} only including six pixels can construct very good edge filters. These second-order filters from Set_{nb2} are better than the estimated 5×5 window second-order filters, and can compete with the GP edge detectors.

A direct merging operation based on Set_{nb1} is not investigated because the set of pixels after the merging operation is approximated as a subset of

Set_{nb1} .

5.5.5 Computational Cost

The training time for evolving GP edge detectors is around 720 seconds on a single machine with CPU 3.1 GHz. The times for estimating the parameters of a linear filter in each run are around 30 seconds for Set_{nb1} and the 3×3 window, 19 seconds for Set_{nb2} , and 84 seconds for the 5×5 window. The times for estimating the parameters of a second-order filter in each run are around 450 seconds for Set_{nb1} and the 3×3 window, 204 seconds for Set_{nb2} , and 2230 seconds for the 5×5 window. When a 5×5 window second-order filter is optimised, the time for the parameter estimation is much longer than evolving a GP edge detector. Since the number of parameters in the estimated 5×5 window second-order filter is large, there is heavy computational cost for estimating a second-order filter using all pixels from the window. Therefore, effectively selecting pixels to construct filters can rapidly reduce computational cost in the training stage. The testing time for the GP edge detectors, and the estimated filters using pixels in Set_{nb1} and Set_{nb2} , are less than 0.05 seconds, which usually satisfies a real-time application, such as detections on videos (generally requiring the detection speed less than 0.1 second).

5.6 Chapter Summary

The goal of this chapter was to investigate pixels implicitly selected by GP. The selected neighbours for discriminating pixels as edge points or non-edge points were extracted from GP low-level edge detectors. Two suitable sets of pixels were used to construct low-level edge detectors.

From the analysis of the pixels existing in the evolved edge detectors, rich pixels were selected from these GP edge detectors and were used to construct linear and second-order filters. The comparisons of the test per-

formance on these filters, the Laplacian and Sobel edge detectors, and the 3×3 and 5×5 window filters, show that the set of pixels obtained from GP with high occurrences has a small size, and it is very effective to construct low-level edge detectors.

Two different strategies were used to search for a suitable set of pixels from the GP edge detectors to construct edge filters. When the pixels are directly searched from the terminal I based on the $s_{n,m}$ function, the set of pixels obviously contains diagonal pairs, which contains rich information for finding edges. When the redundancy from the same direction is removed and the pixels are selected by a merging operation based on $s_{n,m}$ function, the set of pixels has a smaller size necessarily and the pixels in the set around the discriminated pixel are not connected. The test performance on the second-order filters using these pixels shows the set of pixels is very rich and effective to construct edge detectors.

The objective function based on the F -measure method gives good solutions for the estimated filters, although it is not continuous for the estimated parameters of a filter. The continuous objective function based on correlation ratio has very good stability for the optimal filters, but low performance for optimising linear filters. This suggests that the objective function should choose the correlation ratio to optimise a complicated (non-linear) filter if the stability requirement is very strict, otherwise, the F -measure method should always be a good choice, compared with correlation ratio.

This chapter and Chapters 3 and 4 investigate how to evolve low-level edge detectors with very little prior domain knowledge from edge detection. When special domain knowledge, such as the Gaussian-based technique, is given, how to employ the Gaussian-based technique for developing edge detection will be addressed in the following chapter.

Part II

Gaussian-based Edge Feature Construction

Chapter 6

Gaussian-based Feature Construction

6.1 Introduction

In the previous three chapters, the search operators have been designed to automatically construct edge detectors when only training images and their ground truth are given, where very little prior knowledge is provided. Since Gaussian filters are widely used for edge detection [14, 140], Gaussian-based edge detection is considered as the prior domain knowledge in this chapter. How to effectively utilise this knowledge will be addressed in this chapter.

Different Gaussian-based approaches have been investigated based on a single Gaussian filter, such as Laplacian of Gaussian (LoG) [122] and Canny edge detectors [28], or multiple Gaussian filters, such as multi-scale edge detection [140] and surround suppression (SS) [66]. Techniques based on multiple Gaussian filters can improve detection accuracy [140].

However, there are problems in Gaussian-based approaches. Firstly, it is difficult to manually set the parameters (scales) of Gaussian filters. Gaussian filters at different scales are often combined to detect edges. In multi-scale edge detection [14, 170], the scales of Gaussian filters and the

window size affect the detection performance, but the literature does not address how to effectively tune parameters of Gaussian filters. The window size problem (blurring edges in a large window and noise influence in a small window) still exists in Gaussian-based edge detection. It is desirable to develop automatic techniques for setting parameters of Gaussian filters.

Secondly, it is not clear how to effectively combine Gaussian filters. When multi-scale Gaussian filters are combined in a fixed order, the detection performance is often dominated by the first processed Gaussian filter. SS combines different filters without any fixed order to improve detection accuracy [65, 66]. In SS, an operation, called *inhibition*, is used to suppress texture responses. In general, the response from a Difference of Gaussians (DoG) [14] is used in the inhibited term, and the response on the gradient of a Gaussian filter is the inhibited context. In SS, Gabor filters are usually used [66]. Since a two-dimensional Gabor filter is the product of a Gaussian kernel function and a sinusoidal function, and Gaussian filters can replace Gabor filters in SS, SS can still be considered as a kind of Gaussian-based edge detection. The main benefit of SS is to filter noise caused by textures [140]. From SS, effectively combining Gaussian filters can improve detection performance. However, it is difficult to reduce the influence on irregular textures by SS. Therefore, it is desirable to investigate new ways of effectively combining Gaussian filters.

6.1.1 Chapter Goal

The goal of this chapter is to investigate how to effectively use prior Gaussian-based knowledge in GP for edge detection. Some Gaussian-based filters are utilised by GP to construct Gaussian-based edge detectors for performance improvement. A function is proposed for combining different Gaussian filters. Since the Gaussian-based techniques are given as prior knowledge, this investigation mainly focuses on how to effectively and

efficiently evolve Gaussian-based edge detectors.

GP is a time-consuming algorithm to evolve programs when a large image dataset is used for training. In order to reduce the computational cost from a full-image dataset in the training phase, an initial investigation on the relationship between the detection performance and training images is conducted. Specifically, the following research objectives will be investigated.

- Whether GP can effectively evolve good Gaussian-based programs to extract edge features.
- Whether a small set of images can be used to train good Gaussian-based programs.
- Whether using different types of Gaussian filters is better than using a single type of Gaussian filter only to evolve Gaussian-based programs.

6.1.2 Chapter Organisation

In the remainder of this chapter, the second section proposes a GP system to construct Gaussian-based edge detectors. The third section gives the experiment settings. The fourth section presents the results of the experiments with discussions. The fifth section provides further discussions. The last section draws a summary of this chapter.

6.2 Gaussian-based GP System

The proposed Gaussian-based GP system is introduced in this section. In this GP system, the parameters of Gaussian filters and the operations between different Gaussian filters will be automatically generated.

6.2.1 Terminals Based on Gaussian Models

To rapidly find a Gaussian-based edge detector, the terminal set in the proposed GP system includes three types of Gaussian filters: the Gaussian gradient, LoG, and DoG. The Gaussian gradient filter $dg_\sigma(u, v)$ is shown in Equation (6.1), where σ is the scale parameter, and (u, v) is an offset (horizontal and vertical directions) from each discriminated pixel. The LoG filter $ddg_\sigma(u, v)$ is defined in Equation (6.2). The DoG filter $dog_\sigma(u, v)$ is shown in Equation (6.3).

$$\begin{aligned}
 g_\sigma(u, v) &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \\
 \frac{\partial g(u, v)}{\partial u} &= -\frac{u}{2\pi\sigma^4} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \\
 \frac{\partial g(u, v)}{\partial v} &= -\frac{v}{2\pi\sigma^4} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \\
 dg_\sigma(u, v) &= \sqrt{\left(\frac{\partial g(u, v)}{\partial u}\right)^2 + \left(\frac{\partial g(u, v)}{\partial v}\right)^2} \quad (6.1)
 \end{aligned}$$

$$ddg_\sigma(u, v) = \frac{u^2 + v^2 - 2\sigma^2}{2\pi\sigma^6} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \quad (6.2)$$

$$dog_\sigma(u, v) = g_\sigma(u, v) - g_{2\sigma}(u, v) \quad (6.3)$$

The terminal set also includes random constants *rnd* (real numbers) in the range from -10 to 10 based on initial experiments. In this terminal set, the scales σ (real numbers) of all Gaussian filters are randomly generated in the continuous range from 1 to 5 . Let the large scale in the DoG be *double* the small one, so the scale range of all Gaussian filters is from 1 to 10 . Therefore, the coarsest scale (from 2 to 10) covers the range from 3 to 6 as suggested in [16] so that it is possible to find more Gaussian filters.

6.2.2 Function Set

In the multi-scale edge detection technique, it is quite complex to use a function to describe the operation for detecting edges from a coarse solu-

tion to a fine solution or from a fine solution to a coarse solution. However, SS uses an operation to easily present a combination of different Gaussian filters to detect edges [66]. Therefore, the operation is used here to construct new Gaussian-based edge detectors.

Since Gaussian filters in the terminal set can be considered as edge detectors, a simple function set, namely $\{+, -, *, \div, \mathbb{C}\}$ is chosen. Here, $\div$ is protected division, producing a result of 1 for a 0 divisor; and $\mathbb{C}$ is a combination function from SS, which takes two arguments. Let $f_1(x, y)$ and $f_2(x, y)$ be image intensities or outputs from subtrees for a pixel with position (x, y) , $f_1(x, y)\mathbb{C}f_2(x, y)$ represents the operation of the combination function. There are two steps in $f_1(x, y)\mathbb{C}f_2(x, y)$. In the first step, for each pixel, the first argument $f_1(x, y)$ provides the neighbours of the pixel with position (x, y) in a local 7×7 window. The second argument $f_2(x, y)$ provides the values of the relative neighbours (in the 7×7 window) transformed by Equations (6.4) and (6.5). Here, u and v (u' and v') are horizontal and vertical offsets, and $\sum_{u', v'}$ is the sum of $positiveN(u', v', f(x, y))$ based on the 7×7 window. Note that $normN(u, v, f(x, y))$ will be 0 if $\sum_{u', v'}$ is equal to 0. In the second step, convolution of the values (7×7 window) from $f_1(x, y)$ and $f_2(x, y)$ is performed to return a value for $f_1(x, y)\mathbb{C}f_2(x, y)$.

$$positiveN(u, v, f(x, y)) = \max\{f(x + u, v + y), 0\} \quad (6.4)$$

$$normN(u, v, f(x, y)) = \frac{positiveN(u, v, f(x, y))}{\sum_{u', v'} positiveN(u', v', f(x, y))} \quad (6.5)$$

An existing surround suppression technique [66] can be expressed by the Gaussian-based GP system as GG_{SS} in Equation (6.6), where σ_1 and σ_2 are scaling parameters, dg_{σ_1} and dog_{σ_2} are the results after applying Gaussian filters dg_{σ_1} and dog_{σ_2} to an image, and A and B are constants. For a pixel, $dg_{\sigma_1}\mathbb{C}dog_{\sigma_2}$ returns the convolution of dg_{σ_1} with $normN(u, v, dog_{\sigma_2})$ in the 7×7 window. If a pixel is located in a flat area (such as within which the responses dog_{σ_2} of its neighbours are 0), the operation $dg_{\sigma_1}\mathbb{C}dog_{\sigma_2}$ will return 0 and GG_{SS} only depends on dg_{σ_1} and B . For a texture pixel with

a non-zero response, its neighbours have similar responses or edge responses. In the 7×7 window, the number of edge points typically is small, the influence from the edge responses is not obvious. The return value of $dg_{\sigma_1} \mathbb{C}dog_{\sigma_2}$ is almost the same as the dg_{σ_1} value of the pixel. Therefore, GG_{SS} will suppress the responses from textures. When a pixel is on a true edge and its neighbours have texture responses, most of the neighbours normally have weaker responses dg_{σ_1} than the pixel. Therefore, the return value of $dg_{\sigma_1} \mathbb{C}dog_{\sigma_2}$ is obviously lower than the response dg_{σ_1} . GG_{SS} has slightly lower response contrast than dg_{σ_1} .

$$GG_{SS} = dg_{\sigma_1} - A * dg_{\sigma_1} \mathbb{C}dog_{\sigma_2} - B \quad (6.6)$$

6.2.3 Fitness Function

The F -measure is a time consuming evaluation system if a limited offset distance for detected pixels to true edge points is allowed. This is because one-to-one assignment operation (a predicted true edge point is only allowed to match a single true edge point) is required [123]. Fitness functions based on FOM were investigated in Section 4.2 (on page 105). FOM as a fitness function can evolve low-level edge detectors, and $f_{FOM_{avg}}$ (see Equation (4.7) on page 108) has similar detection performance to F -measure. Therefore, in order to allow an offset in training results, $f_{FOM_{avg}}$ is utilised as the fitness function in this Gaussian-based GP system.

6.3 Experiment Settings

The BSD image dataset is employed for training Gaussian-based edge detectors. Any Gaussian filter in the terminal set can be used to detect edges, therefore a new GP edge detector including Gaussian filters is expected to have (better) ability to perform edge detection. Since the fixed threshold 0 is used for all GP edge detectors, several images are employed to check whether a GP edge detector (with threshold 0) can effectively detect true

edge points. If one image is poorly detected, the GP edge detector usually poorly performs detection on other images. It is possible that a large number of images as training data is not necessarily required. When a small set of images is chosen as training data, how many images are required and how to choose images as training data are investigated as follows.

6.3.1 Training Data

To investigate the minimum number of training images required by GP, different sets of images are used as the training data. The first setting of the training data is to only employ one full image. Each of the six images in Figure 6.1 is selected as the training data respectively. These six images are selected based on different edge information, such as the single object in image 42078, irregular textures in image 106020 and many edges in image 23080. In terms of their evolved edge detectors' performance F , the six images will be divided into three levels, namely the worst, middle and best levels.

Secondly, in order to improve performance of the edge detectors evolved by a single training image, two images from the six images are selected to combine as the training data. It is possible that the training images in the best level are sufficient to train edge detectors, and the performance improvement mainly focuses on the edge detectors evolved from the worst and middle levels. The combinations of two images mainly come from the worst and middle levels. There are 15 combinations in total, but only four combinations are chosen. The four combinations are a combination of two images from the worst level, two combinations of one image from the worst level and one image from the middle level, a combination of one image from the middle level and one image from the best level. Chapter 3 used S_{20} to train good low-level edge detectors, so it is considered that S_{20} is good enough to train GP Gaussian-based edge detectors. In Figure 6.1, all images, except for image 23025, are included in the 20 images from

Figure 6.1: Six example training images from the BSD dataset and their ground truth.

S_{20} . The reason to select image 23025 is that different edge information, such as the boundary between the girl and the background with different graylevels, exists in the image. Image 23025 is used to compare with S_{20} for training GP edge detectors.

Chapter 4 (on page 119) sampled 5 subimages (of size 51×51 pixels) from each original image (of size 481×321 pixels) in S_{20} as the training data. To use the original images (not their subimages) as training images and reduce the computational cost during the training stage, two different sampling techniques based on original images as the training data are pro-

posed here. The first technique is to randomly select one image as the evaluation data at each generation; and the second technique is to randomly select a set of images as the training data at each generation. In the second technique, at the beginning, the number of the training images is a small number M_{min} , and after every g_{period} generations, the number is increased by 1. When the number of images is equal to M_{max} , the number of training images will return to a fixed value $M_{restart}$. It is possible that M_{min} training images can distinguish the detection performance of initialised edge detectors. After some generations, these training images may not be good enough to distinguish edge detectors' performance, so more images are required for evaluating evolved edge detectors. When M_{max} training images are used, some evolved edge detectors might be obviously better than the others in the population. Again, M_{min} training images could distinguish the detection performance of these edge detectors.

Note that S_{rnd} now will indicate the first sampling technique based on the 20 full images S_{20} , S_{ada} now will indicate the second sampling technique based on S_{20} , and S_{sub} now will indicate the subimages sampling technique in Chapter 4. S_{20} represents the whole 20 images as the training data.

6.3.2 Parameter Settings

The parameter values for GP are: population size 500; maximum generations 200; maximum depth (of a program) 7; and probabilities for mutation 0.15, crossover 0.80 and elitism (reproduction) 0.05. For the sampling technique S_{ada} , $M_{restart} = M_{min} = 4$, $M_{max} = 8$, and $g_{period} = 10$ are used. These values are chosen based on common settings and initial experiments. There are 30 independent runs for each experiment.

The test performance evaluation is directly based on the binary outputs of GP edge detectors using the fixed threshold 0, without non-maximum suppression post-processing. All GP edge detectors are tested on the *same*

100 BSD test images. To measure the performance of GP edge detectors, the F -measure is used in the testing phase. The test performance is evaluated by the system used in [123]. In the evaluation system [123], a simple thinning operator [106] is employed on the predicted results, then recall and precision are calculated after finishing one-to-one assignment [32, 60]. The test performance F ($f_{0.5}$, see Equation 4.17) is calculated over all pixels of the 100 BSD test images. Here, an offset from a predicted edge point to a true edge point (within 3 pixel distance) is allowed in the test evaluation system.

6.4 Results and Discussions

The test results are separated into three parts: single images, two images, and a set of images. Since there are multiple comparisons among different small training datasets, a multiple comparison based on one-way ANOVA is employed to compare multiple results in this chapter, and Holm's method [75] is used for p -value adjustment with overall significance level of 0.05.

6.4.1 A Single Image as Training Data

Table 6.1 shows the means and standard deviations of F , and means of recall and precision of GP Gaussian-based edge detectors evolved by each single image from the six images. As can be seen, the evolved detectors from images 42078 and 106020 have low performance on the 100 BSD test images, however the evolved detectors from images 23025 and 23080 present good detection performance (compared with the Sobel edge detector with $F = 0.48$ [123]). An interesting observation is that the low performing edge detectors have low recall. The average recall of the evolved edge detectors from image 42078 or 106020 is lower than 0.5. Most of the edge detectors from the six images have precision around 0.5.

Table 6.1: Test performance F values and means of recall and precision of GP Gaussian-based edge detectors from each of the six images as training data respectively.

Image	F (Mean $\pm$ Standard Deviation)	Recall	Precision
42078	0.4072 ± 0.0500	0.3213	0.5741
106020	0.4353 ± 0.0342	0.4634	0.4158
23025	0.5402 ± 0.0077	0.5851	0.5029
68077	0.5006 ± 0.0220	0.5431	0.4812
23080	0.5267 ± 0.0128	0.5913	0.4767
216053	0.5146 ± 0.0139	0.5130	0.5205

Table 6.2: One-way ANOVA (row vs column) for GP Gaussian-based edge detectors from the six images as training data respectively.

	106020	23025	68077	23080	216053
42078	↓	↓	↓	↓	↓
106020		↓	↓	↓	↓
23025			↑	—	↑
68077				↓	—
23080					↑

Table 6.2 gives a statistical comparison between all edge detectors evolved by the six images respectively. Note that all comparisons are based on F values in this chapter. For a pairwise comparison, the first group is from a setting in the first column, and the second group is from a setting in the first row. Here, $\uparrow$ indicates that the first group is significantly better than the second group; and $\downarrow$ indicates that the first group is significantly worse than the second group; otherwise, $—$ indicates no significant difference between the first and second groups. Since there are no direction influences

on the comparison of two groups, some redundant comparison results are removed from the table. From the comparison results, the six groups of results can be divided into three sets. The first set is the worst performing edge detectors, namely the edge detectors from images 42078 and 106020 which are significantly worse than the edge detectors from the other four images. The second set is the middle level for the edge detectors from images 68077 and 216053. These edge detectors are significantly better than the worst edge detectors from the images 42078 and 106020, but significantly worse than the edge detectors from images 23025 and 23080. The third set is the best level for these edge detectors from images 23025 and 23080. In each level, there are no significant differences between the edge detectors from different images, except for the worst level.

Images 42078 and 10602 only contain a single object, and the contrast between the object and the background is high. The edge information in the two images is not rich. For images 68077 and 216043, the edge information from the two images are richer than the edge information from images 42078 and 106028. The two images in the middle level contain different objects and different graylevel gaps between different regions. Images 23025 and 23080 contain the richest edge information, compared with the other four images. In image 23080, there is rich edge information from the building and the person. In image 23025, the edges at the boundary of the person are different, such as the person's boundary between the body and the wall and between the leg and the step, and the boundary of the basket. It seems that images including different edges are good to evolve GP Gaussian-based edge detectors.

6.4.2 Two Images as Training Data

From Table 6.1, it is known that image 42078 obtains the lowest performance of the evolved edge detectors, therefore, image 42078 is mainly used to combine with other images as training data. Four new training

Table 6.3: Test performance F values and means of recall and precision of GP edge detectors from two images as training data respectively.

Pair	F (Mean $\pm$ Standard Deviation)	Recall	Precision
C1	0.5349 ± 0.0213	0.6112	0.4869
C2	0.5328 ± 0.0117	0.6087	0.4753
C3	0.5028 ± 0.0181	0.4651	0.5512
C4	0.5011 ± 0.0205	0.4622	0.5540

datasets are used, and they are training data C1 including images 42078 and 68077, C2 including images 23080 and 216053, C3 including images 42078 and 216053, and C4 including images 42078 and 106020.

Table 6.3 gives the means of F , recall and precision values of the GP Gaussian-based edge detectors evolved by training datasets C1, C2, C3 and C4. First of all, from the table, the means of F values of the evolved edge detectors from all training data with two images are higher than 0.5. Although images 42078 and 106020 get the worst performance when they are individually used to train edge detectors, the combination of the two images C4 obtains good performance. Secondly, two images as the training data can obtain high recall, such as C1 and C2. However, the evolved edge detectors from C3 and C4 have low recall, but high precision.

Table 6.4 gives a comparison between each training data using two images and each training data using a single image. Firstly, from the comparisons among all results from all single image training data and each training data with two images, the evolved edge detectors from C4 are significantly better than the evolved edge detectors from images 42078 and 106020 when they are individually used to train edge detectors respectively. Secondly, the training data using an image from the worst level (of a single image as the training data) and an image from the middle level, namely C1 and C3, obtain results which are significantly better than the

Table 6.4: One-way ANOVA (row vs column) for GP edge detectors from two images and single image as training data.

	42078	106020	23025	68077	23080	216053
C1	↑	↑	—	↑	—	—
C2	↑	↑	—	↑	—	—
C3	↑	↑	↓	—	↓	—
C4	↑	↑	↓	—	↓	—
S_{rnd}	↑	↑	—	↑	—	—

results from the worst level, but not significantly different from the results from the middle level. Thirdly, the training data using an image from the best level and an image from the middle level, namely C2, obtains the results which are significantly better than the results from the worst level and the middle level. Fourthly, the evolved edge detectors from all four training datasets using two images are significantly better than the two worst level images as training data respectively. From these comparisons, it seems that the combination of two images as training data can improve the performance of evolved detectors. When an image is randomly selected as training data, the evolved edge detectors might have low performance. If two images are randomly chosen as training data, the evolved edge detectors at least do not have too low performance, compared with the evolved edge detectors from a single image as training data in the worst level.

Additionally, each combination of two images as the training data, except for C3, has significantly better results than both images from each set as training data respectively. C3 consists of one image from the worst level and one image from the middle level, but the evolved edge detectors from C3 are not significantly different from the evolved edge detectors from the image in the middle level. For a single image as training data,

Table 6.5: One-way ANOVA (row vs column) for GP edge detectors from two images as training data respectively.

	C2	C3	C4
C1	–	↑	↑
C2		↑	↑
C3			–

the comparison between S_{rnd} and a single fixed image will be discussed in Section 6.4.3.

Table 6.5 shows the comparisons among the different combinations based on two images. From the table, it reveals that the evolved edge detectors from the combinations C1 and C2 are significantly better than the evolved edge detectors from the combinations C3 and C4. There are no significant differences between C1 and C2, and between C3 and C4. The combinations with one image from the worst level and one image from the middle level might have the similar performance to the combination with both images from the middle level, and might be the same as the combination with both images from the worst level as well.

Comparing the details of the four combinations of two images from the six images in Figure 6.1, some interesting observations on edge characteristics in these images are found. Firstly, in C1, although image 42078 only has a very obvious object, the graylevels of the boat and hill in image 68077 are close to the graylevels of background. Secondly, in C3, most of the graylevels of the building and person can be clearly distinguished from the background, which is similar to the difference between the object and background in image 42078. Lastly, high contrast exists in two images 42078 and 106020 (C4), but the edges at two different regions in image 106020 are totally different from the edges existing in image 42078, which enriches edge information in C4. From the different combinations of two

Table 6.6: F values and means of recall and precision of GP edge detectors from sampling techniques on 20 images.

Training	F (Mean $\pm$ Standard Deviation)	Recall	Precision
S_{sub}	0.4940 ± 0.0260	0.8403	0.3517
S_{rnd}	0.5279 ± 0.0383	0.6989	0.4485
S_{ada}	0.5521 ± 0.0290	0.6716	0.4869
S_{20}	0.5628 ± 0.0131	0.6681	0.4893

images, it seems that a combination with different edge information can improve the evolved edge detectors' performances.

6.4.3 Sampling

Table 6.6 gives the test results from the different sampling techniques and the training data using all 20 images. Comparing with the training data using a single image and using two images, it is found that recalls of the evolved edge detectors from different sampling techniques or 20 images are very high. Although S_{sub} has the highest recall average in the table, its precision average is the lowest. The lowest average of F values for evolved edge detectors is from S_{sub} . The highest average of F values for evolved edge detectors is from S_{20} , and the mean of F values of the edge detectors from S_{ada} is close to the mean of F values of the edge detectors from S_{20} .

Table 6.7 presents the comparison results among S_{sub} , S_{rnd} , S_{ada} and S_{20} . The evolved edge detectors from the sampling technique using subimages S_{sub} are significantly worse than the sampling techniques using original full images.

Although the evolved edge detectors from S_{rnd} are significantly worse than the evolved edge detectors from S_{ada} and S_{20} , the difference between the mean of F values from the sampling technique S_{rnd} and that of S_{20}

Table 6.7: One-way ANOVA (row vs column) for GP Gaussian-based edge detectors from sampling techniques on 20 images.

	S_{rnd}	S_{ada}	S_{20}
S_{sub}	↓	↓	↓
S_{rnd}		↓	↓
S_{ada}			—

is less than 0.04. Therefore, even though only one image is randomly selected as evaluation data at each generation, the varying training data can still lead to good edge detectors, compared with the training data using all 20 images. Also, from Table 6.4, the evolved edge detectors from S_{rnd} are significantly better than the evolved edge detectors from images 42078, 106020 and 68077, but not significantly different from the evolved edge detectors from the other three images in the table. It shows that a random image at a generation as the evaluation data is generally better than a fixed image in the whole evolution process as the evaluation data.

