

Genetic Programming for Cephalometric Landmark Detection

A thesis submitted for the degree of
Doctor of Philosophy

Andrew Innes B.Eng.

School of Aerospace, Mechanical and Manufacturing Engineering,
Science, Engineering, and Technology Portfolio,
RMIT University,
Melbourne, Victoria, Australia.

29th August 2007

Abstract

The domain of medical imaging analysis has burgeoned in recent years due to the availability and affordability of digital radiographic imaging equipment and associated algorithms and, as such, there has been significant activity in the automation of the medical diagnostic process. One such process, cephalometric analysis, is manually intensive and it can take an experienced orthodontist thirty minutes to analyse one radiology image. This thesis describes an approach, based on genetic programming, neural networks and machine learning, to automate this process. A cephalometric analysis involves locating a number of points in an X-ray and determining the linear and angular relationships between them. If the points can be located accurately enough, the rest of the analysis is straightforward.

The investigative steps undertaken were as follows: Firstly, a previously published method, which was claimed to be domain independent, was implemented and tested on a selection of landmarks, ranging from easy to very difficult. These included the menton, upper lip, incisal upper incisor, nose tip and sella landmarks. The method used pixel values, and pixel statistics (mean and standard deviation) of pre-determined regions as inputs to a genetic programming detector. This approach proved unsatisfactory and the second part of the investigation focused on alternative handcrafted features sets and fitness measures. This proved to be much more successful and the third part of the investigation involved using pulse coupled neural networks to replace the handcrafted features with learned ones. The fourth and final stage involved an analysis of the evolved programs to determine whether reasonable algorithms had been evolved and not just random artefacts learnt from the training images.

A significant finding from the investigative steps was that the new domain independent approach, using pulse coupled neural networks and genetic programming to evolve programs,

was as good as or even better than one using the handcrafted features. The advantage of this finding is that little domain knowledge is required, thus obviating the requirement to manually generate handcrafted features. The investigation revealed that some of the easy landmarks could be found with 100% accuracy while the accuracy of finding the most difficult ones was around 78%.

An extensive analysis of evolved programs revealed underlying regularities that were captured during the evolutionary process. Even though the evolutionary process took different routes and a diverse range of programs was evolved, many of the programs with an acceptable detection rate implemented algorithms with similar characteristics.

The major outcome of this work is that the method described in this thesis could be used as the basis of an automated system. The orthodontist would be required to manually correct a few errors before completing the analysis.

Declaration

This thesis contains work that has not been submitted previously, in whole or in part, for any other academic award and is solely my original research, except where acknowledged. The work has been carried out since the beginning of my candidature on 20 March 2001.

Andrew Innes

School of Aerospace, Mechanical and Manufacturing Engineering

RMIT University

29th August 2007

Acknowledgments

I would like to extend a thanks to my supervisors, Associate Professor Victor Ciesielski and Associate Professor Sabu John, for their feedback on research and the writing of this thesis. A special thanks to Vic's invaluable feedback throughout the drafting of this thesis, which has been a beneficial professional development exercise, which has greatly enhanced my technical writing skills as an engineer.

I would like to thank the many other people who contributed to the ideas presented throughout my candidature. This includes a consultant on the project, Dr. Alan Harvey, who provided expertise within the image processing domain. Others that provided assistance are Dr James Brusey and Nigel Smart, who augmented my learning curve through the embryonic stages of this project. A thankyou is extended to the members within the ECML group, RMIT University.

I am in appreciation of the support from the School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University, and the School of Computer Science, RMIT University, who provided me with the available resources to conduct and complete this piece of research.

Finally, I would like to thank family and friends who have provided moral support and will not have to ask the inevitable question of "have you finished yet?"

I am grateful for the financial support throughout my candidature from the Australian Research Council and industry partner, John Mamutil of Braces Pty Ltd. I am anticipating that the work presented in this thesis can be used as a bridge to automating the process for locating cephalometric landmarks.

Note

Unless otherwise stated, all fractional results have been rounded to the displayed number of decimal figures.

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Goals of the thesis	5
1.2.1	Research questions	5
1.3	Contributions	7
2	Literature review	9
2.1	Computer vision	9
2.1.1	Computer Vision in Medicine	10
2.1.1.1	Noise in X-ray images	11
2.2	Object detection	11
2.2.1	Performance measures	13
2.2.1.1	Weighted sum approach	15
2.2.1.2	Receiver operating characteristic curve	15
2.2.1.3	Multi-objective optimisation using a Pareto-optimal front	17
2.3	Machine learning	17
2.3.1	Types of learning	18
2.3.2	Current research with machine learning	19
2.3.3	Estimating error	19
2.3.4	Holdout	20
2.3.5	Cross-validation	20
2.4	Evolutionary Computation	21