Compared with S_{sub} and S_{rnd} , S_{ada} obtains evolved edge detectors which are significantly better. The increasing number of images selected at each generation possibly makes GP select good Gaussian-based edge detector candidates at each generation. Since there are no significant differences between the evolved edge detectors from S_{ada} and S_{20} , the sampling technique with varying images in a set is efficient to train edge detectors which have similar performance to the edge detectors evolved by all images in the set. Since the number of training images is from four to eight, the average number of training images in all generations is six. From the test performance for the edge detectors evolved by S_{ada} , it seems that only using several training images from a set can train edge detectors performing similarly to the edge detectors evolved by all images in the set. This sampling technique considerably reduces the computational cost over using

the 20 images, while maintaining a similar performance.

6.4.4 GP Edge Detectors vs Existing Gaussian-based Edge Detectors

In order to validate the performance of GP Gaussian-based edge detectors, the Gaussian gradient (GG) and the surround suppression (SS expressed by GG_{SS}) are selected as existing Gaussian-based edge detectors to compare with the evolved edge detectors. There is only the feature extraction stage in GG and SS, so the non-maximum suppression and hysteresis threshold techniques are not used. From the outputs of GG and SS, different thresholds are used to choose the maximum F on the BSD test images as test performance for GG and SS, and F_{max} is employed for their test performance. Based on 52 different thresholds $\frac{k}{52}$ ($k = 0, 1, \dots, 51$), it is found that the F_{max} for GG is 0.5153, and SS is 0.5381. Note that the edge responses from GG and SS are mapped to the range from 0 to 1. Here $\sigma \in \{1.0, 2.0, 5.0\}$ is used to choose the best F_{max} for GG, and for the suppression based on the DoG in SS, the best test performance is chosen from the three sets of the σ pairs (1.0, 2.0), (1.0, 3.0) and (0.8, 3.0). Note that $\sigma = 1.0$ is a very common setting [14], and the other values are used for different detection results. The evolved Gaussian-based edge detectors are rotation invariant, so only the invariant version of surround suppression is chosen, and different directions are not considered.

Table 6.8 shows the comparisons among GG, SS, image 23025, C1, S_{ada} and S_{20} . Here, the highest mean of F values is chosen from training data using a single image, two images, one of the three sampling techniques (S_{sub} , S_{rnd} and S_{ada}), and the 20 images. From the table, only a single Gaussian filter (GG) has the lowest performance in these compared results. In general, the evolved Gaussian-based edge detectors include several Gaussian filters, and SS contains three Gaussian filters. The evolved edge detectors from image 23025 and C1 have no significant differences from SS,

Table 6.8: One-way ANOVA (row vs column) comparing F values from some GP edge detectors with F_{max} values from Gaussian gradients (GG) and surround suppression (SS) on the BSD test images.

	SS	23025	C1	S_{ada}	S_{20}
GG	↓	↓	↓	↓	↓
SS		—	—	↓	↓
23025			—	—	↓
C1			—	↓	↓
S_{ada}					—

but the evolved edge detectors from S_{ada} and S_{20} are significantly better than SS.

In general, only using a single image as the training data is not sufficient to train edge detectors. However, the evolved edge detectors from the single training image 23025 are not worse than the other results in Table 6.8, except for the results from S_{20} . It seems that it is possible to use a single image as training data to evolve good edge detectors as long as it includes rich enough edge information.

The small set C1 is included in S_{20} , and the evolved edge detectors are significantly better than GG and SS, but these edge detectors from C1 are significantly worse than the evolved edge detectors from S_{ada} and S_{20} . It seems that two images as training data can be used to evolve good edge detectors. Also, using a set of images including the two images is better than using the two images to train Gaussian-based edge detectors, even when the evaluation data at each generation does not use all images in the set (the number of selected images is more than 3).

Figure 6.2: Five example detected BSD test images by Gaussian gradient (GG) and surround suppression (SS).

6.4.5 Detected Images from the Best GP Edge Detectors

Figure 6.2 shows five example images from the BSD test image dataset and their ground truth (GT). Also, the detected results from GG and SS are shown in Figure 6.2. The results detected by GG and SS are based on their own best threshold in the overall view of the BSD 100 test images. The best threshold means that F is maximum when the threshold from the set of thresholds ($\frac{k}{52}$) is used. The overall best thresholds for GG and SS are 0.2793 and 0.0392, respectively. These selected images in Figure 6.2 have different textures, and their edges are not easily detected by a normal Gaussian filter, except for image 42049, when the task is very easy.

To visually compare the detected images, Figure 6.3 shows these four images detected by the best evolved edge detector from image 23025 ($F =$

Figure 6.3: Five example BSD test images detected by the GP Gaussian-based edge detectors.

0.5531), C1 ($F = 0.5699$), S_{sub} ($F = 0.5607$), S_{rnd} ($F = 0.5750$), S_{ada} ($F = 0.5998$) and S_{20} ($F = 0.5879$).

Firstly, from the detected results for image 42049, all edge detectors give a good detection result. Only noise from the branch slightly affects the Gaussian-based edge detectors from image 23025, S_{sub} , S_{ada} and S_{20} to perform detection. From the detected results on the 100 BSD test images, it

seems that evolved Gaussian-based edge detectors can perform detection well on simple (non-texture) images.

Secondly, the detected results on the other four different images with textures show the different performance on these edge detectors. For the four images, GG is strongly affected by the different textures in these images. SS can reduce the influence from textures, but is still affected by some textures, such as the irregular background in image 175032 (sticks being out of order) and the heavy texture on the wall in image 385039. Also, SS fails to detect edges in low contrast, such as the body of the object in image 69020.

From the detected results by the evolved Gaussian-based edge detectors, the best edge detector from S_{sub} detects most of the edges, but it is strongly affected by textures, compared with the other evolved edge detectors. The best evolved edge detector from image 23025 detects images 69020 and 385039 with very slight influence from textures, but it is still hard to detect the top boundary of the object in image 69020. For image 175032, it is strongly affected by the irregular background, compared the detection from C1, S_{rnd} , S_{ada} and S_{20} . The best evolved edge detector from C1 has similar detection for image 69020 to the edge detector from image 23025, but it has stronger influence from the wall in image 386039 and weaker influence from the irregular background in image 175032, compared with the latter. The best edge detector from S_{ada} is not affected by the animal skin texture in image 87046, whereas the other edge detectors are heavily affected by it. Note that some of the evolved Gaussian-based edge detectors have clear responses on the boundaries among the stones in the background, but are not as affected by the skin texture as GG and SS. Also, the edge detector from S_{ada} has good detection results for the other images. Based on the BSD test images, the edge detector from S_{ada} is the best edge detector among all evolved Gaussian-based edge detectors in Figure 6.3. From the visually detected images, the GP Gaussian-based edge detectors have good ability to suppress textures.

Figure 6.4: One example GP Gaussian-based edge detector.

6.4.6 Example Evolved GP Gaussian-based Edge Detector

In order to directly present a GP Gaussian-based edge detector, an evolved program is selected from using the image 23025, which is shown in Figure 6.4, where “C” is combination function $\mathbb{C}$. Equation (6.7) describes this GP Gaussian-based edge detector GGP .

$$GGP = dg_{1.53} * dg_{4.55} - ddg_{1.97} * ddg_{4.08} - 9.90 - \frac{2ddg_{1.06} \mathbb{C} ddg_{1.75} \mathbb{C} \frac{ddg_{4.08} + dg_{2.90}}{1.86ddg_{4.09}} - \frac{\frac{ddg_{1.90} + ddg_{4.08}}{dg_{4.47}}}{dg_{4.47}}}{5.87 + dg_{2.90} - ddg_{1.45} \mathbb{C} ddg_{1.97}} \quad (6.7)$$

From this equation, it can be seen that the evolved solution includes different combinations of Gaussian filters, namely multiplication, division, convolution and difference of different combinations. From the parameter σ , GGP includes both fine and coarse solutions for detecting images. Also, this solution includes the three types of Gaussian filters from the terminal set. It seems that the combination of using the three types of Gaussian filters is good to detect edges, rather than using only one alone. Note that $\mathbb{C}$ can be used to suppress texture gradients. For example, “ $dg_{2.90} - ddg_{1.45} \mathbb{C} ddg_{1.97}$ ” is similar to the suppression technique. The division might be potentially used to suppress noise, and the difference of different combinations and the multiplication possibly enhance the

response from the true edges. However, the edge detector is very complicated; further experiments and analyses are required in the future.

6.5 Further Discussions

This section discusses the computational cost, the training images, thickness of detected edges existing in GP Gaussian-based edge detectors, and contributions of Gaussian filters, and the combination function $\mathbb{C}$.

6.5.1 Computational Cost

The time for a GP run evolving a Gaussian-based edge detector with 200 generations is around 4.5 days on a single machine with CPU 3.1GHz when S_{20} (20 images) is used as the training data. So this is a time-consuming algorithm, but this can be sped up by using a computational grid. There are two reasons for the heavy computational cost. Firstly, the maximum tree depth for the GP Gaussian-based programs is seven, so it is possible that a GP program includes more than ten Gaussian filters. In the training stage, the weights of a Gaussian filter in a new program need to be calculated. The time for calculating responses on all Gaussian filters in a program will take most of the execution time. Secondly, the training data has 3088020 ($20 \times 481 \times 321$) pixels, and a GP program needs to take some time to discriminate all pixels.

However, the training time for only using a single image as the training data is around five and half hours in a run. When the training data uses two images, the training time for each run is about 11 hours. For the sampling techniques, the training time for using S_{rnd} as the training data is very close to the training time for using a single image as the training data. It costs about 33 hours to use S_{ada} to evolve a GP Gaussian-based edge detector. The sampling technique S_{ada} reduces more than two thirds of the training time cost in S_{20} . Therefore, efficiently selecting images can

remarkably reduce the computational cost in the GP system, while maintaining good performance.

All GP evolved edge detectors only take less than half a second to detect a BSD image (in the test stage). The depth of the solution in Figure 6.4 is seven (the maximum depth), and the solution includes 17 Gaussian filters and three function operators $\mathbb{C}$, which makes the solution detect one BSD image in slightly longer than 0.1 second. However, simplification of the evolved detector can make this detection time even shorter. From Equation (6.7), the Gaussian filters $ddg_{1.06}$ and $dg_{4.47}$ can be only calculated once, not twice in the tree-based program in Figure 6.4. It is possible to replace $ddg_{1.97}$, $ddg_{1.75}$ and $ddg_{1.45}$ with an approximate LoG. Also, $ddg_{4.08}$ and $ddg_{4.09}$ might be replaced by another approximate LoG. How to automatically and efficiently simplify evolved GP Gaussian-based edge detectors would be investigated in the future. In addition, a restriction on the number of Gaussian filters in a GP edge detector needs to be investigated so that an evolved GP edge detector can perform edge detection less than 0.1 second.

6.5.2 Training Images

In general, training data only using a single image is not sufficient to select good edge detector candidates. However, most GP Gaussian-based edge detectors have distinguished detection performance on some of single images [14, 140]. When a new edge detector is generated from the special terminal set and function set, the detector is expected to have some ability to detect edges if it contains Gaussian filters from the terminal set, since all Gaussian filters in the terminal set have some ability to detect edges. When the terminals of a detector do not include any Gaussian filters, namely its terminals only include random constants rnd , its output will be a constant. This detector fails to detect edges and its fitness is so low that it could not survive in the next generation. Therefore, almost all individuals at each

generation have some ability to detect edges.

However, a GP edge detector including Gaussian filters might not have good detection on the test images. A reason is that only the fixed threshold 0 is used for all GP Gaussian-based edge detectors here. Note that the detected results from a GP edge detector are binary. If the observations of a GP program are larger than a designed threshold (0), the relevant pixels are discriminated as edge points. It is possible that the threshold 0 is not good for a new GP edge detector to discriminate pixels, and the optimal threshold (according to F_{max}) for the observations is not close to 0. A suitable threshold is required for the observations. Here, a suitable threshold means that the performance on the binary edge maps based on the threshold is close to the performance on the best detected results of the observations.

Figure 6.5 shows the detection performance on image 42078, image 106020, and C4 from GG. Here, the horizontal axis indicates thresholds, and the vertical axis indicates the performance on F . To indicate the range of F , “d1” is the position for the performance with $F_{max} - 0.1$ (left), and “d2” for $F_{max} - 0.1$ (right). Considering the standard deviation σ_F of F values, “f1” is the position for $F_{max} - \sigma_F$ (left), and “f2” for $F_{max} - \sigma_F$ (right). For image 42078, if accepting the difference between F and F_{max} less than 0.1, a suitable set of thresholds is from “d1” to “d2” according of its F curve. If a suitable set of thresholds is considered as the relevant F values larger than a fixed constant, such as 0.3, the range of the suitable thresholds for image 42078 is larger than the range of the suitable thresholds for image 106020. However, some thresholds with F larger than 0.3 in image 42078 make F for image 106020 very low, such as the threshold 0.6. Therefore, a threshold is suitable for one detected image, but might be not suitable for another detected image. The combination of images 42078 and 106020 (C4) reduces the range of the thresholds with F larger than 0.3, and the suitable range is accepted for each image (with F larger than 0.3).

Figure 6.5: Detection performance on images 42078 and 106020, and C4 from Gaussian gradients (GG).

The space $Th_i = \{k | F_{i,k} > F_{constant}\}$ can be employed to describe a suitable set of thresholds with F larger than a constant $F_{constant}$ on detected result i , where, k is a threshold, and $F_{i,k}$ is F on detected result i when threshold k is used. The threshold k directly works on the observations. For the combination C_{ij} of two detected results i and j , the space becomes $Th_{C_{ij}} = \{k | F_{C_{ij},k} > F_{constant}\}$. In general, $Th_{C_{ij}}$ is similar to the intersection $\{k | F_{i,k} > F_{constant}, F_{j,k} > F_{constant}\}$ (becoming narrower after the combination), but $Th_{C_{ij}}$ is not totally equal to their intersection, in terms of the space.

From the GP system, the fitness of a GP candidate (edge detector) is mainly determined by the candidate's structure and its threshold used to make edge points. If Th_i of the candidate does not include the fixed threshold 0, its fitness is low even though it has good binary detection results from another threshold (not equal to 0). Therefore, Th_i of a candidate is much more important than the candidate's structure in the GP system. The GP system firstly might choose the candidates whose Th_i includes 0.

When an image is used as the training data, Th_i of a candidate is expected to include 0. However, the position of 0 in Th_i can be different.

When the training images only have clear objects (with high contrast), the range of Th_i of the candidate for detecting the training image is usually large. The threshold 0 of the candidate easily locates at one end of Th_i on the training image so that the threshold 0 is not in the range of Th_i on the test images. However, when the range of Th_i of the candidate is narrow for a training image, Th_i on the test images includes 0 with high probabilities. This is a reason that the images with high contrast objects are not good to evolve GP edge detectors. Since $Th_{C_{ij}}$ of a candidate is normally similar to the intersection $\{k | F_{i,k} > F_{constant}, F_{j,k} > F_{constant}\}$, the combination of two images as the training data makes 0 locate in a narrow $Th_{C_{ij}}$, which makes the candidate more accurate on detecting test images.

For the random sampling of one image from a set of images as the training data, at each generation, the training image might be different, and the intersection of suitable thresholds at least is narrower than the range of the suitable set of thresholds from one image in the set of images. Therefore, a good candidate should have a suitable set of thresholds for different images, and should not poorly perform detection for all images. In a generation, a poor detection result of the current image will make a candidate with good ability to detect other images be removed from the current population. That is why randomly sampling one image as the training data has the potential to find good Gaussian-based edge detectors, but is not very good from the overall view.

When varying images from a set of images are used as training data, a candidate with poor detection performance on one image, but good detection performance on the others, can still survive in the current population. Therefore, the evaluation system considers the overall performance. Since several images are randomly chosen to do the evaluation at each generation, the suitable set of thresholds $Th_{S_{ada}} = \{k | F_{S_{ada},k} > F_{constant}\}$ of a GP edge detector is similar to the intersection $\bigcap_i Th_i (i \in S_{ada})$. The range of $Th_{S_{ada}}$ becomes narrow, and it is approximately close to the set of $Th_{S_{20}}$ for all images.

From Th_i of an edge detector, a hypothesis is that a small set of images sampled from the total M images can indicate the performance of the edge detector. However, how to design formulae based on Th_i to select images is not clear, and needs to be further investigated. If we can establish a relationship between Th_i on each image and the whole dataset, this relationship might help predict the difference from the trained edge detectors with one image and the whole dataset. Also, it helps us to further understand edge detection from the edge response view. The further investigation of Th_i will be done in the future work.

6.5.3 Thickness of Detected Edges

The edges detected by the GP edge detectors (see Figure 6.2) often have thick responses on edges. A potential reason is that large scale Gaussian filters exist in the GP edge detectors. For instance, Equation (6.7) includes $ddg_{4.08}$, $ddg_{4.09}$ and $dg_{4.47}$. Another potential reason is that the ground truth images in the training data (see Figure 6.1) are combined with different observations, so the width of a true edge is possibly more than one pixel. The overlap in prediction for a true edge point is allowed to have more than one response in the fitness function $f_{FOM_{avg}}$. Figure 6.6 shows two example detected images from S_{ada} in Figure 6.2 simply thinned by a thinning operation [106]. Non-maximum suppression [28] could easily be used to thin the detection results, which is not the focus of this thesis.

6.5.4 Contributions on Different Gaussian Filters

Influence of using different types of Gaussian filters is now investigated. In order to check whether using different types of Gaussian filters is better than using a single type of Gaussian filter only, the terminal set now only chooses a single type of filter to evolve programs. The combination function $\mathbb{C}$ might influence the performance of evolved edge detectors, so $\mathbb{C}$ is removed from the function set. The influence of $\mathbb{C}$ will be discussed in the

Figure 6.6: Two example images detected by a Gaussian-based edge detector before and after thinning operations [106].

next subsection. Since image 23025 can be used to obtain good evolved edge detectors, the single image is employed as the training data for analysis of using different types of filters to evolve edge detectors.

Table 6.9 shows the results from only using dg_σ , dog_σ and ddg_σ respectively. Here, the best evolved edge detector (Max) and the worst evolved edge detector (Min) for using each single type of filter are given. The p -values are obtained by the two-sample t -tests between the relevant results and the results using all functions when only image 23025 is used as the training data.

From the tests (p -values), only using a single type of filter is not sufficient to evolve good edge detectors. All of the results from using a single type of filter obviously decrease the test performance, and they are significantly worse than using the three types of filters. Therefore, only combining the same type of Gaussian filter is not as good as combining different types of Gaussian filters for extracting edge features, which is similar to the surround suppression technique using different types of filters to improve detection performance.

Table 6.9: Results (F values) of evolved edge detectors from single type of filter without combination function $\mathbb{C}$ (with training image 23025) on the BSD test images.

Setting	Mean $\pm$ Standard Deviation	Max	Min	p -value
dg_σ	0.5258 ± 0.0029	0.5308	0.5206	0.0000 $\downarrow$
dog_σ	0.5004 ± 0.0074	0.5208	0.4809	0.0000 $\downarrow$
ddg_σ	0.4921 ± 0.0046	0.5004	0.4823	0.0000 $\downarrow$

Table 6.10: One-way ANOVA (row vs column) comparisons (F) among single Gaussian filters used in GP on the BSD test images.

	dog_σ	ddg_σ
dg_σ	$\uparrow$	$\uparrow$
dog_σ		$\uparrow$

Table 6.10 reveals the comparison among the results from the three types of filters. It is interesting that the results from only using dog_σ are significantly better than the results from only using ddg_σ , although dog_σ is considered as approximation of ddg_σ [14]. A potential reason is that each dog_σ filter includes two different Gaussian filters, but a ddg_σ filter only uses a single Gaussian filter. After combining filters with different parameters, the responses from the combination of dog_σ are richer than the responses from the combination of ddg_σ . Therefore, it is possible to obtain better results from only using dog_σ than only using ddg_σ .

Also, the results from using dg_σ filters are significantly better than the results from using dog_σ and ddg_σ , respectively. The mean of F values from using dg_σ filters is 0.0144 less than the mean of F values from using all functions in the GP system (see Table 6.1) when image 23025 is used as training data. It seems that the image Gaussian gradient is important for

Table 6.11: Results (F values) of evolved edge detectors by using single filters and combination function $\mathbb{C}$ (with training image 23025) on the BSD test images.

Setting	Mean $\pm$ Standard Deviation	Max	Min	p -value
$\{dg_\sigma, \mathbb{C}\}$	0.5156 ± 0.0135	0.5379	0.4789	0.0002 $\downarrow$
$\{dog_\sigma, \mathbb{C}\}$	0.4856 ± 0.0334	0.5355	0.4203	0.0247 $\downarrow$
$\{ddg_\sigma, \mathbb{C}\}$	0.4786 ± 0.0376	0.5387	0.3512	0.0592

extracting edge features. However, only combining dg_σ filters is not sufficient to improve detection performance. From the comparison among the three types of filters, it seems that dog_σ and ddg_σ help dg_σ to improve detection performance. In surround suppression, the major responses come from the image gradient. The GP Gaussian-based edge detectors also show similar behaviour.

6.5.5 Combination Function $\mathbb{C}$

The combination function $\mathbb{C}$ is not used for the experiments in Table 6.9. It is possible that the difference between using a single type of filter and the results using the three types of filters are affected by the function $\mathbb{C}$. In order to investigate the influence from function $\mathbb{C}$, the function is added into each single type of filter to evolve edge detectors. Table 6.11 shows the results for the three single types of filters after using function $\mathbb{C}$. Here the p -values are obtained by the two-sample t -tests between without function $\mathbb{C}$ and using function $\mathbb{C}$. It is surprising that using function $\mathbb{C}$ makes test performances decreased for dg_σ and dog_σ .

There are some interesting observations in Table 6.11. Firstly, using each type of filter combined with function $\mathbb{C}$ decreases the detection performance of the evolved edge detectors, in terms of the mean of F values. Secondly, the standard deviations of the evolved edge detectors be-

come larger after using function $\mathbb{C}$. This indicates that the performance of evolved edge detectors becomes less stable. Thirdly, from the comparisons of the best edge detectors and the worst edge detectors in each type of filter with and without function $\mathbb{C}$, the range of the performance of the evolved edge detectors after using function $\mathbb{C}$ becomes large. The best evolved edge detector from each setting with function $\mathbb{C}$ has higher F value than the best edge detector without function $\mathbb{C}$. However, the worst edge detector from each setting in Table 6.11 is obviously worse than the worst edge detector from the relevant setting in Table 6.9. In particular, there is very obvious influence on ddg_σ after using function $\mathbb{C}$, in terms of the performance on the best and worst evolved edge detectors. There is no significant difference between the results from using and without function $\mathbb{C}$ when only ddg_σ is used. However, the relevant worst evolved edge detector (in $\{ddg_\sigma, \mathbb{C}\}$) has very low F , and the relevant best evolved edge detector has higher F than the best evolved edge detector from ddg_σ in Table 6.9.

From the three interesting observations, the function $\mathbb{C}$ certainly influences the evolved edge detectors. It seems that using $\mathbb{C}$ might increase the ability to find good edge detectors from an overview. However, after adding function $\mathbb{C}$ in the function set, the worst evolved edge detector in each setting becomes worse. There are at least two possible reasons for this phenomenon. The first one is that good combinations of a single type of filter and function $\mathbb{C}$ are very hard to find. The other potential reason is that overfitting occurs when only a single type of filter and function $\mathbb{C}$ are employed in the GP system.

In addition, the best edge detector for each setting in Table 6.11 has lower F than the mean of the evolved edge detectors from Table 6.1 when image 23025 is used as the training data. It also shows that using a single type of filter is not good to evolve Gaussian-based edge detectors.

Table 6.12 gives the comparisons among only using a single type of filter with function $\mathbb{C}$. It also shows that the Gaussian gradient is better

Table 6.12: One-way ANOVA (row vs column) comparisons (F) among single Gaussian filters used in GP (including function $\mathbb{C}$) on the BSD test images.

	$\{dog_\sigma, \mathbb{C}\}$	$\{ddg_\sigma, \mathbb{C}\}$
$\{dg_\sigma, \mathbb{C}\}$	$\uparrow$	$\uparrow$
$\{dog_\sigma, \mathbb{C}\}$		—

than the Gaussian second-order derivative to extract edge features. When adding function $\mathbb{C}$ in the function set, there is no significant difference between only using dog_σ and only using ddg_σ .

6.6 Chapter Summary

The goal of this chapter was to investigate evolving Gaussian-based edge detectors using GP. From the different training data, the goal was successfully achieved by using a proposed Gaussian-based GP system to evolve edge detectors. The training data firstly employed a single image (six images as six different training datasets). Four combinations of two images as the training data were applied as well. After analysing the performance on these datasets (single images and the combinations), two random sampling techniques (randomly choosing one image at each generation and randomly choosing several images at each generation) were effectively used for evolving edge detectors.

A small number of full images performs better for evolving Gaussian-based edge detectors than a set of subimages. Some of the single images or combinations of two images as training data were used to train edge detectors which are better than the Gaussian gradient, and similar to the surround suppression technique. From the results, a single image with rich edge information *could* be used to evolve good edge detectors.

The evolved edge detectors from the sampling technique (randomly selecting several images from a set of images at each generation) do not have significant differences in the detection performance, compared with the evolved edge detectors from all images in the set. The evolved edge detectors from the sampling technique are significantly better than the Gaussian gradient and surround suppression on the BSD test image dataset.

From the investigation on each type of filter for constructing Gaussian-based edge detectors, the Gaussian gradient should be the main filter in a combination. However, taking into account performance improvement, different types of Gaussian filters *should* be utilised together.

The combination function $\mathbb{C}$ cannot improve detection performance when only combining a single type of Gaussian filter. In order to obtain edge detectors with good generalisation ability, the combination function $\mathbb{C}$ should choose different types of Gaussian filters as its inputs.

The Gaussian-based edge detection technique as the prior domain knowledge about edges was developed by GP. In the following chapters, GP will be used to address how to use multiple techniques as given domain knowledge for development of edge detection.

Part III

**Composite Edge Feature
Construction**

Chapter 7

Distribution-based Feature Construction

7.1 Introduction

The previous chapter investigated using the Gaussian-based technique to evolve edge detectors by GP. However, a single feature is typically not sufficient to fully identify the edges in an image, and multiple features are useful to improve detection performance [140]. For instance, features extracted from image gradients are not good to detect texture edges [14, 123, 140]. Since different features have their own advantages [140], a combination of features possibly brings the advantages of each basic feature together.

There are two ways of combining a set of features for edge detection. The first way combines different edge detectors as composite features based on a fixed model, such as a logistic regression model [123] or the combination of voting consensus as ground truth from a set of features [46]. The second way finds edge points with responses on different parameters in a method, such as multi-scale approaches [14, 139]. From these methods, using a set of features can improve detection performance [14, 123, 139, 140]. However, combining too many features together does not mean that the

performance is better than the combination of a few features. For example, the performance of Boosted Edge Learning using approximately 50000 features for natural images is only close to a contour detector proposed in [123] with nine local features [38]. How to efficiently and effectively combine features still needs to be investigated.

When GP is utilised to construct features, these composite features are usually evaluated based on accuracy or dispersion [44, 135]. In a binary classification problem, a composite feature should have good performance as its discrimination threshold is varied. The Receiver Operating Characteristic (ROC) [67] has been used to evaluate features based on different thresholds [18]. Also, the Gaussian distribution has been employed for feature performance evaluation [18, 195]. From the application of using the Gaussian distribution, the distribution of the observations of a GP program is very important in feature construction. In general, a composite feature does not directly indicate probabilities for different categories (class labels of the target). If the distribution of the observations of a GP program is estimated, the probability for each category is easy to be obtained. Chapter 4 addressed the problem of transforming observations of GP programs to soft edge maps. It might not be good to directly use observations of GP programs to represent edge responses. Using the probability for each category in edge detection is expected to more suitably represent edge responses.

When a feature is constructed for edge detection, it should indicate different responses on different areas, such as subjective edges, being dependent on the contents of interest to people, and clear boundaries. Here, subjective edges mean that they might be considered as edges by a few people, and clear boundaries mean that they are considered as edges by most people. The contrast between areas around clear boundaries is usually high. Figure 7.1 shows an image from the BSD image dataset [123] and its ground truth. The boundary indicated by “1” is clear, but the boundary indicated by “2” is considered as a true (subjective) edge by a few people

Figure 7.1: One example image with different responses on different edges.

only.

In the probabilistic view, pixels on a clear boundary should have high probabilities to be classified as edge points. The probability for a pixel to be classified as an edge point is expected to describe different edge information. The probabilistic binary approach to the distribution of the observations of an evolved program has been investigated for binary classification [167]. In our initial work, the feature constructed by GP with the Gaussian distribution shows the detected edge maps with high contrast, but they are not good to describe suitable responses on different edges. Suitable (rich) responses mean that responses on clear boundaries are high, responses on non-edge points are very weak, and responses on subjective edges are neither high nor weak. Suitable responses enrich the description of true edges. How to construct edge features with *rich responses* on edges needs to be investigated.

7.1.1 Chapter Goal

The goal of this chapter is to develop a GP system with estimated distribution to construct composite features for representing different edge responses. The main research focuses on the distribution of the observations of GP programs for constructing features to indicate rich responses in edge detection. A GP system is proposed to construct invariant features

at first, and then it is extended to construct variant features. Here, invariant features mean that they are not affected by image rotation. The image gradient and histogram gradient [123] are popularly used to train learning contour detectors, so they are used as basic features. In order to enrich the set of features for edge detection, a measure of image quality [156] is also used as a basic invariant feature. Based on these three basic features, composite features will be constructed automatically by GP. Different from directly using the observations of GP programs, the composite features come from the probabilities for the class “edge point” and the class “non-edge point” after estimating the distribution of the observations of GP programs for each class. Since the distribution of the observations of a GP program is unknown, it would be time-consuming to find the relevant distribution. In this chapter, a prior distribution is assumed for all GP programs before performing estimations. Specifically, we investigate the following research objectives.

- Whether the features constructed by GP with distribution estimation can improve the detection performance, compared with each basic invariant feature alone.
- Which relationship between probability and observations of GP programs (a linear relationship or a non-linear relationship for distribution estimation) is better to construct composite features, in terms of quantitative evaluation (with the F -measure [123]) and qualitative evaluation (from human observations on rich responses).
- Which estimation technique, employing the probability density function (PDF) or the cumulative distribution function (CDF), is better to estimate probabilities for edge points and non-edge points to construct composite features, in terms of quantitative evaluation and qualitative evaluation.
- Whether the features constructed by GP with distribution estimation

are better than the combinations from a Bayesian model [40] (non-linear combination) and a linear Support Vector Machine (SVM) [27] (linear combination).

- Whether evolved programs can be reasonably interpreted.

7.1.2 Chapter Organisation

In the remainder of this chapter, the second section introduces the background of invariant edge features. The third section proposes a GP system using different known edge features to construct composite features. The fourth section gives the experiment settings. The fifth section presents the results of the experiments with discussions. The sixth section extends the GP system from the invariant feature construction to variant feature construction when only considering horizontal edges as the major class in a binary classification problem. The seventh section provides further discussions. The last section draws a summary of this chapter.

7.2 Background of Invariant Features

In a general system to evaluate a feature in edge detection, different thresholds are used to obtain binary edge maps based on the feature, and then final edge maps after post-processing techniques are measured by an evaluation method, such as the F -measure [123]. In general, post-processing techniques, such as the surround suppression [66] and non-maximum suppression [28] techniques, linking approaches, and thinning operations [106], can usefully follow most feature extraction approaches [115]. Although the performance evaluation usually focuses on final edge maps, the evaluation of the features *without* using suppression and linking techniques in edge detection is also important [131].

Feature extraction mainly works on local features for the sake of simplicity and ease of implementation. Local features mainly come from gra-

dient computation, such as image derivatives [28, 66] or local histogram derivatives [123]. The image derivatives based on different directions have been widely applied to different edge detectors [140], such as in the popular Canny edge detector [28]. For instance, to detect a horizontal edge, the derivative of the vertical direction is calculated via using a set of neighbours around a discriminated pixel. The derivative of a Gaussian filter extracted for a horizontal edge is described in Equation (7.1). Here, σ is the scale parameter for the Gaussian filter, and (u, v) is the position of a point relative to a center (discriminated) pixel. After convolution ($\otimes$) of image $I(x, y)$ with the derivative $gd_\theta(u, v)$ (using $\theta = 0^\circ, 90^\circ$) is performed, the image Gaussian filter derivatives $T_{gd,\theta}$ are obtained (see Equation (7.3)). The invariant Gaussian gradient T_{gg} is a combination of the horizontal and vertical direction image derivatives, and it is given in Equation (7.4).

$$gd_{0^\circ}(u, v) = -\frac{v}{2\pi\sigma^4} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \quad (7.1)$$

$$gd_{90^\circ}(u, v) = -\frac{u}{2\pi\sigma^4} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \quad (7.2)$$

$$T_{gd,\theta}(x, y) = gd_\theta(u, v) \otimes I(x, y) \quad (7.3)$$

$$T_{gg}(x, y) = \sqrt{T_{gd,0^\circ}^2(x, y) + T_{gd,90^\circ}^2(x, y)} \quad (7.4)$$

Image histograms have been used in image processing, and local image histogram gradients have shown good performance for detecting edges [123]. A histogram is a visual impression of the distribution of data. In general, two sets of pixels from two different objects or textures have different histograms. For a pixel (x, y) , two groups of pixels from its neighbours can be obtained after separating them by a line in the direction θ . To generate histograms from the two groups, all graylevels from the two groups are binned into different bins. The total number of bins used in this chapter is 32. Here, $l_{\theta,i}$ and $r_{\theta,i}$ are used to indicate the number occurrences for the neighbours located in the bin i from the two groups, respectively. The local image histogram is extracted by comparing the difference of $l_{\theta,i}$ and $r_{\theta,i}$. The local histogram derivative in the direction θ is defined in

Equation (7.5). Here, $l_{\theta,i}$ and $r_{\theta,i}$ describe the distributions of two sets of pixels separated by the boundary with direction θ , so the local histogram derivative is different from the image Gaussian derivative, especially for the low-contrast areas.

$$hd_{\theta}(x, y) = \frac{1}{2} \sum \frac{(l_{\theta,i} - r_{\theta,i})^2}{l_{\theta,i} + r_{\theta,i}} \quad (7.5)$$

Note that $hd_{\theta}(x, y) \geq 0$. The local histogram gradient T_{hg} using all possible directions θ is defined in Equation (7.6). Here, the local histogram gradient T_{hg} is combined with the histogram derivatives from two directions as one invariant feature. Different from T_{gg} (constructed from the horizontal and vertical directions), T_{hg} is constructed from $\theta = 45^{\circ}, 135^{\circ}$ so that different directional responses are used. A reason for only using the two directions is that our previous work [52] shows that the diagonal derivatives are better than the horizontal and vertical derivatives for detecting edges. For the test image dataset in the BSD dataset [123], the performance of T_{hg} with $\theta = 45^{\circ}, 135^{\circ}$ is almost the same as T_{hg} with $\theta = 0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ}$.

$$T_{hg} = \sum_{\theta} hd_{\theta}(x, y) \quad (7.6)$$

The normalised standard deviation has been used for image quality evaluation [156], but it is seldom used as a feature for edge detection in the literature. The local normalised standard deviation is utilised in here. It is extracted based on a small window as a local invariant feature. The local normalised standard deviation T_{sd} is defined in Equation (7.7), where $Mean(x, y)$ and $SD(x, y)$ are the mean and standard deviation of the pixel intensities in a local area around (x, y) , respectively.

$$T_{sd}(x, y) = \frac{SD(x, y)}{Mean(x, y)} \quad (7.7)$$

From our initial investigation, the three basic features are good at detecting clear boundaries. However, the local features from derivatives

Figure 7.2: Detected image based on Gaussian gradients T_{gg} , histogram gradients T_{hg} and normalised standard deviations T_{sd} .

include high responses on non-edge points which are affected by noise or textures, and the responses on some subjective edges and textures are hard to distinguish. For instance, Figure 7.2 shows the features of an image from the BSD image dataset with T_{gg} , T_{hg} and T_{sd} . Four different areas are marked with four ellipses. Area 1 includes a clear boundary between an object in the background and the sky. Areas 2 and 3 include subjective edges. Area 2 has a boundary between two similar background objects, and area 3 includes the boundary between the reflection of an object and the water. Area 4 has a water wave texture, and there are no edge points. From the three basic features, their responses on the clear boundary in area 1 are suitably indicated (higher than the other responses). For the subjective edges, T_{gg} and T_{sd} have a problem to distinguish responses on the boundary in area 2 and the responses on the textures from one object. T_{gg} and T_{sd} have suitable responses on the edge in area 3, which are lower than the responses on the clear boundary in area 1 and higher than the responses on most non-edge points. Although T_{hg} gives suitable responses

on area 2, it wrongly gives responses on the water wave in area 4 so that these responses are higher than the responses on subjective edges in area 3. The surround suppression technique can reduce texture responses on area 2 from the image gradient T_{gg} [66]. Therefore, further feature manipulation on these local features can improve detection performance. However, how to indicate suitable responses on different types of edges and remove responses on non-edge points needs to be addressed.

7.3 Distribution-based GP System for Feature Construction

The GP system using distribution to construct composite features is described in this section. The observations of GP programs for edge points and non-edge points are used to fit given distributions (using PDFs or CDFs) to indicate edge responses.

7.3.1 Terminal Set

In order to construct composite features, basic features are used as terminals. Invariant features are employed for invariant feature construction. Note that only grayscale images are used. Three features are chosen in the terminal set of the proposed GP system. The three features are the image Gaussian filter gradient T_{gg} , the normalised standard deviation T_{sd} , and the histogram gradient T_{hg} . Since the three features have their own characteristics (see Figure 7.2), it is possible to construct new features for improving detection performance. Note that the basic features are not normalised. Random constants rnd in the range from -100 to 100 are chosen in the GP system.

7.3.2 Primitive Functions

The function set contains the four arithmetic operators $\{+, -, *, \div\}$ and three logical operators $\{IF, <, >\}$. Here, $\div$ is protected division, producing a result of 1 for a 0 divisor. IF contains three arguments, the first one is a boolean. IF will return the second argument (a real number) when the first is true, otherwise will return the third argument (a real number). It is possible that a feature is better than another feature for some edges, but worse for other edges. Based on the logical operators, two features are combined together, and GP will automatically choose partial responses (suitably indicating some edge information) from one feature. That is why the logical operators are used.

7.3.3 Composite Features

From a survey of using GP for classification [44], it is very important to estimate the distribution of the observations o of a program for feature construction in classification. Bayesian error rate (the lowest possible error rate for a given class of classifier) is a common way to estimate the effectiveness of constructed features [177, 196]. Also, the minimum distance from observations of evolved programs to a class position (such as its mean) has been used for classification [76]. These approaches using estimation of observations of programs are helpful to find good composite features, but since these features themselves directly use observations of programs, the relationship between a composite feature and the target (such as ground truth in edge detection) is not clear.

In general, if an observation of an evolved program is larger than a threshold, the relevant pixel is discriminated as an edge point. However, in the proposed GP system, the observation is used to indicate the probability for the pixel to be discriminated as the class edge point or the class non-edge point, rather than directly using the observation to classify the relevant pixel. The probability for each class is calculated based on a

known distribution model. A distribution is estimated by using the sample mean and standard deviation of observations of evolved programs for edge points and non-edge points. The probabilities for edge points and non-edge points are used to construct a composite feature. Therefore, the composite feature is based on probability estimation, which more clearly indicates the relationship between the feature and the target (edge points or non-edge points).

When most observations are around a threshold and a few observations are far away from the threshold, it is difficult to describe rich edge responses on different areas. Generally, a clear boundary (high contrast) in an image is easy to be detected. The probability for discriminating a pixel on the boundary as an edge point should be high. Therefore, the probabilities of a pixel belonging to the class edge point and the class non-edge point are utilised here to indicate edge responses. To obtain both probabilities, the distribution of the observations of an evolved program for each class is estimated. After estimating both distributions, both probabilities from the estimated distributions are expected to construct single edge features which clearly indicate edge responses on different areas.

For the estimated distribution from PDFs and CDFs, two equations are proposed combining the probabilities for edge points and non-edge points as edge responses. The first equation for constructing a composite feature is based on a known distribution model from PDFs, which is described in Equation (7.8). Here, $p_+(o, \hat{\mu}_+, \hat{\sigma}_+)$ (abbreviated as $p_+(o)$) is the probability density for edge points based on the known distribution model, $p_-(o, \hat{\mu}_-, \hat{\sigma}_-)$ (abbreviated as $p_-(o)$) is the probability density for non-edge points, $\hat{\mu}_+$ and $\hat{\sigma}_+$ are the sample mean and standard deviation for edge points, $\hat{\mu}_-$ and $\hat{\sigma}_-$ are the sample mean and standard deviation for non-edge points, P_+ is the prior probability for edge points, and P_- is the prior probability for non-edge points. Note that the prior probabilities for edge points and non-edge points are considered to be equal ($P_+ = P_- = 0.5$) since we will use balanced training data (by sampling techniques). After

simplifying $GP_{pdf}(o)$, their probability densities are used to indicate the posterior probabilities for discriminating pixels as edge points and non-edge points. A composite feature using $GP_{pdf}(o)$ is based on Bayesian inversion [40]. More details about Bayesian techniques will be addressed in the next chapter.

$$GP_{pdf}(o) = \frac{p_+(o, \hat{\mu}_+, \hat{\sigma}_+)P_+}{\sum_{k=+,-} p_k(o, \hat{\mu}_k, \hat{\sigma}_k)P_k} = \frac{p_+(o, \hat{\mu}_+, \hat{\sigma}_+)}{p_+(o, \hat{\mu}_+, \hat{\sigma}_+) + p_-(o, \hat{\mu}_-, \hat{\sigma}_-)} \quad (7.8)$$

Since the construction is based on $p_+(o)$ and $p_-(o)$, when the values of observations of a program are close to the means of relevant classes (edge points and non-edge points), the program is considered as a good candidate to construct composite features. When the probability of the observation of a GP program for the relevant pixel as an edge point is higher and the probability for considering as a non-edge point is lower, the pixel should have a stronger response from the composite feature. For a pixel with lower probability for an edge point and higher probability for a non-edge point, the composite feature has a weaker response. For an uncertain area, the composite feature gives a response which is neither strong nor weak. Therefore, the composite feature based on Equation (7.8) directly gives a clear response for a pixel as an edge point or a non-edge point.

However, when threshold techniques are used to discriminate pixels with a composite feature based on Equation (7.8), the probability of a pixel considered as the class edge point is not high when its output from the relevant program is far away from the mean of the class edge point. If the pixel is directly discriminated based on the observation of the program (suppose $\hat{\sigma}_+ = \hat{\sigma}_-$), it will be clearly considered as an edge point (shorter distance to $\hat{\mu}_+$ than $\hat{\mu}_-$) or non-edge point (longer distance to $\hat{\mu}_+$ than $\hat{\mu}_-$). In order to give a suitable probability for this pixel, the second way is proposed to employ CDFs. CDFs indicate a suitable range of responses on edge points and non-edge points, rather than expecting that all observations are around two fixed positions.

Figure 7.3: Probabilities for edge points and non-edge points on the observations of an example edge detector.

Note that a composite feature using Equation (7.8) has a problem of thick responses based on our initial experiment. In order to address the problem, the composite feature using CDFs is defined in Equation (7.9). Here $P_+(o)$ is the estimated probability from the CDF fitted to the observations (considered as a known distribution for edge points), and $P_-(o)$ is the estimated probability for non-edge points. When a composite feature is indicated by Equation (7.9), a pixel with the observation of the relevant program (with $\hat{\sigma}_+ = \hat{\sigma}_-$) far away from $\hat{\mu}_+$ and $\hat{\mu}_-$ has very high probability for an edge point (shorter distance to $\hat{\mu}_+$ than $\hat{\mu}_-$) or a non-edge point (longer distance to $\hat{\mu}_+$ than $\hat{\mu}_-$). When an observation has $P_+(o) < P_-(o)$, its relative pixel is directly considered as a non-edge point and the response is represented by 0.

$$GP_{cdf}(o) = \begin{cases} 0 & \text{if } P_+(o) < P_-(o) \\ P_+(o) - P_-(o) & \text{otherwise} \end{cases} \quad (7.9)$$

Figure 7.3 show the probabilities for edge points and non-edge points on the observations of an example GP edge detector (with Gaussian distributions, $\mu_- = -3.5$, $\mu_+ = 3.5$, $\sigma_- = 2$, and $\sigma_+ = 3$). Note that the direction of CDF curves for edge points and non-edge points are opposite, so $P_-(o)$ is the difference of 1 and the fitted probability from the observations. A

pixel with an observation “A” (close to 14 in the horizontal axis) is not strongly considered as an edge point when the PDFs are used to estimate the observation “A”. In fact, the pixel should be considered as an edge point with a high probability. If the CDFs are employed to estimate “A”, it is easy to discriminate the pixel as an edge point. Also, the overlap between both classes from their CDFs is smaller than the overlap from their PDFs. A composite feature from PDF estimation using Equation (7.8) has a problem when the construction equation is replaced by Equation (7.9). For instance, “A” in Figure 7.3 (a) has no response when PDF estimation uses Equation (7.9).

Note that the estimation based on PDFs and CDFs is not limited to a distribution which can only be estimated by its mean and standard deviations. Other distributions could be employed in this GP system as well.

7.3.4 Fitness Function

In the training stage, when $p_+(o) > p_-(o)$ or $P_+(o) > P_-(o)$, the pixel with o is predicted as an edge point (+), otherwise as a non-edge point (−). The fitness function is based on evolved composite features, following the suggestion of directly evaluating features from [131]. The aim of new features constructed by GP is that they should detect as many true edge points as possible. Recall p_{rec} (see Equation (2.7) on page 35) is the number of pixels on the edges correctly detected as a proportion of the total number of pixels on the edges, so it is a very important indicator. However, only considering recall is not sufficient to evaluate constructed features because high recall might bring low accuracy for non-edge points. Specificity p_{spe} (see Equation (2.8) on page 35) is the number of pixels not on the edges correctly detected as a proportion of the total number of pixels not on the edges. Therefore, specificity p_{spe} is also considered in the fitness function. The fitness function Fit is defined in Equation (7.10) in the training phase.

$$Fit = \frac{2p_{rec}p_{spe}}{p_{rec} + p_{spe}} \quad (7.10)$$

7.4 Experiment Design

To investigate the distribution of observations of GP programs, a piecewise linear relationship and a non-linear relationship are selected for both PDFs and CDFs. The selected known distribution for PDFs with a linear relationship is not the same as the selected known distribution for CDFs with a linear relationship.

7.4.1 Training Data

Since the BSD training image dataset [123] includes 200 images, using all pixels and the relevant basic features will lead to a heavy computational cost. For simplicity, image pixels with the same number of edge points and non-edge points are sampled as the training data. Approximately 250 edge points and 250 non-edge points are randomly sampled from each training image. Therefore, the training data includes approximately 100000 cases.

7.4.2 Estimated Distribution

The Gaussian distribution is commonly employed for evaluating observations of evolved programs [44]. Therefore, it is chosen as the non-linear relationship for distribution estimation in this chapter. In the Gaussian distribution (see Equation (7.11), where $j = +$ or $-$), the aim of good observations is that the observations are as close to the relevant means as possible.

$$g_j(o, \hat{\mu}_j, \hat{\sigma}_j) = \frac{1}{\hat{\sigma}_j \sqrt{2\pi}} \exp \left(-\frac{(o - \hat{\mu}_j)^2}{2\hat{\sigma}_j^2} \right) \quad (7.11)$$

Similar to the Gaussian distribution, the second PDF is based on distance from an observation to $\hat{\mu}_j$, and it is defined by $t_j(o, \hat{\mu}_j, \hat{\sigma}_j)$ in Equation (7.12). Note that $t_j(o, \hat{\mu}_j, \hat{\sigma}_j)$ is transformed to the range from 0 to 1. Different from $g_j(o, \hat{\mu}_j, \hat{\sigma}_j)$, the probability for an observation discriminated as the class edge point or the class non-edge point is linearly de-

creased when its position is away from $\hat{\mu}_j$. This distribution is a symmetric *triangular* distribution. In a triangular distribution, the half range for observations (symmetric distribution) should be $\sqrt{6}\hat{\sigma}$. However, $3\hat{\sigma}$ is employed in this chapter so that a wider range of observations can be accepted to give responses than using $\sqrt{6}\hat{\sigma}$.

$$t_j(o, \hat{\mu}_j, \hat{\sigma}_j) = \begin{cases} 0 & \text{if } |o - \hat{\mu}_j| > 3\hat{\sigma}_j \\ 1 - \frac{|o - \hat{\mu}_j|}{3\hat{\sigma}_j} & \text{otherwise} \end{cases} \quad (7.12)$$

Also, the Gaussian distribution is considered as one known distribution for estimating CDFs, and its CDF is estimated for $P_-(o)$ and $P_+(o)$. In the estimation using CDFs of the Gaussian distribution of observations of a GP program, a pixel has higher probability to be an edge point when its relevant observation (greater than $\hat{\mu}_+$) is farther away from the position $\hat{\mu}_+$ of the program for edge points. In the estimation using CDFs, the long distance from an observation to the position $\hat{\mu}_+$ or $\hat{\mu}_-$ makes the relevant pixel clear to be considered as an edge point or a non-edge point. However, in the PDF estimation, good programs should have observations being close to the position $\hat{\mu}_+$ or $\hat{\mu}_-$. Therefore, the restriction of the range of the observations using CDF are looser than using PDF.

The CDF of the Gaussian distribution is defined in Equation (7.13). $Norm_j(o, \hat{\mu}_j, \hat{\sigma}_j)$ is estimated based on the normal distribution model with $\hat{\mu}_j$ and $\hat{\sigma}_j$.

$$\begin{aligned} erf(x) &= \frac{1}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt \\ \Phi(o) &= \frac{1}{2} \left(1 + erf\left(\frac{o}{\sqrt{2}}\right) \right) \\ Norm_j(o, \hat{\mu}_j, \hat{\sigma}_j) &= \Phi\left(\frac{o - \hat{\mu}_j}{\hat{\sigma}_j}\right) \end{aligned} \quad (7.13)$$

Besides the CDF of the Gaussian distribution, the CDF of a uniform distribution (affine function) is employed, and it is defined in Equation (7.14). Since the CDF of the triangular distribution has a non-linear (quadratic)

relationship between the probability and observations, which is similar to the CDF of the Gaussian distribution (being non-linear), we choose the uniform distribution in the GP system, rather than using the CDF of triangular distribution. It is assumed that observations o are uniformly distributed in the range from $\hat{\mu}_j - 3\hat{\sigma}_j$ to $\hat{\mu}_j + 3\hat{\sigma}_j$.

$$Unif_j(o, \hat{\mu}_j, \hat{\sigma}_j) = \begin{cases} 0 & \text{if } o < \hat{\mu}_j - 3\hat{\sigma}_j \\ \frac{o - \hat{\mu}_j + 3\hat{\sigma}_j}{6\hat{\sigma}_j} & \text{if } |o - \hat{\mu}_j| < 3\hat{\sigma}_j \\ 1 & \text{otherwise} \end{cases} \quad (7.14)$$

In summary, $GP_{pdf}(o)$ has $g_j(o, \hat{\mu}_j, \hat{\sigma}_j)$ (non-linear) and $t_j(o, \hat{\mu}_j, \hat{\sigma}_j)$ (linear) for $p_j(o)$, and $GP_{cdf}(o)$ has $Norm_j(o, \hat{\mu}_j, \hat{\sigma}_j)$ (non-linear) and $Unif_j(o, \hat{\mu}_j, \hat{\sigma}_j)$ (linear) for $P_j(o)$ in this chapter.

7.4.3 Parameter Settings

The window sizes for T_{gg} , T_{sd} and T_{hg} are 7×7 , 3×3 and 13×13 , respectively. It is assumed that T_{gg} uses the 7×7 window to mainly detect local edge details, T_{sd} uses the 3×3 window to find changes in a very small area, and T_{hg} uses the 13×13 window to find difference between two areas. It is possible that T_{gg} has strong responses on textures. T_{hg} , using a large window, is expected to reduce influence from textures. The parameter values for GP are: population size 200; maximum generations 200; maximum depth (of a program) 7; and probabilities for mutation 0.15, crossover 0.80 and reproduction 0.05. These values are chosen based on common settings and initial experiments [140]. There are 30 independent runs for each experiment.

7.4.4 Test Performance

The evaluation is directly based on one feature, without special post-processing techniques, such as non-maximum suppression. To measure the performance of these features constructed by GP, the F -measure (see Equation (2.13))

is used in the testing phase [38, 123]. In the F -measure evaluation system, pixels are discriminated as edge points when their values of the evaluated feature are larger than a threshold. The predicted edges are simply thinned by the thinning operator [123]. After obtaining thinned prediction, an optimal matching operator will be used to match the prediction and the ground truth. Here, α is also set to 0.5 based on a common setting [38, 123]. The previous chapter used a fixed threshold 0 for performance evaluation. In this chapter, all constructed features are evaluated by multiple thresholds. Using different threshold values $\frac{k}{52}$ ($k = 0, 1, \dots, 51$), a maximum $F_{max} = \max\{F_k\}$ will be considered as the measurement for the feature, where F_k is the F value when the threshold value $\frac{k}{52}$ is used.

For fair comparison, the F_{max} values of the three basic features (T_{gg} , T_{sd} , T_{hg}) also are given without special post-processing techniques, such as non-maximum suppression, so their values are different from the final performance evaluation in [123]. In the feature performance evaluation, the Sobel edge detector is the same result as the final detection result evaluated in [123], because the Sobel edge detector does not contain special post-processing techniques.

7.5 Results and Discussions

This section describes the results from the composite features constructed by GP, conducts comparisons among the GP constructed features and existing combination techniques, provides visual detected results, and interprets a GP evolved program.

7.5.1 PDF and CDF

Training Results

Figure 7.4 shows the histograms of observations o of four GP programs from the four settings: $GP_{PDF, Norm}$ (estimation using PDFs of the Gaus-

Figure 7.4: Histograms of observations of a GP program from each setting ($GP_{PDF, Norm}$, $GP_{PDF, Tri}$, $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$) on the training data.

sian distribution), $GP_{PDF, Tri}$ (estimation using PDFs of the triangular distribution), $GP_{CDF, Norm}$ (estimation using CDFs of the Gaussian distribution), and $GP_{CDF, Unif}$ (estimation using CDFs of the uniform distribution) on the *training* data. Since the ranges of observations of the GP program from $GP_{PDF, Tri}$ are very large for true edge points and very narrow for true non-edge points, only the main area is shown, and the other area (not shown, less than one percentage of all areas) is from true edge points. From the training data, GP with PDFs does not have good estimation on the observations of GP programs. It is interesting that most of the observa-

Table 7.1: Comparison of means and ($\pm$) standard deviations F_{max} values among constructed features by GP with PDF and CDF for the BSD test images.

	F_{max}
$GP_{PDF, Norm}$	0.5728 ± 0.0292
$GP_{PDF, Tria}$	$0.5478 \pm 0.0475 \downarrow$
$GP_{CDF, Norm}$	0.5790 ± 0.0218
$GP_{CDF, Unif}$	0.5785 ± 0.0186

tions in each class are located one side of their sample means when PDFs are used. From CDFs, the observations of GP programs in each class are close to the estimated curves. Since one edge feature is used to discriminate a pixel as an edge point based on a threshold, namely its value is larger than the threshold (a range), not close to the threshold (a point or a short area), the estimated distribution using PDFs is different from the actual distribution of observations of GP programs. However, CDFs are based on a range (less than a threshold), which makes the estimated distributions close to the actual distribution of observations.

Test Performance

In order to investigate the test performance on the evolved composite features based on the estimation using PDFs and CDFs, Table 7.1 gives the means and standard deviations of F_{max} of $GP_{PDF, Norm}$, $GP_{PDF, Tria}$, $GP_{CDF, Norm}$, and $GP_{CDF, Unif}$. Here, $\downarrow$ indicates the results from $GP_{PDF, Tria}$ are significantly worse than the others based on one-way ANOVA test with overall significance level 0.05 and p -value adjustment using Holm's method [75] over all comparisons. From these comparisons, the PDF estimated as the triangular distribution is not very good to construct composite features. However, the CDF based on the uniform distribution has

Figure 7.5: F_{max} values from $GP_{PDF, Norm}$, $GP_{PDF, Tria}$, $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$ on the 100 BSD test images.

non-significantly different results, compared with the estimation based on the Gaussian distribution (with PDF or CDF).

Figure 7.5 shows the details for the 30 composite features (30 independent runs) from $GP_{PDF, Norm}$ ("PDF(Norm)"), $GP_{PDF, Tria}$ ("PDF(Tria)"), $GP_{CDF, Norm}$ ("CDF(Norm)") and $GP_{CDF, Unif}$ ("CDF(Unif)"). From the box-plots, the composite features from $GP_{PDF, Tria}$ have a larger range of F_{max} values than the others. It seems that the PDF estimated as the triangular distribution is not stable to construct good features. Comparing the CDFs with the PDFs, the F_{max} values of most of the composite features based on CDFs crowd together, which indicates the CDFs are more robust than the PDFs to use to construct composite features.

Comparing the results in Table 7.1 and Figure 7.5, it seems that the observations of a good evolved edge detector should crowd around a position for each class when the PDF is estimated as the Gaussian distribution, or should stand far away from a threshold for each class when the CDF estimation is used. When observations for each class (edge point or non-edge point) are uniformly distributed over in a range, the estimated probability might not be good to construct features. A potential reason for the behaviour of the test performance of $GP_{PDF, Tria}$ is that the estimated

Table 7.2: Comparison of F_{max} values among constructed features by GP with PDF (Gaussian distribution), image Gaussian gradients T_{gg} , normalised standard deviations T_{sd} , histogram gradients T_{hg} , Sobel edge detector, a Bayesian model [40], and SVM [27] for the BSD test images.

	F_{max}
$GP_{PDF, Norm}$	0.5728 ± 0.0292
T_{gg}	0.5153
T_{sd}	0.4968
T_{hg}	0.5434
Sobel	0.4832
Bayesian	0.5302
SVM	0.5681 ± 0.0033

distributions for edge points and non-edge points have a large overlap. The range of observations for each class on test images might be large so that some observations of an evolved program are located in the overlap area, although the observations from the training data might be separated very well. How to improve the bad features (with low F_{max} value) from the PDF estimated as the triangular distribution will be addressed in Section 7.7.3.

7.5.2 $GP_{PDF, Norm}$ vs Existing Methods

Table 7.2 gives the mean and standard deviation of F_{max} values from the 30 features constructed by $GP_{PDF, Norm}$, and F_{max} values from T_{gg} , T_{sd} , T_{hg} , the Sobel edge detector, a simple Bayesian model based on the general multivariate normal density [40], and a linear Support Vector Machine (SVM) [27]. The parameters of the linear SVM are optimised by a hybrid Partial Swarm Optimisation with Differential Evolution algorithm (HPSO-DE) [54].

Table 7.3: Statistical p -values (t -tests) for the comparisons among GP based on PDFs and CDFs, SVM and the Bayesian model for the BSD test images.

	SVM	Bayesian
$GP_{PDF, Norm}$	0.3820	0.0000 $\uparrow$
$GP_{PDF, Tria}$	0.0276 $\downarrow$	0.0561
$GP_{CDF, Norm}$	0.0130 $\uparrow$	0.0000 $\uparrow$
$GP_{CDF, Unif}$	0.0062 $\uparrow$	0.0000 $\uparrow$
SVM		0.0000 $\uparrow$

The training time for each composite feature constructed by $GP_{PDF, Norm}$ is around 3 hours on a single machine with CPU 3.1 GHz based on an implementation in C++, but the test time is only several milliseconds after loading the basic features into the memory. The test results show that the features constructed by GP effectively improve the detection performance, compared with the three basic features. From the comparisons between GP and the linear SVM, and between GP and the Bayesian model, the composite features from GP with $GP_{PDF, Norm}$ have similar performance (F_{max}) to the combinations from the linear SVM, and are better than the Bayesian model. Therefore, it can be seen that $GP_{PDF, Norm}$ is effective for automatic construction of invariant features.

7.5.3 Comparison Among GP, SVM and the Bayesian Model

Since the calculation in the training stage for GP using GP_{PDF} or GP_{CDF} is almost the same, the training times for these different settings are not remarkably different. Table 7.3 gives p -values from t -tests between the detection results from GP ($GP_{PDF, Norm}$, $GP_{PDF, Tria}$, $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$, respectively) and the linear SVM, between GP ($GP_{PDF, Norm}$, $GP_{PDF, Tria}$, $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$, respectively) and the Bayesian model, and between the linear SVM and the Bayesian model. From the table, the p -

values from the constructed features with the two known PDFs are higher than constructed features with the two CDFs, in terms of the combinations from the linear SVM. From these p -values, the features constructed by GP with the estimated CDFs are significantly better than the linear SVM and Bayesian model with significance level 0.05, using Holm's method [75] for p -value adjustment, if only comparing among $GP_{CDF, Norm}$ $GP_{CDF, Unif}$ and the linear SVM, or among $GP_{CDF, Norm}$ $GP_{CDF, Unif}$ and the Bayesian model. Note that $GP_{PDF, Trial}$ is significantly worse than the linear SVM. It seems that the estimation based on CDFs is effective to construct features for edge detection, which is better than the linear SVM and Bayesian model. Overall, CDF is better than PDF to construct features.

7.5.4 Non-parametric Statistics on the Results from GP, SVM and the Bayesian Model

Since the F_{max} values from $GP_{PDF, Trial}$ are not normally distributed, Mann-Whitney-Wilcoxon tests [102] are employed to compare the features constructed by GP with the combinations from the linear SVM and the Bayesian model. Mann-Whitney-Wilcoxon tests are employed to check whether the number of the features from GP is significantly larger than the combinations from SVM and the Bayesian model. Table 7.4 gives p -values for the comparisons between GP and the linear SVM, and between GP and the Bayesian model. From the table, all the results from the features constructed by GP, except for $GP_{PDF, Trial}$, are significantly better than the combinations from the linear SVM. All results from GP are significantly better than the combination from the Bayesian model. There are no significant differences between the results from $GP_{PDF, Trial}$ and the linear SVM.

In order to compare results among constructed features by GP and the combinations from the linear SVM, the Kruskal-Wallis test [174] is employed, with Holm's method for p -value adjustment with overall significance level 0.05. Since the Bayesian model is clearly worse than GP and

Table 7.4: Statistical p -values (with Mann-Whitney-Wilcoxon tests [102]) for the comparisons among GP based on PDFs and CDFs, SVM and the Bayesian model for the BSD test images.

	SVM	Bayesian
$GP_{PDF, Norm}$	0.0009 $\uparrow$	0.0000 $\uparrow$
$GP_{PDF, Tria}$	0.3422	0.0039 $\uparrow$
$GP_{CDF, Norm}$	0.0000 $\uparrow$	0.0000 $\uparrow$
$GP_{CDF, Unif}$	0.0062 $\uparrow$	0.0000 $\uparrow$
SVM		0.0000 $\uparrow$

Table 7.5: Kruskal-Wallis statistic [174] of F_{max} values among constructed features by GP with PDFs and CDFs on the BSD test images.

	$GP_{PDF, Tria}$	$GP_{CDF, Norm}$	$GP_{CDF, Unif}$	SVM
$GP_{PDF, Norm}$	$\uparrow$	—	—	$\uparrow$
$GP_{PDF, Tria}$		$\downarrow$	—	—
$GP_{CDF, Norm}$			—	$\uparrow$
$GP_{CDF, Unif}$				$\uparrow$

SVM in terms of F_{max} on the test images, only the linear SVM is compared here. In Table 7.5, a comparison for two groups of F_{max} values comes from the relevant indication in the first column (the first group) and the relevant indication in the first row (the second group). Here, “ $\uparrow$ ” indicates that the results from the first group is significantly better than the results from the second group, “ $\downarrow$ ” indicates that the results from the first group is significantly worse than the results from the second group, and “—” for no significant difference. From the compared statistical results, all features from GP, except for $GP_{PDF, Tria}$, are significantly better than the combinations from the linear SVM.

Figure 7.6: Recall and precision of the best composite features from $GP_{PDF, Norm}$, $GP_{PDF, Tria}$, $GP_{CDF, Norm}$, $GP_{CDF, Unif}$ and SVM, and the Bayesian Model on the 100 BSD Test Images.

From the non-parametric tests, the features constructed by GP with CDFs are significantly better than the combinations of the linear SVM and the Bayesian method. Although there are no significant differences between the results from $GP_{PDF, Norm}$ and the results from GP using CDFs ($GP_{CDF, Norm}$ and $GP_{CDF, Unif}$), GP using CDFs improves the detection performance on the natural images, in terms of the comparisons with the linear SVM.

7.5.5 Recall and Precision for Best Evolved Programs

Figure 7.6 reveals the recall and precision values of the best composite features from $GP_{PDF, Norm}$, $GP_{PDF, Tria}$, $GP_{CDF, Norm}$, $GP_{CDF, Unif}$ and SVM, and the simple Bayesian model. Here, “@” is the position for F_{max} in each curve. From an overall view of these curves, the composite features constructed by GP are better than the combinations from the linear SVM and the Bayesian model. The curves of the best composite features con-

structed by $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$ almost overlap, and the curves for $GP_{PDF, Norm}$ and $GP_{PDF, Tri}$ are close to the curves for $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$. The interesting phenomenon is that the estimation using a fixed position for each class (in PDFs of the Gaussian distribution) have a narrower range for the recall and precision values than the estimation using a wide range (such as CDFs). In the Bayesian method and $GP_{PDF, Norm}$, the positions from the observations of a program or basic features are strongly related to a fixed position for each class, and good dispersion makes the observations for a class very close to the relevant mean of the class. Since the position for basic features is multiple dimensional, the range of recall and precision values are narrower than the range from the best feature constructed by $GP_{PDF, Norm}$.

7.5.6 Qualitative Evaluation on Detected Images

Figure 7.7 visually shows the detection results for four example images from T_{gg} , T_{sd} , T_{hg} . As can be seen, T_{gg} finds most of the true edge points but is affected by textures in the four images. T_{sd} has good responses on the clear boundaries of objects, but is also affected by textures. T_{hg} suppresses the influence of textures and noise, but has too stronger responses on the discontinuous background, such as the water wave in image 101087.

Figure 7.8 visually shows the detection results for the same four example images from $GP_{PDF, Norm}$, $GP_{PDF, Tri}$, $GP_{CDF, Norm}$, $GP_{CDF, Unif}$, the Bayesian model, and the linear SVM. For the features constructed by GP, there are several interesting observations. Firstly, $GP_{PDF, Norm}$ gives obvious responses on most of the true edge points, but still has responses on some textures, such as the wall texture in image 385039. Note that the responses on the wall are lower than the responses on the boundary of the wall, although the responses on the wall have high magnitudes. Secondly, $GP_{PDF, Tri}$ gives responses for almost all pixels in the four images. Since the estimation is based on the PDF of the triangular distri-

Figure 7.7: Four example images detected by T_{gg} , T_{sd} , T_{hg} .

bution, the non-edge points still have very low responses. However, the contrast of the responses between edge points and non-edge points are high. The responses from $GP_{PDF,Tria}$ are much richer than the responses from $GP_{PDF,Norm}$. Thirdly, $GP_{CDF,Norm}$ and $GP_{CDF,Unif}$ almost give the same detection results, which have clearly rich responses on subjective true edge points. For example, the boundary of the reflection in the water from image 101087 only marked by a few observers has lower responses in the best composite features from $GP_{CDF,Norm}$ and $GP_{CDF,Unif}$ than the other responses on the clear boundaries in the image. For the non-edge points, both composite features have very weak or no responses.

Figure 7.8: Four example images detected by $GP_{PDF, Norm}$, $GP_{PDF, Tria}$, $GP_{CDF, Norm}$, $GP_{CDF, Unif}$, the Bayesian model, and the linear SVM.

The composite feature from the Bayesian model presents similar detection characteristics (high responses on most true edge points) to $GP_{PDF, Norm}$. However, the composite feature does not totally suppress the high responses on discontinuous changes in background, which come from T_{hg} , such as the responses on the water wave in image 101087. The combination from the linear SVM has similar characteristics (wide range of re-

sponses on most true edge points) to $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$. However, the high responses on non-edge points from T_{hg} are not suppressed very well. For example, the responses on the discontinuous area from the water in image 101087 are similar to the responses on the subjective true edge points from the boundary of the reflection.

In summary, $GP_{CDF, Norm}$ and $GP_{CDF, Unif}$ effectively construct features via removing high responses on non-edge points and giving rich responses on different edge points. The best features constructed by GP based on the known distribution models are better than the combinations from the linear SVM and the Bayesian model.

7.5.7 Example Evolved Program

In order to analyse evolved programs, Equation (7.15) presents an evolved program with $F_{max} = 0.6009$ for its constructed feature. Here, $CF_{PDF, Norm}$ indicates an evolved program from $GP_{PDF, Norm}$, and $A > B ? L : R$ is equal to “If $A > B$ then L else R ”. The evolved program has $\hat{\mu}_- = -11.31$, $\hat{\mu}_+ = -5.3$, $\hat{\sigma}_- = 26.26$, and $\hat{\sigma}_+ = 27.84$. This program is complicated, and it includes six small parts. From these small parts, we can find several interesting expressions.

$$CF_{PDF, Norm} = \frac{cf_1 + cf_2}{cf_3 cf_4 + cf_6} \quad (7.15)$$

$$cf_1 = 15272.87 T_{sd} T_{hg} + (T_{sd} > T_{hg} ? 39.66 : 80.19) \quad (7.16)$$

$$cf_2 = 95.23 - (61.23 T_{gg} T_{sd} + 39.67) (39.67 - \frac{0.76}{T_{sd}}) \quad (7.17)$$

$$cf_3 = 18.66 T_{gg} > T_{hg} ? 80.19 T_{gg} T_{hg} : 80.19 T_{sd} \quad (7.18)$$

$$cf_4 = 18.66 T_{gg} > 39.67 T_{sd} ? 156.64 : (81.53 - 80.19 T_{hg}) \quad (7.19)$$

$$cf_5 = T_{hg} < T_{gg} ? (T_{hg} - 94.03) : 55.55 T_{hg} \quad (7.20)$$

$$cf_6 = cf_5 < 23.86 ? 95.12 : (18.66 T_{hg} + 80.19 - T_{gg}) \quad (7.21)$$

Firstly, the logical operators IF , $>$ and $<$ are used to combine different basic features or constant values to construct the composite feature, such as $(T_{sd} > T_{hg} ? 39.66 : 80.19)$ in cf_1 (see Equation (7.16)). It seems

that the three operations are important for constructing composite features. A potential reason is that the logical operations can help the program to automatically select parts of basic features, which accurately indicate responses on edge points or non-edge points. Based on different comparisons, partial ranges of the basic features or combinations of basic features are considered as factors to help classify pixels. Secondly, multiplication might be used to enhance the responses on edge points [12], such as $T_{sd}T_{hg}$ in cf_1 and $80.19T_{gg}T_{hg}$ in cf_3 (see Equation (7.18)). Lastly, the division possibly performs a comparison between two sub-programs ($cf_1 + cf_2$ and $cf_3cf_4 + cf_6$). The two sub-programs might be used to construct sub-features to improve detection performance. However, division in here is complicated, which would be investigated in future.

Additionally, the expressions which include the three logical operators are different. Firstly, two basic features are directly compared in cf_1 . A constant value will be obtained after the comparison. Secondly, basic features are zoomed before a comparison, such as $18.66T_{gg}$ in cf_3 . Since the response magnitudes for edge points from basic features are different, it is possible to use a zoom operation to transform the response magnitudes from the two basic features to the same range so that the comparison is more reasonable. In cf_3 , a scale factor is applied to T_{gg} , and then T_{gg} is compared with T_{hg} . When the high responses on true edge points are given, $80.19T_{gg}T_{hg}$ is chosen; when the low responses on non-edge points occur, $80.19T_{sd}$ is chosen. From the sub-expression cf_3 , high responses on non-edge points from T_{hg} are suppressed. In cf_4 , T_{gg} and T_{sd} are zoomed, and then their transformations are compared. From the characteristics of T_{gg} and T_{sd} , when the condition $18.66T_{gg} > 39.67T_{sd}$ is true, the relevant pixels might belong to noise or textures. The reason is that T_{gg} has strong responses and T_{sd} has weak responses in this condition (pixels as noise or textures). When T_{sd} has low values, most of the relevant pixels are from background. When T_{gg} has high values, most of the relevant pixels are from boundaries, but also could be textures. In cf_4 , the logical operation

might suppress textures and noise; since a large number will be obtained if the condition is true, and a small number will be given for a false condition. Note that cf_4 is in the denominator part and the basic features are normalised from 0 to 1. Equation (7.20) (cf_5) also helps to suppress the high responses on non-edge points from T_{hg} .

7.6 Variant Feature Construction

To extend this GP system to construct variant edge features, only edge information from horizontal directional edges (the direction of edges, not the horizontal derivative) are extracted to train GP programs, and these evolved GP programs give suitable responses on horizontal directional edges. Different from training data used for invariant feature construction, the training data only contains partial edge information (namely the horizontal edges, not all true edges in images) from desired outputs.

7.6.1 Training Data on Horizontal Edges

In order to obtain responses on horizontal edges from basic features, the Gaussian gradient and histogram gradient are given only from the horizontal edges. The image Gaussian derivative $T_{gd,0^\circ}$ (see Equation (7.3)) and histogram derivative $T_{hd,0^\circ}$ (see Equation (7.5)) are utilised. Since T_{sd} (see Equation (7.7)) is *invariant*, the third basic variant feature $T_{f,0^\circ}$ is from an F -test at the horizontal direction (0°). For each pixel, a horizontal line, located at the middle of a local (7×7) window, splits all pixels in the window into two groups. The F -test is employed to indicate the difference between the two groups.

For simplicity, image pixels are sampled with the same ratio of edge points and non-edge points as the training data. The horizontal edge points based on three straight connected edge points in the horizontal direction are marked as positive labels, and the others are negative la-

bels. Approximately 125 horizontal edge points, 125 non-horizontal edge points and 250 non-edge points are randomly sampled from each training image. Therefore, the training data includes approximately 100,000 $((125 + 125 + 250) * 200 \text{ images})$ cases and the three features at the horizontal direction. The window size for $T_{gd,0^\circ}$, $T_{f,0^\circ}$ and $T_{hd,0^\circ}$ is 7×7 .

Therefore, the terminal set for constructing variant features used $T_{gd,0^\circ}$, $T_{f,0^\circ}$ and $T_{hd,0^\circ}$ to replace T_{gg} , T_{sd} and T_{hg} . The other settings in the terminal and function sets stay the same as the settings for constructing invariant features. For distribution estimation, only $GP_{pdf, Norm}$ is selected. Since $GP_{pdf, Norm}$ has high contrast responses between edge points and non-edge points, the fitness function directly utilises the overall accuracy (see Equation (2.5) on page 35).

For the population size, it is increased to 500 because only training programs on horizontal edges is more difficult than training programs on all edge points. Here, an evolved program is required to only give strong responses on horizontal edges. Also, the maximum depth is increased to 8. The other settings are keep the same as the construction of invariant features.

7.6.2 Results From Variant GP Programs

Overall Results

Table 7.6 presents the mean and standard deviation of F_{max} values of the results from the combination (*square root of sum of squares*) of 30 features constructed by GP and their rotated versions $T_{GP,90^\circ}$, and F_{max} values from T_{gd} , T_f , T_{hd} . Here, an evolved program detects an image rotated by 90° , and then the result is rotated back as $T_{GP,90^\circ}$. Also, an estimated Bayesian model (using multivariate normal density to combine the three basic features and Equation (7.8)) [40] based on the sampling dataset is used to extract features at 0° for the test images when $T_{gd,0^\circ}$, $T_{f,0^\circ}$, $T_{hd,0^\circ}$ are considered as independent variables. T_f combines $T_{f,0^\circ}$ and $T_{f,90^\circ}$ (vertical

Table 7.6: Comparison of F_{max} values among constructed features by GP, image Gaussian derivatives T_{gd} , F -tests T_f , histogram derivatives T_{hd} , and a Bayesian model for the 100 BSD test images.

	F_{max}
GP	0.5776 ± 0.0015
T_{gd}	0.4737
T_f	0.5489
T_{hd}	0.542
Bayesian	0.5274

direction). The combination method for T_{gd} , T_f and T_{hd} chooses the maximum response on the horizontal and vertical directions. The training time for each constructed feature is around 11 hours (CPU 3.1 GHz), but the testing time is very short (several milliseconds). Note that the three basic features are pre-calculated and ready to use for testing.

The results from GP are significantly better than the others based on the one sample t -test with significance level 0.05. The test results show that the variant features constructed by GP significantly improve the detection performance based on the combination in the horizontal and vertical directions. However, the combination of the three features by the Bayesian model does not improve the detection performance. Only using the horizontal response information to estimate a Bayesian model, the estimated model has worse detection results than the results from T_f and T_{hd} . Therefore, we can see that GP is effective for automatic construction of variant features, only using one directional edge information.

From Table 7.6, the standard deviation of the detection results from GP is very low. Therefore, the evolved variant features have good stability to perform edge feature extraction, although they are trained based only on the horizontal edge information.

Figure 7.9: Details for recall and precision of T_{gd} , T_f , T_{hd} and GP (average).

Comparison among GP, T_{gd} , T_f and T_{hd}

Figure 7.9 shows the details for recall and precision with different threshold levels. Here, “@” is the position for the F_{max} . Compared with the detection from T_{gd} , T_f and T_{hd} , the curve for the averages of recall and precision of the detection results from GP is obviously better than the three basic curves from T_{gd} , T_f and T_{hd} . Therefore, the recall vs precision curves also show that GP can construct good variant features from fixed directional edge information. Based on Equation (7.8) ($T_{GP,\theta}$), the 30 features constructed by GP do not consider responses with low probabilities for discriminating pixels as edge points (p_+ is not larger than p_-), which is a reason that the curve for GP is short (only showing the points with both precision and recall higher than 0.5).

7.6.3 Detected Images

Figure 7.10 visually presents some detection results from GP, T_{gd} , T_f , T_{hd} and the estimated Bayesian model, where “GT” is ground truth. The four example images presented from the BSD test images include very different content. Firstly, it is found that the detected results from T_{gd} are affected

Figure 7.10: Four example detected images based on the different variant features.

by noise and textures, such as the responses on the wall texture in image 385039. Secondly, most of the detected results from T_f are correct, but some have weak responses on edges in low contrast areas, such as the

boundaries of the objects in images 62090 and 106024. Thirdly, histogram derivatives T_{hd} have stronger responses on edges in low contrast areas, but overreact to tiny discontinuities (because of overweighting discontinuities from local histograms), such as the water wave in image 101087.

The variant feature constructed by the Bayesian model improves responses on edges in low contrast areas, such as the boundary of the object in image 69020, but still has problems existing in the basic features, such as the responses on the walls in image 385039. However, the detected results from GP present very good detection performance on these images. The constructed feature enhances responses on edges in low contrast areas, such as the boundary of the object in image 69020. Also it suppresses noise and textures, such as the background in image 69020. From these detected images, the feature constructed by GP has the advantages from the basic features, and avoids some disadvantages from them. Therefore, these examples confirm the effectiveness of the GP method.

7.7 Further Discussions

This section discusses edge responses indicated by distribution estimation and a fixed threshold, how to suppress wrong responses from basic features, influence on the range of observations of evolved edge detectors, thickness of edges from GP edge detectors, and the construction equations GP_{pdf} and GP_{cdf} .

7.7.1 Fixed Threshold vs Estimated Distribution

Note that the composite features are constructed based on a known distribution. If a fixed threshold is used to *directly* discriminate the observations of a program, the test performance (F) with threshold 0 for the detected binary edge maps on the BSD test images is 0.4774 ± 0.0085 , which is significantly worse than the test performance on the features constructed by

the estimated distribution with the PDFs and CDFs. The averages of recall and precision with threshold 0 are 0.9013 and 0.3248, respectively. When the soft edge maps are normalised from the observations of the evolved programs, some of edge maps have very bad performance because the range of the relevant observations is sparse and huge. Therefore, the performance for their soft edge maps (lower than the performance on the binary edge maps) are not given.

Figure 7.11 shows two example images detected by the best evolved program directly using the fixed threshold (to discriminate pixels). From the detection results, the detected edges are very thick and the detection results are affected by noise and textures. A reason for the low performance of the evolved programs directly using threshold 0 is that there are no strict restrictions on the distribution of their observations. Since observations are only required to be larger or less than a threshold, the difference from the responses on different edge points and noise is hard to distinguish. The detected results have high recall and low precision. Note that F_{max} combines recall and precision, but F_{it} combines recall and specificity, which is a potential reason that GP with F_{it} and threshold 0 fails to construct good binary features. However, since the training data is balanced, the influence from the combinations of different indications should not be heavy. Martin, Fowlkes and Malik [123] indicated that an edge is one-dimensional, but the number of true non-edge points will grow as N^2 when the number of true edge points grows as N for an $N \times N$ area. It is possible that the specificity is not suitable to be used to combine with recall as a fitness function for evolving binary composite features when the test measure technique employs a (balanced) combination of recall and precision. Further investigation on this phenomenon will be conducted in the future.

When the distribution of the observations of an evolved program is used, the soft edge maps based on the probability of pixels belonging to edge points or non-edge points are easily obtained. The constructed fea-

Figure 7.11: Two example images detected by an evolved program directly using threshold 0 to classify pixels.

tures based on distribution of estimation are from 0 to 1, so it is avoided that the observations of an evolved program have a huge and sparse range.

When a known distribution requires observations to be as close to a position as possible, the soft edge maps based on the estimation have very high contrast between the edge points and non-edge points. This suggests that the PDF of the Gaussian distribution should be chosen if a desired composite feature for edge detection is required to have high contrast between edge points and non-edge points. Also, a distribution whose observations in each class crowd together could be used to obtain similar performance (the high contrast). However, a constructed feature based on the PDF of the Gaussian distribution may not indicate different responses on subjective edge points. In order to obtain rich responses on subjective edge points, a distribution with observations locating around a position (in a wide range) is suggested as an estimated PDF. The experiment result from the PDF of the triangular distribution shows the characteristic of rich responses. However, since a wide range is used, the overlap problem from both estimated PDFs (the classes edge point and non-edge point) exists, which needs to be addressed in the future. A distribution model from generalized normal distributions might be helpful to construct features with strong responses on clear boundaries, suitable responses on subjective edges and very weak responses on non-edge points.

Another way to obtain rich edge responses is to use a CDF to estimate the distribution of observations for each class. Since observations are al-

lowed to lay over a very wide range, the Gaussian and uniform distributions can give high responses on clear boundaries between different areas. The experiment results suggest that constructed features based on CDFs might not be remarkably affected by the type of a continuous distribution model. From the subjective characteristic of edge detection, features constructed by GP should choose CDFs to do distribution estimation, rather than choosing PDFs.

7.7.2 Suppression of Wrong Responses

In order to improve detection performance, the combinations of the basic features decrease high responses on non-edge points. From the analysis of the evolved program $CF_{PDF, Norm}$ in Section 7.5.7 (on page 264), a general approach to suppress wrong responses is found. The first step is to search for wrong responses from one local feature or multiple features. In order to find wrong responses, a comparison between different basic features is performed. If the characteristics of these basic features are known, responses on edge points, noise or textures will be easily found, such as a comparison between T_{hg} and T_{gg} (cf_3 , see Equation (7.18)). The second step is to keep accurate responses on one basic feature or a combination of basic features, and suppress the found wrong responses. To suppress the wrong responses, different methods can be used, such as replacing the high wrong response magnitudes with a low value, or choosing responses from another basic feature or a combination of basic features. Whether this method is effective to improve detection performance will be further investigated in the future.

In general, a suppression technique mainly focuses on one major feature, such as T_{gg} in the surround suppression [66]. However, the analysis of the evolved program $CF_{PDF, Norm}$ suggests that different combinations of basic features might be better to improve detection performance than a single feature. In order to check the difference from suppression on a sin-

gle major feature and a combination of basic features manipulated by the approach found from the evolved program, a further investigation will be conducted in the future.

7.7.3 Influence of the Range of Observations of Programs

A few constructed features from $GP_{PDF,Tria}$ have very bad performance. Via analysing the distributions of the observations of the evolved programs from $GP_{PDF,Tria}$, it is found that these features are from the evolved programs with large $\hat{\sigma}_-$ and $\hat{\sigma}_+$. Since the observations in a large and sparse range are not good to construct edge features [56], a large range of observations should be avoided in evolved features. In order to improve the bad features estimated based on the PDF, a restriction on the range of the observations of programs is proposed, and a new fitness function is defined in Equation (7.23). In the new fitness function Fit_{RS} , the restriction term RS (see Equation (7.22)) requires the sum of squares of the standard deviations $\hat{\sigma}_+$ and $\hat{\sigma}_-$ to be less than the space range parameter var_{max} . Also, the observations of an evolved program are required so that $\hat{\mu}_+$ should be larger than $\hat{\mu}_-$. Here, var_{max} is set to 50.

$$RS(\mu_+, \mu_-, \sigma_+, \sigma_-) = \begin{cases} 1 & \text{if } \hat{\mu}_+ > \hat{\mu}_- \text{ and } \hat{\sigma}_+^2 + \hat{\sigma}_-^2 < var_{max} \\ 0 & \text{otherwise} \end{cases} \quad (7.22)$$

$$Fit_{RS} = \frac{rs}{r+s} + \frac{1}{2}RS(\hat{\mu}_+, \hat{\mu}_-, \hat{\sigma}_+, \hat{\sigma}_-) \quad (7.23)$$

After using the restriction term RS , the test performance (F_{max}) of $GP_{PDF,Tria}$ is 0.5719 ± 0.0262 (mean $\pm$ standard deviation), which is significantly better than the results without using RS , according to a two-sample t -test with significance level 0.05. After using RS , the mean is increased and the standard deviation becomes smaller. Therefore, when $GP_{PDF,Tria}$ is employed to construct features, the restriction on the range of observations of evolved programs is recommended.

The best evolved program from $GP_{PDF,Tria}$ without using RS has $\hat{\mu}_- = -0.07$, $\hat{\sigma}_- = 126.19$, $\hat{\mu}_+ = 31.65$, and $\hat{\sigma}_+ = 8952.88$. The best evolved

program from $GP_{PDF,Tria}$ using RS has $\hat{\mu}_- = 0.02$, $\hat{\sigma}_- = 0.01$, $\hat{\mu}_+ = 0.03$, and $\hat{\sigma}_+ = 0.01$. From their estimated means and standard deviations, the overlap is reduced after using the restriction term RS . The term RS is effective to avoid the problem of the range of observations being sparse and huge.

Figure 7.12 shows two example images detected by the best features from $GP_{PDF,Tria}$ with fitness functions Fit and Fit_{RS} . The detected results from fitness function Fit_{RS} have thick responses on edges from the human view. The reason is that $p_+(o)$ from the evolved program with Fit_{RS} slightly decreases when a discriminated pixel on a true edge moves to a position nearby the true edge. Since the restriction term RS limits the range of the observations of an evolved program, the change of the observation from a pixel on a true edge point to a pixel nearby the true edge is not obvious. The grayscale levels for representing these responses are high, so the detected edges look thick, although their grayscale levels are different.

The detected images from the fitness function Fit_{RS} have weaker responses on non-edge points than the detection results from the fitness function Fit . When the observations of an evolved program have a small range, the overlap for both classes is not large.

However, even though the ranges of observations of some evolved programs with CDFs are large (such as $\hat{\sigma}_-$ of one evolved program larger than 10^{12}), the performance of $GP_{CDF,Norm}$ and $GP_{CDF,Unif}$ are very robust. It seems that the range of the observations of evolved programs with CDFs does not obviously affect the test performance (F_{max}). A reason is that the overlap in CDFs is smaller than PDFs when the same estimated parameters are used, and an observation far away from their means is distinguishable to give very high probability for one class and very low probability for the other. Therefore, from the robustness view, CDF, rather than PDF, is suggested for feature construction in GP.

Note that $GP_{PDF,Tria}$ still has the problem of the range of observations

Figure 7.12: Two example images detected by evolved programs from $GP_{PDF,Tria}$ with fitness functions Fit and Fit_{RS} .

after replacing 3σ in Equation (7.12) (on page 250) with $\sqrt{6}\sigma$. The test performance using $\sqrt{6}\sigma$ and without the restriction is 0.5363 ± 0.1245 , which is lower than $GP_{PDF,Tria}$ using 3σ .

7.7.4 Thickness of Edges

The detected edges from $GP_{PDF,Norm}$ are thick. A potential reason is that the pixels nearby the true edge points have a vector value of the three basic features which are close to the estimated $\hat{\mu}_+$ (of the three basic features). The probability for marking these pixels as edge points is high, and it is very close to the probability for the pixels on the true edges. From the 256 graylevels, they are distinguished hardly from human observations, so the detected edges look very thick. However, $GP_{PDF,Tria}$, $GP_{CDF,Norm}$ and $GP_{CDF,Unif}$ estimate the probability for a pixel as an edge point in a suitably wide range, which makes the responses on pixels nearby edges and pixels on edges distinguishable.

7.7.5 Construction Equations GP_{pdf} and GP_{cdf}

GP_{pdf} employs a density function and Bayesian inversion to construct edge features, but GP_{cdf} directly employs a fitted CDF to construct edge feature. Whether Equations GP_{pdf} and GP_{cdf} can be replaced by each other needs to be investigated. In order to compare Equations GP_{pdf} and GP_{cdf} , GP_{pdf} is applied to the estimation using CDFs and GP_{cdf} is applied to the estimation using PDFs.

The construction equation using PDFs (GP_{pdf} , see Equation (7.8)) is different from the equation using CDFs (GP_{cdf} , see Equation (7.9)). The difference of the probability of the class edge point ($p_+(o)$) and the probability of the class non-edge point $p_-(o)$ is not suitably used in the estimation with PDFs. From Figure 7.3 (a), the difference of both probabilities for an observation at the position “A” is almost to 0, so it wrongly describes a response on a true edge point. When GP_{cdf} is used to indicate responses for the estimated PDFs, some constructed features based on the normal distribution will have no responses on true edge points. Although some observations have $p_+(o)$ slightly larger than $p_-(o)$, the difference of $p_+(o)$ and $p_-(o)$ might be mapped to 0 in the 256 grayscale levels range, such as “A”. The probabilities for edge points and non-edge points are not very close in the estimated triangular distributions. The performance for using GP_{cdf} in GP with the triangular distributions also decreases, but it is not very obvious. The test performance of $GP_{PDF, Norm}$ (replacing GP_{pdf} with GP_{cdf}) is 0.5069 ± 0.1390 , and the test performance of $GP_{PDF, Tria}$ (replacing GP_{pdf} with GP_{cdf}) is 0.5302 ± 0.0913 . Clearly, the standard deviation from $GP_{PDF, Norm}$ become larger after using the difference of $p_+(o)$ and $p_-(o)$, and the performance is decreased obviously. However, the performance of $GP_{PDF, Tria}$ after replacing GP_{pdf} with GP_{cdf} is not decreased sharply. This suggests that for an estimated distribution, including all observations crowding in a centre point, it is not good to use the difference of the probabilities for the two classes to describe edge responses.

When GP_{pdf} is employed to construct feature by GP with CDF estima-

tion, the performance for GP using the Gaussian distribution is 0.5787 ± 0.0216 , which is slightly lower than $GP_{CDF, Norm}$ using GP_{cdf} . The performance for GP using CDFs of the uniform distribution and GP_{pdf} is 0.5719 ± 0.02593 , which is slightly lower than $GP_{CDF, Unif}$ using GP_{cdf} .

Although the results from GP with CDF estimation and GP_{pdf} have no significant difference from the results from GP with the same estimation and GP_{cdf} , the detected images from the subjective view are different. Figure 7.13 shows two example detected images from GP with CDF estimation and GP_{pdf} . From the visual results, the responses on edges are thick. It is found that GP using GP_{pdf} has the problem of thick responses. From the probabilistic description, pixels nearby a true edge have high probabilities to be described as edge points and not very low probabilities to be described as non-edge points. Pixels on the edges have slightly higher probabilities to be described as edge points than these pixels nearby the true edge, and have lower probabilities to be described as non-edge points than the latter. Using GP_{pdf} , the response values from the pixels nearby the true edge are similar to the response values from pixels on the edges. If using difference of the probabilities for describing edge points and non-edge points, the pixels nearby the true edge reduce their responses, compared to the pixels on the true edge. Therefore, GP_{cdf} indicates edge responses which are thinner than the responses from GP_{pdf} . Also, the rich responses on different edges are not obviously described in these detected image from GP with CDF and GP_{pdf} .

To summarise, the construction equations affect the test performance of the features constructed by GP with the estimation from the PDFs of the triangular distribution, but they do not affect the test performance of the features constructed by GP with the estimation from the CDFs of the Gaussian distribution or the uniform distribution, in terms of F_{max} . However, from the subjective view, it is suggested that the difference of the probabilities for describing edge points and non-edge points should be chosen when CDF estimation is utilised.

Figure 7.13: Two example detected images from GP with CDF and GP_{pdf} .

7.8 Chapter Summary

The goal of this chapter was to investigate the distribution of the observations of a GP program to construct edge features for edge detection. Via using a piecewise linear relationship and a non-linear relationship for PDFs and CDFs to estimate the observations of the evolved programs, the composite invariant features were constructed by GP. A benchmark image dataset (the BSD image dataset) was employed for this study. Also, the extension from constructing invariant edge features to variant edge features has been investigated.

The experiment results show that the invariant (or variant) features constructed by GP are better than the three basic features, i.e., the image Gaussian gradient (or derivative), the local normalised standard deviation (or the feature extracted by the F -test technique), and the image local histogram gradient (or derivative). The constructed features from GP combine some advantages from the three basic features, and appear to reduce their disadvantages. The features constructed by GP based on the estimation using CDFs (of the Gaussian distribution and the uniform distribution) are better than the combinations from the Bayesian model using the general multivariate normal density and the linear SVM. The experiment

results show that GP has some ability to find a way of efficiently combining different features together and describe rich responses on different edges.

From the experiment results, the constructed features from the estimation on the PDF with a linear relationship (the triangular distribution) is worse than the estimation on the PDF with a non-linear relationship (the Gaussian distribution), and the estimations on the CDFs with a relationship function (the uniform distribution) and a non-linear relationship (the Gaussian distribution). From the visually detection results, it is suggested that the CDFs (of the Gaussian distribution and the uniform distribution) are better than the PDFs (of the Gaussian distribution and the triangular distribution) to obtain rich responses on boundaries between different areas and good accuracy on non-edge points. Besides, GP_{cdf} is better than GP_{pdf} to represent rich and thin edge responses, but GP_{cdf} failed to represent edge responses of the features constructed by GP using PDFs.

Also, an evolved program was reasonably explained. From the analysis of the evolved program, the approach found from the evolved program appears to suppress wrong responses on non-edge points.

In addition, the range of observations of the programs evolved by GP using the PDF of triangular distribution might decrease the detection performance when the range is large. A evolved program with a small range of observations should be preferred.

This chapter focused on investigating how to combine basic features. The combination operators (functions) are standard and without prior domain knowledge. In the next chapter, combination techniques using some prior domain knowledge will be developed by GP for the performance improvement in automatic feature construction.

Chapter 8

High-level Feature Construction

8.1 Introduction

Chapter 7 shows that GP has successfully been applied to feature construction by combining basic features and estimating the distribution of the observations of evolved programs. The basic edge features come from different prior domain knowledge (the Gaussian gradient T_{gg} , normalised standard deviation T_{sd} , and histogram gradient T_{hg}). The constructed GP edge features were significantly better than the combination from the simple Bayesian model using a general multivariate normal density [40]. How to improve the performance of the combination from the Bayesian model should be investigated.

From machine learning [40], different methods, such as Bayesian-based techniques, can be applied for combining basic features. Bayesian-based techniques have been widely applied to classification [82]. Since these techniques are based on applying Bayes' theorem [40], human experts can understand the structures of the models used in the Bayesian-based techniques. However, there are still existing issues in the Bayesian-based techniques, such as how to effectively select features and design suitable structures [82]. It is therefore desirable to develop a GP system to automatically select features and design structures for performance improvement

of a Bayesian-based technique.

8.1.1 Chapter Goal

The goal of this chapter is to investigate automatic high-level feature construction for edge detection using GP and Bayes' theorem. Here, a high-level feature means that it is constructed based on some basic features and combination techniques with prior domain knowledge. In order to build up a Bayesian-based GP system, the simple Bayesian model using a general multivariate normal density is employed here. The simple Bayesian model is proposed as a function and/or a terminal to construct different Bayesian-based programs. Specifically, the following research objectives will be investigated.

- Whether the evolved Bayesian-based programs are better than the simple Bayesian model for constructing edge features.
- Whether there are differences between using all basic features and randomly selecting a set of features in a Bayesian-based function.
- Whether using the simple Bayesian model as a terminal can achieve better performance than using the Bayesian model as a function.
- Whether the evolved Bayesian-based programs can be reasonably interpreted.

8.1.2 Chapter Organisation

In the remainder of this chapter, the second section proposes the Bayesian-based GP system to construct high-level features. The third section gives the experiment settings. The fourth section presents the results of the experiments with discussions. The fifth section provides further discussions. The last section draws a summary of this chapter.

8.2 Bayesian-based GP System

In order to employ existing techniques to combine basic features into composite features, functions or terminals based on existing techniques are designed in the proposed GP system. The function set includes functions implementing a Bayesian model and general algebraic operators. The terminal set includes not only basic features, but also a simple Bayesian model.

8.2.1 Bayesian-based Function and Terminal

Edge detection is considered here as a binary classification problem. Let $j = 0$ be the class non-edge point, $j = 1$ be the class edge point, and x be a d -component vector-valued random variable. P_j is the prior probability for the class non-edge point or edge point. Let $p_{x|j}$ be the state-conditional probability density function for x (given $j = 0$ or 1 being the true state). The posterior probability $p_{j|x}$ can be estimated from $p_{x|j}$ by Bayesian inversion (Equation (8.1)) [40]. Here the conditional probability density function $p_{x|j}$ is estimated by the general multivariate normal density (see Equation (8.2)), where, $\hat{\mu}_j$ is the d -component sample mean vector for class j , $\hat{\Sigma}_j$ is the d -by- d sample covariance matrix, $|\hat{\Sigma}_j|$ is its determinant and $\hat{\Sigma}_j^{-1}$ is its inverse, and $(x - \hat{\mu}_j)^T$ is the transpose of $x - \hat{\mu}_j$. Note that $|\hat{\Sigma}_j|$ might be 0. To avoid this situation, each diagonal element in $\hat{\Sigma}_j$ has a tiny tolerance value $\epsilon = 1.0e - 12$ added to it.

$$p_{j|x} = \frac{p_{x|j}P_j}{\sum_{k=0}^1 p_{x|k}P_k} \quad (8.1)$$

$$p_{x|j} = \frac{1}{(2\pi)^{\frac{d}{2}} |\hat{\Sigma}_j|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \hat{\mu}_j)^T \hat{\Sigma}_j^{-1} (x - \hat{\mu}_j) \right) \quad (8.2)$$

In order to improve the performance of the standard Bayesian model, a Bayesian-based function is defined in Equation (8.3). Here, s is the function argument, and x is one of the possible combinations from the basic features T_{gg} , T_{sd} , and T_{hg} . The function argument s is one basic feature or a subtree. Since the class edge point is the major class in edge detection, the

output is a one-dimensional variable based on the posterior probability of the class edge point. In edge detection, multiple outputs are not necessary, but the function can support multiple outputs, which is a potential future application to multiple classification problems.

$$b_{1|x}(s) = \frac{p_{1|x,s}}{\sum_{j=0}^1 p_{j|x,s}} \quad (8.3)$$

The range of the output for the function $b_{1|x}(s)$ is from 0 to 1, and a composite feature from the simple Bayesian model [40] can be described as $b_{1|x_1}(T_{hg})$, where $x_1 = \{T_{gg}, T_{sd}\}$. Note that the argument s of the Bayesian-based function $b_{1|x}$ can be the output of another $b_{1|x}$, such as $b_{1|x}(b_{1|x}(s))$.

Therefore, the Bayesian-based function has three behaviours: (1) selecting basic features; (2) combining basic features and s (possibly as an intermediate combination); and (3) giving edge responses after estimating the relevant multivariate normal density.

In order to check whether the simple Bayesian model as a terminal is good to combine basic edge features, $b_{1|x_1}(T_{hg})$ is also considered as a terminal. Here x_{bayes} indicates $b_{1|x_1}(T_{hg})$ used as a terminal. Since x_{bayes} combines different basic edge features, the GP system allows a tree only consisting of a single terminal, e.g., x_{bayes} (the simple Bayesian model) as a complete program.

General Algebraic Operators

In order to enrich the evolved Bayesian-based programs, ordinary algebraic operators need to be added into the function set for constructing composite features. However, the constructed composite features are expected to represent soft edge maps. A sparse and large range of the values of a constructed feature is not suitable to represent soft edge maps. In Chapter 4, this problem has been addressed. Also, in Chapter 7, directly using observations of evolved programs with normal algebraic operators was found to not be good to construct soft edge maps. The transformation

by a function used in the Bayesian-based GP system requires the mapped values to be located in a suitable range. In general, a feature, which is used to represent soft edge maps, is normalised from 0 to 1. In the simple Bayesian model, the output of a Bayesian-based function or x_{bayes} is from 0 to 1. Therefore, it is useful if all operations in the Bayesian-based GP system are required to map their inputs into the same space.

For linear operations, two general algebraic operators $\{\oplus, \ominus\}$ are added into the function set. The operation of $\oplus$ is defined in Equation (8.4), and the operation of $\ominus$ is defined in Equation (8.5). Here, s_a and s_b are the arguments of the operators $\oplus$ and $\ominus$, and the operator $\ominus$ is the absolute value of difference of s_a and s_b . The ranges of s_a and s_b are from 0 to 1, so the range of the outputs of both operators is also from 0 to 1.

$$s_a \oplus s_b = \frac{s_a + s_b}{2} \quad (8.4)$$

$$s_a \ominus s_b = |s_a - s_b| \quad (8.5)$$

Non-linear general algebraic operators could be used in the function set. Since this chapter only focuses on the Bayesian model, the application of using different general algebraic operators will be investigated in the future work.

8.2.2 The Other Terminals

The basic features are considered as terminals and they are normalised from 0 to 1. Note that random constants are not included in the terminal set because a Bayesian-based program estimates the relevant multivariate density to construct a composite feature, rather than directly using its observations as a feature. Also, the outputs for all operations in the GP system are located in the range from 0 to 1. The operator $\oplus$ will automatically adjust the scales for its input (such as Equation (8.4)) so that its output is still located in the range from 0 to 1.

8.2.3 Fitness Function

From the set of terminals and functions, the range of outputs of a program is from 0 to 1, which is directly considered as the value of a composite feature. The aim of the new constructed features is to detect as many true edge points as possible, so the fitness function Fit (see Equation (8.6), more details on page 248) is used in here. Recall p_{rec} and specificity p_{spe} are calculated when a threshold is used to discriminate outputs of a program as edge points or non-edge points.

$$Fit = \frac{2p_{rec}p_{spe}}{p_{rec} + p_{spe}} \quad (8.6)$$

Since there are no random constants in the terminal set and the range of observations of evolved programs is restricted from 0 to 1, multiple thresholds are used to find the maximum of Fit . Here, only three thresholds are employed to find the maximum value of Fit as the fitness for the relevant program, aiming at evolving programs with high contrast edge responses. The three thresholds are 0.25, 0.5 and 0.75. In order to check whether three thresholds are enough for evaluating evolved programs, 30 thresholds ($\frac{i}{31}, i = 1, 2, \dots, 30$) are also used to find maximum Fit . To distinguish the two different settings on Fit , $Fit_{t=3}$ indicates Fit with the three thresholds, and $Fit_{t=30}$ indicates Fit with the 30 thresholds.

8.3 Experiment Settings

The training dataset is the same as in Section 7.4.1 (page 249). Here, x_{all} indicates that all basic features (T_{gg} , T_{sd} and T_{hg}) are used in the terminal set. Also, $b_{1|x_{all}}$ indicates that the Bayesian-based function always includes all basic features, and $b_{1|x}$ indicates that the Bayesian-based function randomly selects one of all combinations of the basic features. In order to investigate the influence of different terminals and functions, different settings for the terminal and function sets are listed in Table 8.1.

Table 8.1: Settings for terminals and functions in the Bayesian-based GP system.

Setting		Terminals	Functions
Bayesian	$Set_{bf,rand}$	$\{x\}$	$\{b_{1 x}, \oplus, \ominus\}$
Function	$Set_{bf,all}$	$\{x\}$	$\{b_{1 x_{all}}, \oplus, \ominus\}$
Bayesian Terminal	Set_{bt}	$\{x, x_{bayes}\}$	$\{\oplus, \ominus\}$
	$Set_{bt,all}$	$\{x, x_{bayes}\}$	$\{b_{1 x_{all}}, \oplus, \ominus\}$
	$Set_{bt,rand}$	$\{x, x_{bayes}\}$	$\{b_{1 x}, \oplus, \ominus\}$
Full Set	Set_{full}	$\{x, x_{bayes}\}$	$\{b_{1 x}, b_{1 x_{all}}, \oplus, \ominus\}$

Firstly, the Bayesian-based functions are investigated. Setting $Set_{bf,rand}$ automatically selects basic features to construct composite features. Setting $Set_{bf,all}$ restricts that all basic features must be included in a Bayesian-based function. The purpose for different settings on Bayesian-based functions is to find the difference between a large space (using $b_{1|x}$ to include possible combinations of basic features) and a narrow space (must use all basic features) for feature construction. Secondly, three settings Set_{bt} , $Set_{bt,all}$ and $Set_{bt,rand}$ are used to investigate the relationship between the terminal x_{bayes} and the functions $b_{1|x}$ and $b_{1|x_{all}}$. Lastly, a full set of terminals and functions is given in setting Set_{full} .

The parameter values for GP are: population size 50; maximum generations 50; maximum depth (of a program) 5, probabilities for mutation 0.15, crossover 0.80 and elitism (reproduction) 0.05. These values are chosen based on common settings and initial experiments [53]. Note that a Bayesian-based program has some ability to combine the basic features after estimating sample means and standard distributions in the program. A larger population and a larger number of generations are not necessarily required.

Each experiment is repeated 30 independent runs. The test perfor-

mance employs F_{max} (see Section 7.4.4 on page 251).

8.4 Results and Discussions

This section provides experiment results with discussions. The results from $Fit_{t=3}$ will be given, then the results are compared with existing techniques. A comparison between $Fit_{t=3}$ (three thresholds) and $Fit_{t=30}$ (30 thresholds) will be conducted. After showing detected images from $Fit_{t=3}$ and $Fit_{t=30}$, an example evolved Bayesian-based program will be interpreted.

8.4.1 Overall Results From $Fit_{t=3}$

Table 8.2 gives the means, standard deviations (s.d.), the maximum (“Max”), and the minimum (“Min”) of F_{max} values of the evolved programs for the six settings when fitness function $Fit_{t=3}$ is used. The “ t -test” column reports the p -values obtained from the comparisons between the relevant results (first column as the first group) and the results from $Set_{bf,rand}$ (as the second group) by using two-sample t -tests; and “MWW” reports the p -values obtained from the comparisons between the relevant results and the results from $Set_{bf,rand}$ by using Mann-Whitney-Wilcoxon tests [102].

From an overall view, setting $Set_{bf,rand}$ has the highest mean and the maximum F_{max} . It seems that setting $Set_{bf,rand}$ has the best test performance on the BSD test images. In regards of the p -values from the t -tests, the results from $Set_{bf,rand}$ are significantly better than the results from the other settings, except for Set_{full} . From the Mann-Whitney-Wilcoxon tests, the results from $Set_{bf,rand}$ are also significantly better than the results from $Set_{bf,all}$, Set_{bt} and $Set_{bt,all}$. Since only settings $Set_{bf,rand}$, Set_{full} and $Set_{bt,rand}$ include $b_{1|x}$, it seems that $b_{1|x}$ is important to construct high-level features for performance improvement when fitness function $Fit_{t=3}$ is used.

Table 8.2: Test performance F_{max} for the six settings using fitness function $Fit_{t=3}$ on the BSD test image dataset.

Setting	Mean $\pm$ s.d.	Max	Min	t -test	MWW
$Set_{bf,rand}$	0.5591 ± 0.0150	0.5847	0.5199		
$Set_{bf,all}$	0.5423 ± 0.0053	0.5504	0.5349	0.0000 $\downarrow$	0.0000 $\downarrow$
Set_{bt}	0.5391 ± 0.0000	0.5391	0.5391	0.0000 $\downarrow$	0.0000 $\downarrow$
$Set_{bt,all}$	0.5381 ± 0.0137	0.5664	0.5102	0.0000 $\downarrow$	0.0000 $\downarrow$
$Set_{bt,rand}$	0.5513 ± 0.0107	0.5733	0.5277	0.0278 $\downarrow$	0.0543
Set_{full}	0.5513 ± 0.0181	0.5841	0.5102	0.0798	0.0603

Figure 8.1: Boxplots for the F_{max} values of the results (30 replications) from the six settings using fitness function $Fit_{t=3}$ on the 100 BSD test images.

Figure 8.1 reveals the F_{max} values of the results from the six settings using $Fit_{t=3}$. Here the box labels are the indices of the six settings, and each box represents the relevant setting. From these boxplots, the results from Set_{bt} are the same. From Figure 8.1 and Table 8.2, a few evolved programs from settings $Set_{bf,rand}$, $Set_{bt,all}$ and Set_{full} have very bad test performance, and their F_{max} values are less than 0.52. From an overall view, all settings, except for Set_{bt} , have at least half of their evolved programs with F_{max} val-

ues larger than 0.54. From Table 7.2 in Section 7.5.2 (page 256), the highest F_{max} among the three basic features is 0.5434. Table 7.2 also showed a simple Bayesian method with $F_{max} = 0.5302$, which is obviously lower than most of the evolved programs from the six settings here, except for Set_{bt} . Therefore, GP can effectively evolve Bayesian-based programs to improve detection performance when fitness function Fit only uses the three thresholds to find the maximum value.

Bayesian-based Function

From the comparison between $Set_{bf,rand}$ (including $b_{1|x}$) and $Set_{bf,all}$ (including $b_{1|x_{all}}$), the latter is significantly worse. If using $Fit_{t=3}$, the Bayesian program including all three basic features is not good to find good Bayesian-based programs. From the boxplots in Figure 8.1, the test performances of the evolved programs from $Set_{bf,all}$ are located in a narrow range. However, the test performance of the evolved programs from $Set_{bf,rand}$ spreads over a larger range. A reason is that function $b_{1|x_{all}}$ always includes the three basic features. After estimating an evolved program including $b_{1|x_{all}}$, the test performance is strongly affected by the combination of three basic features (as a Bayesian model). Note that fitness function $Fit_{t=3}$ only uses three thresholds to find the maximum value, and a Bayesian model with three basic features has high contrast responses for edge points. However, function $b_{1|x}$ randomly selects basic features, it is possible that only one basic feature is selected and the evolved program has high test performance. Since the combination of three basic features using $b_{1|x}$ are varied and the evolved programs from using $b_{1|x_{all}}$ always include all basic features, an evolved program from using function $b_{1|x}$ may have lower F_{max} than the evolved programs from using function $b_{1|x_{all}}$. From the minimum value of test performance F_{max} in Table 8.2 and Figure 8.1, a bad evolved program exists in setting $Set_{bf,rand}$, but there are no outliers in setting $Set_{bf,all}$.

Therefore, when Fit only uses the three thresholds to find the max-

imum value as the fitness of an evolved program, the evolved program with $b_{1|x_{all}}$ is strongly connected to the combination of all basic features and the test performance is located in a stable range; but evolved programs with $b_{1|x}$ have flexible structures to combine basic features, which brings good test performance on some evolved programs in most cases, at the same time, leads to bad test performance on a few evolved programs.

Bayesian-based Terminal

When x_{bayes} is added into the terminal set, all evolved programs from Set_{bt} using $Fit_{t=3}$ are equal to the single node tree x_{bayes} . It seems that $Fit_{t=3}$ could not find any better combination from x_{bayes} and the three basic features without using Bayesian-based functions, compared with x_{bayes} . Note that the test performance F_{max} on x_{bayes} is a bit higher than the simple Bayesian model in Table 7.2 in Section 7.5.2. The change is caused by x_{bayes} adding tolerance ϵ in $\hat{\Sigma}_j$.

After adding Bayesian-based functions, the results from $Set_{bt,all}$ are similar to the results from $Set_{bf,all}$, and the results from $Set_{bt,rand}$ are similar to the results from $Set_{bf,rand}$. It is surprising that there are two evolved programs from $Set_{bf,all}$ with F_{max} values less than 0.52. In $Set_{bf,all}$, $b_{1|x_{all}}$ and x_{bayes} include the three basic features, but there are still two programs with bad test performance. From the boxplots in Figure 8.1, the terminal x_{bayes} might be not good for constructing high-level features when fitness function $Fit_{t=3}$ is used. This suggests that directly combining the simple Bayesian model with all basic features might be not good for performance improvement.

Combination

When all functions and terminals are added into the GP system, namely using Set_{full} , the results from Set_{full} are not significantly different from the results from $Set_{bf,rand}$. It seems that adding x_{bayes} and $b_{1|x_{all}}$ in setting

Table 8.3: Comparison between the results from $Set_{bf,rand}$ using $Fit_{t=3}$, image Gaussian gradients T_{gg} , normalised standard deviations T_{sd} , histogram gradients T_{hg} , Sobel edge detector and the simple Bayesian terminal x_{bayes} on the 100 BSD test images.

	F_{max}	p -value
$Set_{bf,rand}$	0.5591 ± 0.0150	
T_{gg}	0.5153	0.000 ↓
T_{sd}	0.4968	0.000 ↓
T_{hg}	0.5434	0.000 ↓
Sobel	0.4832	0.000 ↓
x_{bayes}	0.5391	0.000 ↓

$Set_{bf,rand}$ does not affect the evolved programs when $Fit_{t=3}$ is used. Although the search space for candidate solutions from Set_{full} is larger than the space from $Set_{bf,rand}$, GP can still effectively discover good programs to construct high-level features.

8.4.2 $Set_{bf,rand}$ using $Fit_{t=3}$ vs Existing Techniques

Table 8.3 presents the comparisons between the results from $Set_{bf,rand}$ using $Fit_{t=3}$ and the results from T_{gg} , T_{sd} , T_{hg} , the Sobel edge detector, and x_{bayes} . The p -values are obtained from the comparison between the relevant results (first row as the first group) and the results from $Set_{bf,rand}$ using $Fit_{t=3}$ (as the second group) when one-sample t -tests are used. In addition, the 95% confidence interval (based on the t -test) for the evolved programs from $Set_{bf,rand}$ is (0.5533, 0.5648). These results show that the features constructed by GP significantly improve the detection performance. However, the F_{max} value from x_{bayes} is lower than the basic feature T_{hg} , so it seems that the simple combination does not improve the detection performance (F_{max}). Therefore, GP is effective for automatic construction of

Figure 8.2: F_{max} values for the features constructed by the evolved 30 programs from GP using $Set_{bf,rand}$ and $Fit_{t=3}$.

new features, and improves the performance of using the Bayesian model to combine basic edge features for edge detection.

In order to check the details of the performance on all evolved programs, Figure 8.2 gives the relevant F_{max} values for all 30 evolved Bayesian-based programs from $Set_{bf,rand}$ with $Fit_{t=3}$. Most of the evolved Bayesian-based programs have higher F_{max} than the best basic feature T_{hg} , and most of the evolved programs obtain good performance. However, the constructed features from three evolved programs are worse than T_{hg} (with F_{max}). The lowest F_{max} value of the constructed feature from the worst evolved program is only 0.52. How to improve the performance on the worst evolved program will be a future work.

Recall and Precision of an Evolved Program

Figure 8.3 shows different values of recall and precision for T_{gg} , T_{sd} , T_{hg} , x_{bayes} , and a program evolved by GP (with $F_{max} = 0.5847$). Here, “@” is the position for the F_{max} . From the different thresholds, it is clear that the evolved program has higher precision than the others when recall is not too low. For T_{gg} , recall is high but precision is too low, and precision is

Figure 8.3: Recall and precision for T_{gg} , T_{sd} , T_{hg} , x_{bayes} , and an evolved program.

high but recall is too low.

One interesting observation from Figure 8.3 is that x_{bayes} has a narrow range for recall and precision. A reason for this phenomenon is that the Bayesian Equation (8.1) gives very low probabilities for most non-edge points and very high probabilities for most edge points. Therefore, the change of the threshold does not strongly affect the detection performance. Compared with the basic features, x_{bayes} and the evolved program have very high recall values for many thresholds. It seems that the composite features from the Bayesian model can easily discriminate pixels as edge points or non-edge points with a set of thresholds. This is similar to the characteristics of edge responses (high contrast) on the composite features constructed by using Gaussian distribution estimation. Whether the multivariate density using other distributions can be used to obtain rich edge responses will be investigated in the future.

Table 8.4: Test performance F_{max} for the six settings using $Fit_{t=30}$ on the BSD test image dataset.

Setting	Mean $\pm$ s.d.	Max	Min	t -test	MWW	$Fit_{t=3}$
$Set_{bf,rand}$	0.5754 ± 0.0134	0.5968	0.5509			0.0000 $\uparrow$
$Set_{bf,all}$	0.5707 ± 0.0104	0.5927	0.5531	0.1397	0.0726	0.0000 $\uparrow$
Set_{bt}	0.5706 ± 0.0198	0.5966	0.5087	0.2824	0.2697	0.0000 $\uparrow$
$Set_{bt,all}$	0.5706 ± 0.0166	0.5950	0.5250	0.2278	0.4705	0.0000 $\uparrow$
$Set_{bt,rand}$	0.5656 ± 0.0230	0.5946	0.4998	0.0518	0.1290	0.0053 $\uparrow$
Set_{full}	0.5684 ± 0.0159	0.5939	0.5087	0.0743	0.2697	0.0006 $\uparrow$

8.4.3 Fitness Functions $Fit_{t=3}$ vs $Fit_{t=30}$

Table 8.4 gives the means, standard deviations, maximum and minimum of F_{max} values for the six settings using $Fit_{t=30}$. Here, “ t -test” indicates p -values obtained from the comparisons between the relevant results (setting in the first column as the first group) and the results from $Set_{bf,rand}$ using two-sample t -tests, MWW indicates p -values from their comparisons with the Mann-Whitney-Wilcoxon tests, and “ $Fit_{t=3}$ ” indicates p -values obtained from the comparisons between the results from $Fit_{t=30}$ and $Fit_{t=3}$ (using the same setting) with paired-sample t -tests. The paired-sample t -tests are used for the comparisons between $Fit_{t=30}$ and $Fit_{t=3}$ because of the same initial population.

There are some interesting observations from Table 8.4. Firstly, the test results from $Fit_{t=30}$ are significantly better than the results from $Fit_{t=3}$. It seems that the evaluation based on three thresholds is not good to find good features (in terms of F_{max}), although Bayesian-based functions or terminals give high contrast responses. A potential reason is that Bayesian-based outputs are a combination of basic features with the two general algebraic operators. Since combining the three basic features needs multiple thresholds to find the maximum Fit values, a program, including

the combination of Bayesian-based subtrees and subtrees constructed by basic features and the two general algebraic operators, might give rich responses on edge points and non-edge points. After using 30 thresholds to find the maximum of Fit as fitness for evolved programs, subtrees constructed by basic features and the two general algebraic operators might strongly affect the fitness of the relevant program.

Secondly, when $Fit_{t=30}$ is used, there are no significant differences between the results from $Set_{bf,rand}$ and the other settings. Using multiple thresholds is easier to find a threshold which is closer to the optimal threshold for Fit . Some evolved programs evaluated by $Fit_{t=3}$ are not good, but they have good binary outputs by another threshold, not including in the thresholds used in $Fit_{t=3}$. Since $Fit_{t=30}$ is more reasonable than $Fit_{t=3}$ to evaluate programs' fitness, all settings have good results. From the t -tests and the Mann-Whitney-Wilcoxon tests, there is no influence from using the Bayesian-based node in the function set or the terminal set. The way of selecting the full set of basic features or randomly selecting basic features does not obviously affect the test performance of the evolved problems.

Thirdly, the maximum F_{max} in each setting is increased. It seems that the ability to find good programs to construct high-level features is improved after using $Fit_{t=30}$.

Lastly, the worst evolved programs from settings $Set_{bf,rand}$ and $Set_{bf,all}$ are improved obviously when $Fit_{t=30}$ replaces $Fit_{t=3}$, but the worst evolved programs from settings Set_{bt} , $Set_{bt,rand}$ and Set_{full} become even worse, in terms of F_{max} . It is interesting that multiple thresholds do not improve the test performance of the worst evolved programs for the three settings. From each setting including x_{bayes} , the worst test performance from their results is lower than the test performance on each of the basic features. Therefore, the simple Bayesian model with all basic features might be not good to use as a terminal, and it is suggested that the Bayesian combination technique should only be used as a function to combine basic features.

Figure 8.4: Two example images detected by T_{gg} , T_{hg} , T_{sd} , x_{bayes} , and the best evolved programs from GP with $Set_{bf,rand}$ using $Fit_{t=3}$ and $Fit_{t=30}$.

8.4.4 Detected Images

Figure 8.4 shows two example detected images from the three basic features, and the best evolved programs from $Fit_{t=3}$ and $Fit_{t=30}$ with setting $Set_{bf,rand}$. From the detected images, the detected images by x_{bayes} and the evolved programs have obviously strong responses on true edge points, which is similar to T_{hg} . The strong responses on non-edge points from T_{hg} are mostly suppressed by x_{bayes} and the evolved programs. However,

some non-edge points with high responses from T_{hg} still have high responses in x_{bayes} . Comparing the responses from T_{sd} and T_{gg} , most of the non-edge points with high responses from T_{hg} in these two images are obviously decreased in the GP evolved programs. Therefore, the Bayesian-based GP system effectively constructs new composite features.

From the visual results detected by the program from $Fit_{t=3}$, the strong responses on non-edge points from discontinuous background areas (such as T_{hg}) are decreased. Comparing with visual results from $Fit_{t=3}$, the responses on these non-edge points are further decreased in the two images detected by the evolved program from $Fit_{t=30}$. However, the weak responses on non-edge points from the background in the two detected images from $Fit_{t=30}$ are increased a bit. Since these increased responses are still obviously weaker than the responses on edge points, the influence can be neglected. Also, from the subjective view, the detected images from $Fit_{t=30}$ is thinner than the detected images from $Fit_{t=3}$. These observations suggest that the multiple thresholds in Fit can improve the ability of finding better programs to construct features.

Figure 8.5 shows the image 69020 detected by the best evolved program from $Set_{bf,rand}$ with $Fit_{t=3}$ and the combination x_{bayes} . As can be seen, the detected image by the best evolved program from GP is improved in areas 1 and 2. Also, the responses on edge points are obviously higher than the non-edge points. Most of the true non-edge points have very weak responses (very dark) for this image, which is the same as the detection result from x_{bayes} . Although x_{bayes} decreases responses on most of the non-edge points, it wrongly strengthens responses on a few non-edge points.

8.4.5 Example Evolved Bayesian-based Program

The best evolved Bayesian-based GP program from $Set_{bf,rand}$ with $Fit_{t=3}$ is described by Equation (8.7), where BG_{bayes_1} indicates an evolved Bayesian-

Figure 8.5: Image 69020 detected by an evolved program from $Set_{bf,rand}$ with $Fit_{t=3}$ and the combination x_{bayes} .

based program, and subscript indices 0, 1 and 2 represent the basic features T_{gg} , T_{sd} and T_{hg} (in b_x , abbreviated from $b_{1|x}$), respectively. Function $b_{0,1}(T_{gg})$ is used to construct a model with the multivariate including T_{gg} and T_{sd} . Note that the proposed function accepts redundant variables, such as $x = \{T_{gg}, T_{gg}\}$. From the first part $b_{0,1,2}(b_{0,1}(b_{0,1}(T_{gg})))$, the formula aims at finding true edge points detected by T_{gg} and then constructing a feature with three basic variables. Since T_{gg} has high recall, this first part should be good at finding true edge points. The second part $b_2(b_2(b_2(T_{hg})))$ only includes the basic feature T_{hg} . After repeating estimation on true edge points (using the second part), the responses for those non-edge points with high responses in T_{hg} should be decreased. Since the output of the Bayesian-based function is based on the class edge point, the accuracy (precision) for the constructed feature should not be too low. Generally, the responses on clear boundaries (easily detected) are accurately given. Therefore, the second part possibly focuses on the boundary detection.

$$BG_{bayes_1} = b_{0,1,2}(b_{0,1}(b_{0,1}(T_{gg}))) \oplus b_2(b_2(b_2(T_{hg}))) \quad (8.7)$$

Figure 8.6: Two example images detected by two parts of the best evolved method from GP.

In order to visually present the characteristics from the two parts, Figure 8.6 shows two example images detected by the two parts of the best evolved method. “First” means the images detected by the first part $b_{0,1,2}(b_{0,1}(b_{0,1}(T_{gg})))$, and “Second” for the second part $b_2(b_2(b_2(T_{hg})))$. From the two example detected images, the second part appears to be better than the first part to detect true edge points. Since the first part focuses on finding edge points (possibly with low precision), the combination of both parts gives high responses on boundaries and low responses on non-edge points. The test performances (F_{max}) on the first part and second part for the 100 BSD test images are 0.5231 and 0.5436, and the complete BG_{bayes_1} has $F_{max} = 0.5847$. Compared with T_{hg} , the second part does not improve the F_{max} value, but it decreases response magnitudes for non-edge points. Iteratively estimating a single feature might be helpful to improve detection performance, which is a future work.

In order to further check the performance of both parts, Figure 8.7 reveals the details of recall and precision for the two parts of the evolved programs. From the curves, it is found that the recall and precision values of the second part are almost located on the curve of T_{hg} . Different from

Figure 8.7: Recall and precision for an evolved program, its two parts and T_{hg} .

T_{hg} , the recall and precision values of the second part of the evolved program crowd in an area (with not too low recall and not too low precision). From the view of the performance based on different thresholds, repeating estimation of a single variable with its estimated outputs transforms the variable values into a suitable range which is not strongly affected by the change of a threshold. Although neither the first part nor the second part is better than T_{hg} , the program combining the two parts is clearly significantly better.

8.5 Further Discussions

This section discusses the influence of the general algebraic operators, computational cost for the settings, and convergence of the GP system.

8.5.1 Influence of General Algebraic Operators

The results from $Fit_{t=3}$ and $Fit_{t=30}$ (in Tables 8.2 and 8.4) show that multiple thresholds obviously affect the test performance of the evolved programs. When Set_{bt} and $Fit_{t=3}$ are used, the GP system only finds x_{bayes} as the final solution. However, when Set_{bt} and $Fit_{t=30}$ are used, the evolved programs are significantly better the results from Set_{bt} with $Fit_{t=3}$.

Since $Fit_{t=3}$ mainly focuses on selecting programs with high contrast edge responses, the two general algebraic operators do not work well on combinations of basic features. If the terminal set only includes x and the function set only includes the two general algebraic operators, the test performance (F_{max}) for using $Fit_{t=3}$ is 0.5459 ± 0.0087 . From the evolved programs, it is found that 25 of the 30 evolved programs are equal to T_{hg} . Therefore, it is hard to find a good combination of basic features when $Fit_{t=3}$ is used. This suggests that combining basic features with the two general algebraic operators might not be good to construct composite features with high contrast edge responses.

When $Fit_{t=30}$ is used to evolve programs, the terminal set only includes x , and the function set only includes the two general algebraic operators, the test performance (F_{max}) is 0.5714 ± 0.0125 . As can be seen, only using the two operators can also obtain good composite features. The test performance on Set_{bt} with $Fit_{t=30}$ is very close to the test performance without using x_{bayes} . In order to check whether the evolved programs from Set_{bt} with $Fit_{t=30}$ include any x_{bayes} , the evolved programs are simplified. From the simplification, the number of the evolved programs without x_{bayes} is only nine out of 30 (less than one third). Therefore, x_{bayes} is still helpful to construct good composite features. Equation (8.8) gives an evolved program BG_{bayes_2} ($F_{max} = 0.5941$) from Set_{bt} with $Fit_{t=30}$, where $\frac{3}{8}$ and $\frac{5}{8}$ are scale parameters after simplification. The evolved program BG_{bayes_2} includes x_{bayes} , T_{hg} and T_{gg} .

$$BG_{bayes_2} = \frac{3T_{gg}}{8} \oplus \frac{5T_{hg}}{8} \oplus [(T_{hg} \ominus T_{gg}) \ominus (x_{bayes} \oplus T_{hg})] \quad (8.8)$$

Figure 8.8: Two example images detected by two evolved programs from Set_{bt} using $Fit_{t=30}$.

Figure 8.8 shows two example images detected by BG_{bayes_2} , and an evolved program without x_{bayes} (from Set_{bt} using $Fit_{t=30}$, and $F_{max} = 0.5940$). Comparing the visual results, the red circled areas (marked as number 1) in the two images detected by the evolved program without x_{bayes} have lower edge responses than the relevant results from BG_{bayes_2} . A reason is that the responses on composite features combined by the general algebraic operators are dependent on the responses in the basic features, but Bayesian-based programs increase the edge response contrast. Since the technique only using general algebraic operators is different from the Bayesian-based technique, further investigation on both techniques in the GP system will be conducted in the future.

8.5.2 Computational Cost

Table 8.5 gives the training time (mean $\pm$ standard deviations in seconds) for evolving programs by using the six settings respectively. Here, the p -value is obtained from the comparison between the results from $Fit_{t=3}$ and $Fit_{t=30}$ in each setting by using a paired-sample t -test because of the

Table 8.5: Training time (means $\pm$ standard deviations in seconds) from $Fit_{t=3}$ and $Fit_{t=30}$ using the six settings.

Setting	$Fit_{t=3}$	$Fit_{t=30}$	p -value
$Set_{bf,rand}$	190.9207 ± 61.6907	19.8340 ± 13.6920	0.0000 $\downarrow$
$Set_{bf,all}$	73.6683 ± 10.6519	14.9423 ± 12.3262	0.0000 $\downarrow$
Set_{bt}	47.2473 ± 11.3341	73.2913 ± 47.6183	0.0131 $\uparrow$
$Set_{bt,all}$	314.3730 ± 74.1631	53.6197 ± 46.0516	0.0000 $\downarrow$
$Set_{bt,rand}$	111.5797 ± 16.8419	63.4163 ± 41.6157	0.0000 $\downarrow$
Set_{full}	167.5110 ± 38.1950	43.1970 ± 38.7859	0.0000 $\downarrow$

same initial population. Note that $\downarrow$ means that the training time of using $Fit_{t=3}$ is significantly longer than the training time of using $Fit_{t=30}$, and $\uparrow$ means that the training from the former is significantly shorter than the latter. All experiments are run on single machines with CPU 3.1 GHz at the full speed state (the initial CPU speed is 1.6 GHz for the power saving feature). Note that the value for $Fit_{t=3}$ or $Fit_{t=30}$ is calculated after only visiting the training data once, so the calculation cost on $Fit_{t=3}$ and $Fit_{t=30}$ is very close when evaluating the same program.

From an overall view, the training time for each setting is quite short. The test time for detecting an image can be ignored (several milliseconds) because all basic features are loaded into the memory before executing a program.

From the table, the training time of using $Fit_{t=3}$ is significantly longer than the training time of using $Fit_{t=30}$ in each setting, except for Set_{bt} . Since subtrees including the two general algebraic operators might be important to construct a good program, the number of Bayesian-based functions might be reduced in the evolved good programs from $Fit_{t=30}$. Note that an algebraic operator has less computational cost than a Bayesian-based function. Since a Bayesian-based function needs to calculate the

Figure 8.9: Means of the best fitness at each generation in each setting.

sample means and standard deviations of its input, the training cost from $Fit_{t=3}$ is higher than the training cost from $Fit_{t=30}$, except for Set_{bt} . Since the evolved programs from Set_{bt} only do not include Bayesian-based functions, the training time of using $Fit_{t=30}$ in Set_{bt} is longer than the training time of using $Fit_{t=3}$ in Set_{bt} . Also there is another reason for using $Fit_{t=30}$ being longer than using $Fit_{t=3}$ for Set_{bt} , that is, $Fit_{t=3}$ in Set_{bt} only finds x_{bayes} as final solutions; but $Fit_{t=30}$ in Set_{bt} needs to find good subtrees constructed by the two general algebraic operators, which takes some computational cost.

8.5.3 Convergence

Since the population size and maximum generation are very small in the settings, it is worth investigating the convergence of this Bayesian-based GP system. Figure 8.9 reveals that the average best fitness at each generation in the six settings when $Fit_{t=3}$ or $Fit_{t=30}$ is used. From Figure 8.9 (a), since Set_{bt} only finds x_{bayes} as the final solutions, the best fitness value stays horizontal. The other settings almost reach constant values after generation 40 when $Fit_{t=3}$ is used. For $Fit_{t=30}$ (see Figure 8.9 (b)), all settings are approximately convergent to constant values after generation 30.

$Set_{bf,rand}$ and $Set_{bf,all}$ increase a bit in fitness after generation 43. Therefore, Figures 8.9 (a) and (b) indicate that the GP system is convergent or very closely convergent at generation 50.

8.6 Chapter Summary

The goal of this chapter was to investigate automatic high-level feature construction for edge detection using GP and Bayes' theorem. The Bayesian technique was employed to combine basic features via using a general multivariate normal density. The goal was successfully achieved by evolving Bayesian-based programs with Bayesian-based functions and terminals, and two general algebraic operators.

From the results, firstly, the evolved Bayesian-based programs are better than the simple Bayesian model directly using the multivariate normal density, and also have high contrast edge responses. Secondly, when the fitness function only uses three thresholds to search programs with high contrast edge responses, the Bayesian-based function with randomly selected basic features is better than the function using the full set of basic features. However, when the fitness function uses 30 thresholds, there are no significant differences in terms of the test performance F_{max} . Thirdly, in order to obtain better high contrast edge responses on a composite features, using the general multivariate normal density as a function is better than using it as a terminal to evolve programs. Lastly, further analysis of the best evolved program reveals that iteratively estimating a single feature may help to improve detection performance.

Chapter 9

Conclusions and Future Work

The overall goal of this thesis was to investigate GP for automatic edge feature extraction using different amounts of existing knowledge from very little (only using raw pixel intensities and ground truth) to more advanced domain knowledge such as Gaussian filters. This goal was achieved by developing a number of new methods in GP to evolve edge detectors to extract edge features. These new methods were evaluated by applying GP to an image benchmark dataset with a large numbers of images (200 training images and 100 test images) for edge detection. The test performance of the evolved programs were improved from low-level feature construction to high-level feature construction.

The rest of this chapter summarises the research objectives achieved by this thesis and the main conclusions from Chapters 3–8, and provides discussions for future research.

9.1 Achieved Objectives

This thesis has achieved the six research objectives in Section 1.2 (on page 6).

- This thesis developed a GP method to automatically select neighbours to construct low-level edge features in Chapter 3. A search

operator with single pixels and two search operators with blocks of pixels were designed to automatically extract effective edge features. When only full images and their ground truth are given (without any other prior domain knowledge), the GP system evolved edge detectors that are better than the Sobel and Laplacian edge detectors.

- This thesis developed several new fitness functions in GP to construct low-level edge features in Chapter 4. The investigation on new fitness functions has been conducted by employing F -measure and FOM from both the pixel view and the image view. Also, a new fitness function combined both matching directions (predictions and ground truth) in FOM to effectively improve the performance of the evolved edge detectors. The two new fitness functions, restricting the distance from the used fixed threshold to observations, effectively obtained good soft edge maps of evolved edge detectors.
- This thesis analysed pixels implicitly selected by GP and validated that actual pixels selected by GP are rich to construct edge filters in Chapter 5. A merge operation was proposed to extract a set of rich neighbours from GP edge detectors. These extracted neighbours are better than all neighbours in 3×3 and 5×5 windows to construct second-order edge filters.
- This thesis developed a GP system to evolve effective Gaussian-based edge detectors in Chapter 6. Manually combining Gaussian filters and tuning the Gaussian filter parameters were avoided in the GP system. A sampling technique (adaptively selecting a small number of training images in each generation) was proposed to evolve Gaussian-based edge detectors. Experiments show that GP evolved Gaussian-based edge detectors are better than the Gaussian gradient and the surround suppression techniques.
- This thesis proposed a new GP system for composite feature con-

struction with estimated distribution to represent different responses on edges in Chapter 7. The GP system using cumulative distribution functions (CDFs) to construct edge features performed better than the GP system using probability density functions (PDFs). Experiments show that the evolved features can reasonably indicate correct responses on different edges.

- This thesis proposed a new GP system to evolve high-level edge features in Chapter 8. A simple Bayesian model was employed as a function (in the function set) to evolve programs to construct composite features. Experiments show that the evolved programs performed better than the simple Bayesian model to obtain composite features.

9.2 Main Conclusions

The main conclusions for the six research objectives in Section 1.2 are drawn in this section.

9.2.1 Low-level Edge Feature Construction

Search Operators for Feature Construction

Chapter 3 investigated three search operators (finding a single pixel, a single block of pixels and two blocks of pixels) to construct low-level edge detectors. When only full training images and their ground truth are given, GP can use the three search operators to evolve good low-level edge detectors for detecting edges in natural images. These three search operators based on full images avoided manually setting the window size. The goal of automatically searching pixels to construct low-level edge detectors was successfully achieved via using the three search operators in GP.

When the search operator with a single block of pixels was used to evolve edge detectors, the detection accuracy was higher than using the search operator with single pixels only. The detected images showed that the evolved edge detectors from using the search operator with a single block of pixels have some ability to suppress textures. However, the computational cost of the evolved edge detectors is increased.

When the search operator with two blocks of pixels was added into the function set, the detection accuracy of the evolved edge detectors did not have significant change. However, the evolved edge detectors (after adding the search operator with two blocks of pixels) were much faster than the edge detectors evolved by GP using the search operator with a single block of pixels to detect edges.

The edge detectors evolved by GP using the search operator with single pixels outperformed the Laplacian and Sobel edge detectors. Compared with the edge detectors evolved by GP using a moving 5×5 window, the edge detectors evolved by GP using the search operator with single pixels had an obviously faster speed to detect edges (only half of the test time of the edge detectors evolved by GP using the 5×5 window). The evolved edge detectors using blocks of pixels had significantly better detection accuracy than the Gaussian gradient.

The search operator with single pixels has low computational cost for evolved edge detectors, but does not suppress textures. The search operator with a single block of pixels is helpful to suppress textures, but brings heavy computational cost for evolved edge detectors. The search operator with two blocks of pixels is helpful to suppress textures, and does not bring heavy computational cost for evolved edge detectors. Using the search operator with single pixels in GP is suggested for evolving effective low-level edge detectors with fast detecting speed when textures in images do not strongly affect extracted edges. If texture suppression is needed, the search operator with two blocks of pixels is suggested.

Fitness Functions for Feature Construction

Chapter 4 investigated fitness functions in GP for evolving low-level edge detectors. The goal of investigating fitness functions in GP for constructing low-level edge detectors was successfully achieved by proposing a number of new fitness functions based on different evaluation criteria.

To investigate the influence of the evaluation from the pixel view and the image view, *F*-measure and FOM as fitness functions from the pixel and the image views have been investigated. A fitness function from the pixel view only contains one step, evaluating all pixels across all different images. A fitness function from the image view contains two steps, getting a measurement for each image and summarising the measurements of all images. The experiment results indicate that *F*-measure is better to train programs from the pixel view than from the image view. On the other hand, FOM is suggested for the evaluation from the image view using the arithmetic mean.

To investigate the difference between *F*-measure and FOM as fitness functions to train edge detectors, the results from *F*-measure and FOM were compared in terms of the detection accuracy after using one-to-one assignment (only allowing one predicted edge point to match one true edge point if possible, and vice versa). The experiment results show that *F*-measure is better to train edge detectors than FOM.

To investigate the matching directions (between predictions and ground truth) in FOM, a proposed FOM variant combined both matching directions and *F*-measure. The experiment results show that the combination of both matching directions in FOM is better to train edge detectors than the standard FOM and the standard *F*-measure.

To investigate soft edge maps transformed from observations of the evolved edge detectors, fitness functions based on spread of observations of evolved edge detectors were proposed. The results show that a proposed fitness function using restriction of a short distance from observations to threshold 0 (based on *F*-measure) is better than *F*-measure to ob-

tain soft edge maps. Another proposed fitness function with restriction on observation spread after using the non-linear transformation on observations is better to obtain soft edge maps of the evolved edge detectors than the fitness function using F -measure or the correlation ratio of observations.

In addition, the evolved edge detectors from the BSD image dataset were applied to the ETHZ image dataset. For the ETHZ image dataset, the results from the fitness functions with F -measure and FOM are better than the Canny edge detector, in terms of detection accuracy.

Analysis of Pixels Selected by GP

The goal of analysing low-level edge detectors evolved by GP was successfully achieved in Chapter 5 via investigating constructing edge filters using only the pixels extracted from GP evolved edge detectors.

Two sets of pixels selected from GP edge detectors have been used to construct good edge filters. Via applying the proposed merge operation, a rich but compact set of neighbours was efficient and effective for constructing second-order filters. These were not significantly different from the GP edge detectors but were significantly better than the Sobel edge detector. This indicates that valuable information can be extracted from the evolved programs.

9.2.2 Gaussian-based Edge Feature Construction

The goal of investigating ways of evolving effective Gaussian-based edge detectors was achieved in Chapter 6 via employing three different types of Gaussian filters as terminals and a proposed function $\mathbb{C}$ to combine Gaussian filters in GP. The three types of Gaussian filters are Difference of Gaussians, the Gaussian gradient, and the Gaussian second-order derivative.

The experiment results show that the effective Gaussian-based edge detectors evolved by GP are significantly better than the Gaussian gra-

dient and the surround suppression techniques. Automatically evolving Gaussian-based programs by GP avoids manually combining Gaussian filters and tuning parameters of Gaussian filters.

The experiment results also show that a small number of full images *can* be used to train effective Gaussian-based edge detectors. An adaptive sampling technique (randomly selecting a small set of images in 20 images) was developed to train Gaussian-based edge detectors. These edge detectors are not significantly different from the evolved results from using the whole set of 20 images. A suitable space for Gaussian-based candidate edge detectors was introduced for explaining why the random and adaptive sampling technique worked well on evolving Gaussian-based edge detectors.

The three types of Gaussian filters have been analysed. The experiment results show that a good combination should choose different types of Gaussian filters rather than choosing a single type of Gaussian filter only.

In addition, an evolved example edge detector was interpreted. We find that a subtree, similar to the surround suppression technique, is included in the evolved program. It seems that GP can further develop Gaussian-based techniques.

9.2.3 Composite Edge Feature Construction

Distribution-based Feature Construction

When a set of basic features from different feature extraction methods was given, GP was used to construct composite features for performance improvement in Chapter 7. The goal of investigating automatic construction of composite edge features was achieved by using an estimated *distribution* to construct composite features. Basic invariant features (the image gradient, the normalised standard deviation and the histogram gradient) have been employed to construct composite invariant features.

The composite features constructed by GP using the Gaussian distribution are significantly better than the three basic features. From visually detected images, the features constructed by GP with the Gaussian distribution increase response contrast on edge points and suppress textures on non-edge points. The constructed features from GP combine some advantages from the three basic features, and appear to reduce their disadvantages.

From the results, the estimation using the PDF with the linear relationship (triangular distribution) for feature construction is worse than the estimation on the PDF with the non-linear relationship (Gaussian distribution). There are no significant differences between the estimation on the CDFs with the linear relationship (uniform distribution) and with the non-linear relationship (Gaussian distribution).

The experiment results show that the composite features constructed by GP with CDFs (linear and non-linear distribution) are significantly better than the composite features from GP using PDFs with the linear distribution (triangular distribution), but are not significantly different from the features constructed by GP using the non-linear PDFs (Gaussian distribution), in terms of detection accuracy. From the visually detected images, the composite features constructed by GP using CDFs are rich to indicate responses on subjective edges (marked by only a few people). From a quantitative view, CDF is better than PDF to be used for composite feature construction in GP.

The features constructed by GP using CDFs (of Gaussian distribution and uniform distribution) are better than the combinations from the Bayesian model using the general multivariate normal density and the linear SVM, in terms of detection accuracy (quantitative view) and rich responses on subjective edge points (qualitative view). The results show that GP has some ability to find a way of efficiently combining different features together and describe rich responses on different edges.

Also, an evolved program was reasonably explained. From the analy-

sis of the evolved program, the approach found from the evolved feature appears to suppress wrong responses on non-edge points. It suggests that GP has potential to find new technique algorithms to further develop edge detection methods.

The GP system has been successfully applied to construct variant features using the horizontal edge information only as the training data. The constructed horizontal edge features have stable performance on detecting edges.

High-level Feature Construction

A simple Bayesian technique using the general multivariate normal density has been developed for the proposed GP system in Chapter 8. The goal of investigating automatic high-level feature construction was achieved by applying this GP system to construct high-level features. Bayesian-based programs are constructed by Bayesian-based functions, Bayesian-based terminals, basic features (as terminals), and two general algebraic operators.

The experiment results show that the Bayesian-based programs evolved by GP using the Bayesian-based function (randomly selecting basic features) are significantly better than the combination of the simple Bayesian model directly using the multivariate normal density (on the basic features), and the three basic features. Similar to the simple Bayesian model, the features constructed by the Bayesian-based programs had high contrast edge responses.

When the fitness function used in GP was based on three thresholds, the Bayesian-based function including all basic features was worse than the function randomly selecting basic features to construct high-level features by GP. However, when the fitness function was based on 30 thresholds, there were no significant differences between the function using all basic features and randomly selecting basic features.

When the fitness function was based on three thresholds and Bayesian-

based terminals were used, the evolved programs were significantly worse than the programs evolved by GP using the Bayesian-based function. However, when 30 thresholds were used in the fitness function, there are no differences between the evolved programs with the Bayesian-based terminal and the Bayesian-based functions, in terms of detection accuracy.

An evolved program was interpreted. Further analysis of the evolved program reveals that iteratively estimating a single feature may help improve detection performance.

9.3 Limitations

When the training data is based on full images, the Gaussian-based GP system will take a long time to evolve edge detectors. Each run for using the 20 training images in Section 6.5.1 took around 4.5 days. When an evolved program is used to extract edge features, the evaluation is based on features, not the final binary thinned edge maps. If a thinning operation is included in the training stage, the computational cost will be increased. Also, if one-to-one assignment is considered, the computational cost will increase hugely. Therefore, the heavy computational cost (in training) is a problem for the Gaussian-based GP system. Even for low-level feature construction, the GP system will have very heavy computational cost if including post-processing techniques and one-to-one assignment for the evaluation in training. Note that the test time (applying the evolved GP detectors to unseen images for edge detection) is quite short, and not significantly different from the existing techniques. These proposed techniques are suitable for problems/tasks that do not mind the long offline training but have a high demand of test accuracy (and a short test time). They are not suitable for real-time problems that requires fast online training.

In order to address the heavy computational cost problem, one easy solution is to use parallel computing to decrease the computational cost

on a single machine. Also, in the Gaussian-based GP system, the parameters of each Gaussian filter in a program need to be calculated when the program detects images. A solution is to use approximate parameters for these filters. Therefore, before evolving programs, all results from a set of fixed filters are calculated, and they are kept in the memory during the evolving stage. However, a large enough memory is required.

Since the ground truth used in this thesis are not final thinned binary edge maps, the thickness problem exists in the evolved program. However, the thick edges from GP edge detectors can be easily thinned by a simple thinning operation [106]. Section 6.5.3 (on page 225) showed example detected images thinned by the thinning operation. Also, non-maximum suppression will be helpful to reduce the thickness of edge responses [28, 140].

9.4 Future Work

This section highlights some recommended areas of future work.

1. Combining Evolved Low-level Features

This thesis only focuses on edge feature extraction. When the evolved edge detectors are used to extract edge features, how to efficiently and effectively combine these evolved edge detectors to further improve detection performance is a challenge.

2. Multi-objective Optimisation

In Chapter 3, the size of the best program is increased after some generations, but the fitness value is only slightly improved. When evolving edge detectors, the detection accuracy and detection time are important for real time applications. Therefore, both the size of an evolved program and its detection accuracy should be addressed in the future.

3. Analysis of Evolved Programs

From the analysis of low-level edge detectors, a rich but compact set of pixels is good to construct second-order edge filters. The evolved programs possibly contain implicit new knowledge (such as new structures after simplifying the programs). Extracting new knowledge is a challenge. It is worth investigating building blocks in evolved programs to further understand the edge detection problem.

4. Gaussian-based GP system

The GP system only employs three types of Gaussian filters to construct Gaussian-based edge detectors. In order to improve detection performance, it is worth adding more Gaussian-based filters into the terminal set. The existing transformation-based techniques, such as Gabor filters [85], are expected to be combined for detection performance improvement.

5. Estimated Distribution

When an estimated distribution is applied to construct composite edge features, features with high contrast edge responses can be obtained from GP using PDFs of the Gaussian distribution, and rich edge features can be obtained from GP using CDFs or PDFs of the triangular distribution. However, the training information on true edge points are not fully used. A sampled pixel is simply considered as an edge point or a non-edge point. From the ground truth, the number of people marking a pixel as an edge point is different. When a weight is used to describe how possible it is to consider a pixel as a true edge point, whether the weight can be added into estimated distribution for performance improvement (in terms of the quantitative and qualitative views) needs to be investigated.

6. High-level Feature Construction

The simple Bayesian model and the two general algebraic operators are used in the Bayesian-based GP system for constructing edge features. The fitness function using three thresholds was used to construct edge features with high contrast edge responses. However, using three thresholds for the fitness evaluation is not as good as using 30 thresholds to evolve composite features, in terms of detection accuracy. Since the Bayesian model is simple, more functions based on Bayesian inference techniques need to be developed for performance improvement.

7. Generalisation

It is possible that GP has the potential work on edge detection with colour images. This thesis only focuses on the grayscale images, the colour information are not considered. However, colours contain more edge information than grayscale levels [123]. If colour information is applied to the GP system, the test performance would be expected to be improved. In the GP system, colour information will increase the number of terminals, or evolved edge detectors from different colour information will be combined together. For example, in the RGB system, the terminal individual image will be divided into three terminals: red channel, green channel and blue channel. It is easy to extend the GP system for evolving programs to detect colour images. Also, it is suggested that GP with distribution could work on feature construction. This thesis utilises GP with estimated distribution to construct composite features for edge detection. Since the estimated distribution is based on observations of programs, the GP system can be easily extended to construct features for other classification problems. Finally, the GP system might have the potential to work on different tasks (such as face detection and objective detection) when ground truth for edges is changed to desired outputs (faces or objects).

Bibliography

- [1] AGAIAN, S. A., PANETTA, K. A., NERCESSIAN, S. C., AND DANAHY, E. E. Boolean derivatives with application to edge detection for imaging systems. *IEEE Transactions on Systems, Man, and Cybernetics. Part B* 40, 2 (2010), 371–382.
- [2] ALIPOOR, M., IMANDOOST, S., AND HADDADNIA, J. Designing edge detection filters using particle swarm optimization. In *Proceedings of the 18th Iranian Conference on Electrical Engineering (ICEE)* (2010), pp. 548–552.
- [3] ANDO, S. Image field categorization and edge/corner detection from gradient covariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), 179–190.
- [4] ARBELÁEZ, P., MAIRE, M., FOWLKES, C., AND MALIK, J. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 5 (2011), 898–916.
- [5] ARGYLE, E., AND ROSENFELD, A. Techniques for edge detection. *Proceedings of the IEEE* 59, 2 (1971), 285–287.
- [6] ATKINS, D., NESHATIAN, K., AND ZHANG, M. A domain independent genetic programming approach to automatic feature extraction for image classification. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2011), pp. 238–245.

- [7] AWCOCK, G. J., AND THOMAS, R. *Applied Image Processing*. Macmillan, 1995.
- [8] AZAD, P., GOCKEL, T., AND DILLMANN, R. *Computer Vision: Principles and Practice*. Elektor, 2008.
- [9] BADDELEY, J. A. An error metric for binary images. In *Proceedings of the IEEE Workshop on Robust Computer Vision* (1992), pp. 59–78.
- [10] BALLARD, D. H., AND BROWN, C. M. *Computer Vision*. Prentice Hall, 1982.
- [11] BANZHAF, W., NORDIN, P., KELLER, R., AND FRANCONI, F. *Genetic Programming: An Introduction*. Morgan Kaufmann, 1997.
- [12] BAO, P., ZHANG, L., AND WU, X. Canny edge detection enhancement by scale multiplication. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 9 (2005), 1485–1490.
- [13] BASTÜRK, A., AND GÜNAY, E. Efficient edge detection in digital images using a cellular neural network optimized by differential evolution algorithm. *Expert Systems with Applications* 36, 2 (2009), 2645–2650.
- [14] BASU, M. Gaussian-based edge-detection methods: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 32, 3 (2002), 252–260.
- [15] BENNAMOUN, M., BOASHASH, B., AND KOO, J. Optimal parameters for edge detection. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* (1995), vol. 2, pp. 1482–1488.
- [16] BERGHOLM, F. Edge focusing. *IEEE Transactions on Image Processing* 9 (1987), 726–741.

- [17] BERTERO, M., POGGIO, T., AND TORRE, V. Ill-posed problems in early vision. *Proceedings of the IEEE* 76, 8 (1988), 869–889.
- [18] BHOWAN, U., JOHNSTON, M., AND ZHANG, M. Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 42, 2 (2011), 406–421.
- [19] BOLIS, E., ZERBI, C., COLLET, P., LOUCHET, J., AND LUTTON, E. A GP artificial ant for image processing: preliminary experiments with EASEA. In *Proceedings of the 4th European Conference on Genetic Programming* (2001), pp. 246–255.
- [20] BOUAYNAYA, N., CHARIF-CHEFCHAOUNI, M., AND SCHONFELD, D. Theoretical foundations of spatially-variant mathematical morphology part I: binary images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 5 (2008), 823–836.
- [21] BOUAYNAYA, N., AND SCHONFELD, D. Theoretical foundations of spatially-variant mathematical morphology part II: gray-level images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 5 (2008), 837–850.
- [22] BOUKERROUI, D., NOBLE, J. A., AND BRADY, M. On the choice of band-pass quadrature filters. *Journal of Mathematical Imaging and Vision* 21 (2004), 53–80.
- [23] BOVIK, A., HUANG, T. S., AND MUNSON, D.C., J. The effect of median filtering on edge estimation and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9*, 2 (1987), 181–194.
- [24] BOVIK, A. C., HUANG, T. S., AND MUNSON, D. C. J. Nonparametric tests for edge detection in noise. *Pattern Recognition* 19 (1986), 209–219.

- [25] BOWYER, K., KRANENBURG, C., AND DOUGHERTY, S. Edge detector evaluation using empirical ROC curves. *Computer Vision and Image Understanding* 84, 1 (2001), 77–103.
- [26] BRAMEIER, M. F., AND BANZHAF, W. *Linear Genetic Programming*. Springer, 2007.
- [27] BURGESS, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining Knowledge Discovery* 2, 2 (1998), 121–167.
- [28] CANNY, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8, 6 (1986), 679–698.
- [29] CHAIRA, T., AND RAY, A. K. A new measure using intuitionistic fuzzy set theory and its application to edge detection. *Applied Soft Computing* 8 (2008), 919–927.
- [30] CHAN, T., AND VESE, L. Active contours without edges. *IEEE Transactions on Image Processing* 10, 2 (2001), 266–277.
- [31] CHEN, T., WU, Q. H., RAHMANI-TORKAMAN, R., AND HUGHES, J. A pseudo top-hat mathematical morphological approach to edge detection in dark regions. *Pattern Recognition* 35, 1 (2002), 199–210.
- [32] CHERKASSKY, B. V., AND GOLDBERG, A. V. On implementing push-relabel method for the maximum flow problem. *Algorithmica* 19 (1994), 390–410.
- [33] CHUANG, E., AND SHER, D. χ^2 test for feature detection. *Pattern Recognition* 26, 11 (1993), 1671–1681.
- [34] CROUSE, M., NOWAK, R., AND BARANIUK, R. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on Signal Processing* 46, 4 (1998), 886–902.

- [35] DAUBECHIES, I. The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory* 36, 5 (1990), 961–1005.
- [36] DEBAYLE, J., AND PINOLI, J.-C. General adaptive neighborhood image processing: part I: introduction and theoretical aspects. *Journal of Mathematical Imaging and Vision* 25 (2006), 245–266.
- [37] DEBAYLE, J., AND PINOLI, J.-C. General adaptive neighborhood image processing: part II: practical application examples. *Journal of Mathematical Imaging and Vision* 25 (2006), 267–284.
- [38] DOLLAR, P., TU, Z., AND BELONGIE, S. Supervised learning of edges and object boundaries. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2006), vol. 2, pp. 1964–1971.
- [39] DONOSER, M., RIEMENSCHNEIDER, H., AND BISCHOF, H. Linked edges as stable region boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2010), pp. 1665–1672.
- [40] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.
- [41] EBNER, M. On the edge detectors for robot vision using genetic programming. In *Proceedings of Horst-Michael Groß, Workshop SOAVE 97 - Selbstorganisation von Adaptivem Verhalten* (1997), pp. 127–134.
- [42] EIBEN, A. E., AND SMITH, J. *Introduction to Evolutionary Computing*. Springer, 2003.
- [43] ESPARCIA-ALCÁZAR, A. I., Ed. *Applications of Evolutionary Computing*, vol. 7835. Springer, Lecture Notes in Computer Science, 2013.
- [44] ESPEJO, P. G., VENTURA, S., AND HERRERA, F. A survey on the application of genetic programming to classification. *IEEE Transactions*

- on Systems Man Cybernetics Part C: Applications and Reviews* 40 (2010), 121–144.
- [45] FEI-FEI, L., FERGUS, R., AND PERONA, P. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 4 (2006), 594–611.
- [46] FERNÁNDEZ-GARCÍA, N., CARMONA-POYATO, A., MEDINA-CARNICER, R., AND MADRID-CUEVAS, F. Automatic generation of consensus ground truth for the comparison of edge detection techniques. *Image and Vision Computing* 26, 4 (2008), 496–511.
- [47] FERRARI, V., TUYTELAARS, T., AND GOOL, L. V. Object detection by contour segment networks. In *Proceedings of the European Conference on Computer Vision* (2006), vol. 3953, pp. 14–28.
- [48] FISHER, R. A. *Statistical Methods for Research Workers* (14th Edition). Edinburgh, U.K.: Oliver & Boyd, 1970.
- [49] FREI, W., AND CHEN, C.-C. Fast boundary detection: A generalization and a new algorithm. *IEEE Transactions on Computers* C-26, 10 (1977), 988–998.
- [50] FU, K.-S., AND ROSENFELD, A. Pattern recognition and image processing. *IEEE Transactions on Computers* C-25, 12 (1976), 1336–1346.
- [51] FU, W., JOHNSTON, M., AND ZHANG, M. Hybrid particle swarm optimisation algorithms based on differential evolution and local search. In *Proceedings of the 23rd Australasian Joint Conference on Artificial Intelligence* (2010), pp. 313–322.
- [52] FU, W., JOHNSTON, M., AND ZHANG, M. Analysis of diagonal derivatives in edge detectors evolved by genetic programming. In *Proceedings of the Twenty-sixth International Conference on Image and Vision Computing New Zealand* (2011), pp. 345–350.

- [53] FU, W., JOHNSTON, M., AND ZHANG, M. Genetic programming for edge detection: a global approach. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2011), pp. 254–261.
- [54] FU, W., JOHNSTON, M., AND ZHANG, M. A hybrid particle swarm optimisation with differential evolution approach to image segmentation. In *Proceedings of the International Conference on Applications of Evolutionary Computation: EvoApplications* (2011), pp. 173–182.
- [55] FU, W., JOHNSTON, M., AND ZHANG, M. Automatic construction of invariant features using genetic programming for edge detection. In *Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence* (2012), vol. 7691, pp. 144–155.
- [56] FU, W., JOHNSTON, M., AND ZHANG, M. Soft edge maps from edge detectors evolved by genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2012), pp. 24–31.
- [57] GANESAN, L., AND BHATTACHARYYA, P. Edge detection in untex-tured and textured images: a common computational framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 27, 5 (1997), 823–834.
- [58] GEUSEBROEK, J.-M., SMEULDERS, A., AND VAN DE WEIJER, J. Fast anisotropic gauss filtering. *IEEE Transactions on Image Processing* 12, 8 (2003), 938–943.
- [59] GIRÓN, E., FRERY, A. C., AND CRIBARI-NETO, F. Nonparametric edge detection in speckled imagery. *Mathematics and Computers in Simulation* 82, 11 (2012), 2182–2198.
- [60] GOLDBERG, A. V., AND KENNEDY, R. An efficient cost scaling algo-rithm for the assignment problem. *Mathematical Programming* 71, 2 (1995), 153–177.

- [61] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.
- [62] GOLONEK, T., GRZECHCA, D., AND RUTKOWSKI, J. Application of genetic programming to edge detector design. In *Proceedings of the International Symposium on Circuits and Systems* (2006), pp. 4683–4686.
- [63] GÓMEZ-MORENO, H., MALDONADO-BASCÓN, S., AND LÓPEZ-FERRERAS, F. Edge detection in noisy images using the support vector machines. In *Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks: Connectionist Models of Neurons, Learning Processes and Artificial Intelligence—Part I* (2001), pp. 685–692.
- [64] GORDON, R., AND RANGAYYAN, R. Feature enhancement of mammograms using fixed and adaptive neighborhoods. *Applied Optics* 23, 4 (1984), 560–564.
- [65] GRIGORESCU, C., PETKOV, N., AND WESTENBERG, M. Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on Image Processing* 12, 7 (2003), 729–739.
- [66] GRIGORESCU, C., PETKOV, N., AND WESTENBERG, M. A. Contour and boundary detection improved by surround suppression of texture edges. *Image and Vision Computing* 22, 8 (2004), 609–622.
- [67] HANLEY, J. A., AND MCNEIL, B. J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143, 1 (1982), 29–36.
- [68] HARALICK, R. Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6, 1 (1984), 58–68.

- [69] HARALICK, R., AND LEE, J. Context dependent edge detection and evaluation. *Pattern Recognition* 23, 1-2 (1992), 1–19.
- [70] HARDING, S., AND BANZHAF, W. Genetic programming on GPUs for image processing. *International Journal of High Performance Systems Architecture* 1, 4 (2008), 231–240.
- [71] HARRIS, C., AND BUXTON, B. Evolving edge detectors with genetic programming. In *Proceedings of the First Annual Conference on Genetic Programming* (1996), pp. 309–314.
- [72] HARVEY, N. R., BRUMBY, S. P., PERKINS, S., THEILER, J., SZYMANSKI, J. J., BLOCH, J. J., PORTER, R. B., GALASSI, M., AND YOUNG, A. C. Image feature extraction: Genie vs conventional supervised classification techniques. *IEEE Transactions on Geoscience and Remote Sensing* 40 (2001), 393–404.
- [73] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [74] HOLLINGWORTH, G., SMITH, S., AND TYRRELL, A. Design of highly parallel edge detection nodes using evolutionary techniques. In *Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing* (1999), pp. 35–42.
- [75] HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70.
- [76] HONG, G., AND ASOKE, K. N. Breast cancer diagnosis using genetic programming generated feature. *Pattern Recognition* 39, 5 (2006), 980–987.
- [77] HUANG, F., AND SU, J. Multiple face contour detection using adaptive flows. In *Proceedings of the 5th Chinese Conference on Advances in Biometric Person Authentication* (2004), pp. 137–143.

- [78] HUANG, J. S., AND TSENG, D. H. Statistical theory of edge detection. *Computer Vision, Graphics, and Image Processing* 43 (1988), 337–346.
- [79] HUTTENLOCHER, D., KLANDERMAN, G., AND RUCKLIDGE, W. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, 9 (1993), 850–863.
- [80] ILLINGWORTH, J., AND KITTLER, J. A survey of the Hough transform. *Computer Vision, Graphics and Image Processing* 44 (1988), 87–116.
- [81] JANSCHKE, M. Maximum expected F-measure training of logistic regression models. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing* (2005), pp. 692–699.
- [82] JIANG, L., WANG, D., CAI, Z., AND YAN, X. Survey of improving naive bayes for classification. In *Proceedings of the 3rd International Conference on Advanced Data Mining and Applications* (2007), pp. 134–145.
- [83] JORDAN, M. I. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* 6 (1994), 181–214.
- [84] KADAR, I., BEN-SHAHAR, O., AND SIPPER, M. Evolution of a local boundary detector for natural images via genetic programming and texture cues. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (2009), pp. 1887–1888.
- [85] KAMARAINEN, J.-K., KYRKI, V., AND KALVIAINEN, H. Invariance properties of gabor filter-based features-overview and applications. *IEEE Transactions on Image Processing* 15, 5 (2006), 1088–1099.

- [86] KASS, M., WITKIN, A., AND TERZOPOULOS, D. Snakes: Active contour models. *International Journal of Computer Vision* 1 (1988), 321–331.
- [87] KICINGER, R., ARCISZEWSKI, T., AND JONG, K. D. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers and Structures* 83, 23–24.
- [88] KIRSCH, R. Computer determination of the constituent structure of biological images. *Computers and Biomedical Research* 4 (1971), 315–328.
- [89] KITCHEN, L., AND ROSENFELD, A. Edge evaluation using local edge coherence. *IEEE Transactions on Systems, Man and Cybernetics* 11, 9 (1981), 597–605.
- [90] KOKARE, M., BISWAS, P. K., AND CHATTERJI, B. N. Edge based features for content based image retrieval. *Pattern Recognition* 36, 11 (2003), 2649–2661.
- [91] KOKKINOS, I. Boundary detection using F-measure-, filter- and feature- (F3) boost. In *Proceedings of the 11th European Conference on Computer Vision: Part II* (2010), pp. 650–663.
- [92] KONISHI, S., YUILLE, A., COUGHLAN, J., AND ZHU, S. C. Fundamental bounds on edge detection: an information theoretic evaluation of different edge cues. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1999), vol. 1, pp. 1573–1579.
- [93] KONISHI, S., YUILLE, A., COUGHLAN, J., AND ZHU, S. C. Statistical edge detection: learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 1 (2003), 57–74.
- [94] KOVESI, P. Phase congruency: A low-level image invariant. *Psychological Research* 64 (2000), 136–148.

- [95] KOZA, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, 1992.
- [96] KOZA, J. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines* 11 (2010), 251–284.
- [97] KOZA, J., BENNETT III, F. H., ANDRE, D., AND KEANE, M. A. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, 1999.
- [98] KOZA, J. R. Non-linear genetic algorithms for solving problems by finding a fit composition of functions. United States Patent 5136686, 1992.
- [99] KRAWIEC, K. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines* 3, 4 (2002), 329–343.
- [100] KRAWIEC, K., HOWARD, D., AND ZHANG, M. Overview of object detection and image analysis by means of genetic programming techniques. In *Proceedings of Frontiers in the Convergence of Bioscience and Information Technologies* (2007), pp. 779–784.
- [101] KRID, M., DAMMAK, A. M., AND MASMOUDI, D. S. Hardware implementation of a pulse mode neural network-based edge detection system. *International Journal of Electronics and Communications (AEU)* 63 (2009), 810–1820.
- [102] KRUSKAL, W. H. Historical notes on the Wilcoxon unpaired two-sample test. *Journal of the American Statistical Association* 52, 279 (1957), 356–360.
- [103] KUO, Y.-H., LEE, C.-S., AND LIU, C.-C. A new fuzzy edge detection method for image enhancement. In *Proceedings of the Sixth IEEE*

- International Conference on Fuzzy Systems* (1997), vol. 2, pp. 1069–1074.
- [104] LACROIX, V. The primary raster: a multiresolution image description. In *Proceedings of the 10th International Conference on Pattern Recognition* (1990), vol. I, pp. 903–907.
- [105] LALIGANT, O., AND TRUCHETET, F. A nonlinear derivative scheme applied to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 2 (2010), 242–257.
- [106] LAM, L., LEE, S.-W., AND SUEN, C. Thinning methodologies—a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 9 (1992), 869–885.
- [107] LANGDON, W., GUSTAFSON, S., AND KOZA, J. The genetic programming bibliography. <http://www.cs.bham.ac.uk/wbl/biblio/> (2013).
- [108] LEE, M., LEUNG, S., PUN, T., CHEUNG, H., AND LEE, A. Edge detection by genetic algorithm. In *Proceedings of the International Conference on Image Processing* (2000), vol. 1, pp. 478–480.
- [109] LIANG, L. R., AND LOONEY, C. G. Competitive fuzzy edge detection. *Applied Soft Computing* 3, 2 (2003), 123–137.
- [110] LIM, D. H. Robust edge detection in noisy images. *Computational Statistics & Data Analysis* 50, 3 (2006), 803–812.
- [111] LIM, D. H., AND JANG, S. J. Comparison of two-sample tests for edge detection in noisy images. *Journal of the Royal Statistical Society. Series D (The Statistician)* 51, 1 (2002), 21–30.
- [112] LINDBERG, T. Edge detection and ridge detection with automatic scale selection. In *Proceedings of 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1996), pp. 465–470.

- [113] LIU, S., WANG, X., QIU, X., AND HE, Y. Mathematical morphology edge detection algorithm of remote sensing image with high resolution. In *Proceedings of the 1st International Conference on Information Science and Engineering* (2009), pp. 1323–1326.
- [114] LONCARIC, S. A survey of shape analysis techniques. *Pattern Recognition* 31, 8 (1998), 983–1001.
- [115] LOPEZ-MOLINA, C., DE BAETS, B., AND BUSTINCE, H. Quantitative error measures for edge detection. *Pattern Recognition* 46, 4 (2013), 1125–1139.
- [116] LOPEZ-MOLINA, C., DE BAETS, B., BUSTINCE, H., SANZ, J., AND BARRENECHEA, E. Multiscale edge detection based on gaussian smoothing and edge tracking. *Knowledge-Based Systems*, 0 (2013).
- [117] LU, S., WANG, Z., AND SHEN, J. Neuro-fuzzy synergism to the intelligent system for edge detection and enhancement. *Pattern Recognition* 36, 10 (2003), 2395–2409.
- [118] LYVERS, E., AND MITCHELL, O. Precision edge contrast and orientation estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10, 6 (1988), 927–937.
- [119] MACLEOD, I., AND ARGYLE, E. Comments on “techniques for edge detection”. *Proceedings of the IEEE* 60, 3 (1972), 344–344.
- [120] MALLAT, S. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, 7 (1987), 674–693.
- [121] MARAGOS, P., AND VACHIER, C. Overview of adaptive morphology: Trends and perspectives. In *Proceedings of the 16th IEEE International Conference on Image Processing* (2009), pp. 2241–2244.

- [122] MARR, D., AND HILDRETH, E. Theory of edge detection. *Proceedings of the Royal Society of London, Series B, Biological Sciences* 207, 1167 (1980), 187–217.
- [123] MARTIN, D., FOWLKES, C., AND MALIK, J. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 5 (2004), 530–549.
- [124] MARTIN, D., FOWLKES, C., TAL, D., AND MALIK, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the 8th International Conference on Computer Vision* (2001), vol. 2, pp. 416–423.
- [125] MATTHEWS, B. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta* 405 (1975), 442–451.
- [126] MCKAY, R. I., HOAI, N. X., WHIGHAM, P. A., SHAN, Y., AND O'NEILL, M. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines* 11 (2010), 365–396.
- [127] MILLER, J. F., AND HARDING, S. L. Cartesian genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (Companion)* (2008), pp. 2701–2726.
- [128] MITTAL, A., SOFAT, S., HANCOCK, E., AND MOUSSET, S. A statistical operator for detecting weak edges in low contrast images. In *Proceedings of the 9th International Conference on Image Analysis and Recognition - Volume Part I* (2012), pp. 89–96.
- [129] MONTANA, D. J. Strongly typed genetic programming. *Evolutionary Computation* 3 (1994), 199–230.

- [130] MONTES, H. A., AND WYATT, J. L. Cartesian genetic programming for image processing tasks. In *Proceedings of the Neural Networks and Computational Intelligence* (2003), pp. 185–190.
- [131] MORENO, R., PUIG, D., JULIA, C., AND GARCIA, M. A new methodology for evaluation of edge detectors. In *Proceedings of the 16th IEEE International Conference on Image Processing* (2009), pp. 2157–2160.
- [132] MORRONE, M., AND OWENS, R. Feature detection from local energy. *Pattern Recognition Letters* 6, 5 (1987), 303–313.
- [133] NALWA, V. S., AND BINFORD, T. O. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 6 (1986), 699–714.
- [134] NESHATIAN, K., AND ZHANG, M. Unsupervised elimination of redundant features using genetic programming. In *Proceedings of the 22nd Australasian Joint Conference on Artificial Intelligence* (2009), pp. 432–442.
- [135] NESHATIAN, K., ZHANG, M., AND ANDREAE, P. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation* 16, 5 (2012), 645–661.
- [136] NORDIN, P. A compiling genetic programming system that directly manipulates the machine code. In *Advances in genetic programming* (1994), J. Kinnear and E. Kenneth, Eds., MIT Press, pp. 311–331.
- [137] OLSON, C., AND HUTTENLOCHER, D. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Image Processing* 6, 1 (1997), 103–113.
- [138] OPPENHEIM, A., AND LIM, J. The importance of phase in signals. *Proceedings of the IEEE* 69, 5 (1981), 529–541.

- [139] PAPARI, G., CAMPISI, P., PETKOV, N., AND NERI, A. A biologically motivated multiresolution approach to contour detection. *EURASIP Journal on Applied Signal Processing* 2007 (2007), 119–119.
- [140] PAPARI, G., AND PETKOV, N. Edge and line oriented contour detection: state of the art. *Image and Vision Computing* 29 (2011), 79–103.
- [141] PAPARI, G., AND PETKOV, N. An improved model for surround suppression by steerable filters and multilevel inhibition with application to contour detection. *Pattern Recognition* 44, 9 (2011), 1999–2007.
- [142] PARK, D. J., NAM, K. M., AND PARK, R.-H. Edge detection in noisy images based on the co-occurrence matrix. *Pattern Recognition* 27, 6 (1994), 765–775.
- [143] PASZKOWICZ, W. Genetic algorithms, a nature-inspired tool: Survey of applications in materials science and related fields. *Materials and Manufacturing Processes* 24, 2 (2009), 174–197.
- [144] PELI, T., AND MALAH, D. A study of edge detection algorithms. *Computer Graphics and Image Processing* 20, 1 (1982), 1–21.
- [145] PELLEGRINO, F., VANZELLA, W., AND TORRE, V. Edge detection revisited. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 34, 3 (2004), 1500–1518.
- [146] PERONA, P. Steerable-scalable kernels for edge detection and junction analysis. *Image Vision and Computing* 10 (1992), 663–672.
- [147] PERONA, P., AND MALIK, J. Detecting and localizing edges composed of steps, peaks and roofs. In *Proceedings of the Third International Conference on Computer Vision* (1990), pp. 52–57.

- [148] PETROVIC, N., AND CRNOJEVIĆ, V. Universal impulse noise filter based on genetic programming. *IEEE Transactions on Image Processing* 17, 7 (2008), 1109–1120.
- [149] PINHO, A. J., AND ALMEIDA, L. B. Edge detection filters based on artificial neural networks. In *Proceedings of the 8th International Conference on Image Analysis and Processing* (1995), pp. 159–164.
- [150] POLI, R. Genetic programming for image analysis. In *Proceedings of the First Annual Conference on Genetic Programming* (1996), pp. 363–368.
- [151] POLI, R., LANGDON, W. B., AND MCPHEE, N. F. *A Field Guide to Genetic Programming*. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk>, 2008. With contributions by J. R. Koza.
- [152] PRATT, W. K. *Digital Image Processing: PIKS Inside, 3rd edition*. Wiley, 2001.
- [153] PREWITT, J. M. S. Object enhancement and extraction. *Proceedings of The IEEE* (1970).
- [154] QIU, P. *Image Processing and Jump Regression Analysis*. Wiley, 2005.
- [155] REN, X. Multi-scale improves boundary detection in natural images. In *Proceedings of the 10th European Conference on Computer Vision: Part III* (2008), pp. 533–545.
- [156] REZAI-RAD, G., AND LARIJANI, H. H. A new investigation on edge detection filters operation for feature extraction under histogram equalization effect. In *Proceedings of the Geometric Modelling and Imaging* (2007), pp. 149–153.

- [157] RIVEST, J., SOILLE, P., AND BEUCHER, S. A nonlinear derivative scheme applied to edge detection. *Journal of Electronic Imaging* 2, 5 (1993), 326–336.
- [158] ROBINSON, G. Edge detection by compass gradient masks. *Computer Graphics and Image Processing* 6 (1977), 492–501.
- [159] RODRÍGUEZ-VÁZQUEZ, K., AND OLIVER-MORALES, C. Multi-branches genetic programming as a tool for function approximation. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2004), pp. 719–721.
- [160] SAPPA, A., AND DEVY, M. Fast range image segmentation by an edge detection strategy. In *Proceedings of Third International Conference on 3-D Digital Imaging and Modeling* (2001), pp. 292–299.
- [161] SCHINDLER, K., AND SUTER, D. Object detection by global contour shape. *Pattern Recognition* 41, 12 (2008), 3736–3748.
- [162] SCHUNCK, B. Edge detection with Gaussian filters at multiple scales. In *Proceedings of the IEEE Workshop on Computer Vision, Representation and Control* (1987), pp. 208–210.
- [163] SETAYESH, M., ZHANG, M., AND JOHNSTON, M. Detection of continuous smooth and thin edges in noisy images using constrained particle swarm optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2011), pp. 45–52.
- [164] SETAYESH, M., ZHANG, M., AND JOHNSTON, M. Edge detection using constrained discrete particle swarm optimisation in noisy images. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2011), pp. 246–253.
- [165] SHENOI, B. A. *Introduction to Digital Signal Processing and Filter Design*. John Wiley and Sons, 2006.

- [166] SHYI-CHYI, C., AND WEN-HSIANG, T. Image compression by moment-preserving edge detection. *Pattern Recognition* 27, 11 (1994), 1439–1449.
- [167] SMART, W., AND ZHANG, M. Using genetic programming for multiclass classification by simultaneously solving component binary classification problems. In *Proceedings of the 8th European Conference on Genetic Programming* (2005), pp. 227–239.
- [168] SOH, L.-K., AND TSATSOULIS, C. Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices. *IEEE Transactions on Geoscience and Remote Sensing* 37, 2 (1999), 780–795.
- [169] SOILLE, P. *Morphological Image Analysis: Principles and Applications*, 2nd ed. Springer-Verlag, 2003.
- [170] SONG, D. M., AND LI, B. Derivative computation by multiscale filters. *Image and Vision Computing* 16, 1 (1998), 43–53.
- [171] SONG, W., FENG, G., AND TIECHENG, L. Evaluating edge detection through boundary detection. *EURASIP Journal on Applied Signal Processing 2006* (2006), 1–15.
- [172] SONG, W.-J., AND PEARLMAN, W. Edge-preserving noise filtering based on adaptive windowing. *IEEE Transactions on Circuits and Systems* 35, 8 (1988), 1048–1055.
- [173] SONKA, M., HLAVAC, V., AND BOYLE, R. *Image Processing, Analysis, and Machine Vision (3rd edition)*. Cengage Learning, 2007.
- [174] SPURRIER, J. D. On the null distribution of the Kruskal-Wallis statistic. *Journal of Nonparametric Statistics* 15, 6 (2003), 685–691.
- [175] SULLIVAN, J., AND CARLSSON, S. Recognizing and tracking human action. In *Proceedings of the 7th European Conference on Computer Vision-Part I* (2002), pp. 629–644.

- [176] SUN, J., GU, D., CHEN, Y., AND ZHANG, S. A multiscale edge detection algorithm based on wavelet domain vector hidden Markov tree model. *Pattern Recognition* 37, 7 (2004), 1315–1324.
- [177] TAN, X., BHANU, B., AND LIN, Y. Fingerprint classification based on learned features. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 35, 3 (2005), 287–300.
- [178] UNSER, M., ALDROUBI, A., AND EDEN, M. B-spline signal processing. Part I. Theory. *IEEE Transactions on Signal Processing* 41, 2 (1993), 821–833.
- [179] UNSER, M., ALDROUBI, A., AND EDEN, M. B-spline signal processing. Part II. Efficiency design and applications. *IEEE Transactions on Signal Processing* 41, 2 (1993), 834–848.
- [180] VAN VLIET, L. J., YOUNG, I. T., AND BECKERS, G. L. A nonlinear Laplace operator as edge detector in noisy images. *Computer Vision, Graphics and Image Processing* 45, 2 (1989), 167–195.
- [181] VERZAKOV, S., PAČLÍK, P., AND DUIN, R. Edge detection in hyperspectral imaging: Multivariate statistical approaches. In *Proceedings of the Joint IAPR International Workshops on Structural, Syntactic and Statistical Pattern Recognition* (2006), vol. 4109, pp. 551–5591.
- [182] VRIENT, J. Accuracy of the zero crossings of the second directional derivative as an edge detector. *Multidimensional Systems Signal Processing* 4, 3 (1993), 227–251.
- [183] WANG, J., AND TAN, Y. A novel genetic programming based morphological image analysis algorithm. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (2010), pp. 979–980.

- [184] WANG, S., KUBOTA, T., SISKIND, J. M., AND WANG, J. Salient closed boundary extraction with ratio contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005), 546–561.
- [185] WHIGHAM, P. A schema theorem for context-free grammars. In *Proceedings of the IEEE International Conference on Evolutionary Computation* (1995), vol. 1, pp. 178–181.
- [186] WU, P., AND CHEN, Q. A novel SVM-based edge detection method. *Physics Procedia* 24, Part C (2012), 2075–2082.
- [187] XIAO, Z., AND HOU, Z. Phase based feature detector consistent with human visual system characteristics. *Pattern Recognition Letters* 25, 10 (2004), 1115–1121.
- [188] XIE, H., AND ZHANG, M. Parent selection pressure auto-tuning for tournament selection in genetic programming. *IEEE Transactions on Evolutionary Computation* 17, 1 (2013), 1–19.
- [189] XU, P., MIAO, Q., SHI, C., ZHANG, J., AND LI, W. An edge detection algorithm based on the multi-direction shear transform. *Journal of Visual Communication and Image Representation* 23, 5 (2012), 827–833.
- [190] YAO, X., Ed. *Evolutionary Computation: Theory and Applications*. World Scientific, 1999.
- [191] YI, S., LABATE, D., EASLEY, G., AND KRIM, H. A shearlet approach to edge analysis and detection. *IEEE Transactions on Image Processing* 18, 5 (2009), 929–941.
- [192] YITZHAKY, Y., AND PELI, E. A method for objective edge detection evaluation and detector parameter selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 8 (2003), 1027–1033.

- [193] YU, Y.-H., AND CHANG, C.-C. A new edge detection approach based on image context analysis. *Image and Vision Computing* 24, 10 (2006), 1090–1102.
- [194] ZHANG, M., CAGNONI, S., AND OLAGUE, G. Evolutionary computer vision. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers* (2009), pp. 3355–3380.
- [195] ZHANG, M., AND SMART, W. Using Gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recognition Letters* 27, 11 (2006), 1266–1274.
- [196] ZHANG, Y., LI, H., NIRANJAN, M., AND ROCKETT, P. Applying cost-sensitive multiobjective genetic programming to feature extraction for spam e-mail filtering. In *Proceedings of the 11th European Conference on Genetic Programming* (2008), pp. 325–336.
- [197] ZHANG, Y., AND ROCKETT, P. I. Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection. In *Proceedings of the Conference on Genetic and Evolutionary Computation* (2005), pp. 795–802.
- [198] ZHANG, Y., AND ROCKETT, P. I. A generic multi-dimensional feature extraction method using multiobjective genetic programming. *Evolutionary Computation* 17, 1 (2009), 89–115.
- [199] ZHAO, Y.-Q., GUI, W.-H., CHEN, Z.-C., TANG, J.-T., AND LI, L.-Y. Medical images edge detection based on mathematical morphology. In *Proceedings of the 27th Annual International Conference of the Engineering in Medicine and Biology Society* (2005), pp. 6492–6495.
- [200] ZHENG, S., LIU, J., AND TIAN, J. W. A new efficient SVM-based edge detection method. *Pattern Recognition Letters* 25, 10 (2004), 1143–1154.

- [201] ZIOU, D., AND TABBONE, S. Edge detection techniques: An overview. *International Journal of Pattern Recognition and Image Analysis* 8, 4 (1998), 537–559.