2.5	Genetic Programming	22
2.5.1	Outline of Tree-Based Genetic Programming	24
2.5.1.1	Population Initialisation	26
2.5.1.2	Genetic Operations	28
2.5.2	Vision and image applications related to GP	29
2.5.2.1	GP applied to object detection problems	30
2.5.2.2	GP applied to image classification problems	32
2.5.2.3	GP applied to image processing problems	33
2.5.2.4	Terminal set	34
2.5.2.5	Function set	35
2.5.2.6	Performance measures using genetic programming	37
2.5.3	Parameters	39
2.5.3.1	Termination criteria	40
2.5.4	Some issues with using genetic programming	41
2.5.4.1	Improving program comprehensibility	42
2.6	Pulse Coupled Neural Networks	43
2.6.1	The PCNN model	44
2.6.2	PCNNs applied to image-related applications	45
2.7	Automatic cephalometric landmarking	46
2.7.1	Previous work in cephalometric landmarking	48
2.7.1.1	Traditional computer vision	48
2.7.1.2	Machine learning	50
2.7.2	A critical review of automated cephalometric landmarking	52
3	Data Preparation	56
3.1	What are Cephalograms?	56
3.2	Defining a Search Area Using a Heuristic	56
3.3	Image Processing	59
3.4	Pre-calculation of Feature Values	60
3.5	Dataset Selection	61

4	Domain Independent Approach: Pixels as Features	63
4.1	Introduction	63
4.2	Methodology	64
4.3	Configuration of Genetic Programming	66
4.3.1	The Terminal Set: Pixels as Features	66
4.3.2	The Function Set	68
4.3.3	Fitness Evaluation	68
4.3.4	Parameters	71
4.4	Results	71
4.5	Summary	80
5	Domain Dependent Approach: Handcrafted Shapes	81
5.1	Introduction	81
5.1.1	Chapter Goals	82
5.2	Methodology	82
5.3	Genetic Programming Configuration	83
5.3.1	The Terminal Set	83
5.3.2	The Function Set	84
5.3.3	Genetic Programming Parameters	84
5.4	Variations of Fitness Evaluation	84
5.4.1	Highest Output	87
5.4.1.1	Motivation	87
5.4.1.2	Results	89
5.4.2	Highest Output: Evaluating every second pixel position	96
5.4.2.1	Motivation	96
5.4.3	Binary Output	100
5.4.3.1	Motivation	100
5.4.3.2	Results	101
5.4.3.3	Discussion	102
5.4.4	Highest Output: Uncertain region	103

5.4.4.1	Motivation	103
5.4.4.2	Results	108
5.4.5	Highest output: Minimum Distance Error	109
5.4.5.1	Motivation	109
5.4.5.2	Results	111
5.5	Function Set	113
5.5.1	Motivation	113
5.5.1.1	Results	114
5.5.2	Analysis of a linear function set: $\{+, -\}$	119
5.6	Highest Output: Pixels as Features	121
5.6.1	Motivation	121
5.6.2	Results	122
5.7	Summary	125
6	Learning with Features from Pulse Coupled Neural Networks	127
6.1	Introduction	127
6.2	Can regions of interest be extracted using a segmentation algorithm?	128
6.2.1	Pulse Coupled Neural Network segmentation	128
6.2.1.1	Segmentation results	129
6.2.1.2	PCNN derived shapes	131
6.3	Genetic Programming: Learning from PCNN derived shapes	133
6.3.1	Motivation	133
6.3.2	Methodology	133
6.3.3	The Function Set	135
6.3.4	Fitness Evaluation	135
6.3.5	Case2: PCNN derived and Handcrafted shapes	135
6.3.5.1	Motivation	135
6.3.5.2	Results	138
6.3.6	Case 3: PCNN derived shapes only	143
6.3.6.1	Motivation	143

6.3.6.2	Results	143
6.3.7	Case 4: PCNN derived shapes and quadrants	145
6.3.7.1	Motivation	145
6.3.7.2	Results	146
6.4	Comparison of Feature Sets	148
6.5	Determining Input Window Size	153
6.5.1	Motivation	153
6.5.2	Results	154
6.6	Summary	155
7	Analysis of Evolved Programs	156
7.1	Genetic Programming Configuration	157
7.2	Fitness function	157
7.3	Terminal set	159
7.4	Function set	160
7.5	Results	161
7.5.1	Nose landmark	161
7.5.1.1	Case 1: ‘+’ operator	161
7.5.1.2	Case 2: ‘+, -’ operators	163
7.5.2	Sella landmark	172
7.5.2.1	Case 2: ‘+, -’ operators	172
7.6	Summary	184
8	Conclusions	186
8.1	Research Questions	186
8.2	Comparison with other work	188
8.3	Further Work	189
	Appendix	219
	A Parameter Settings	219
A.1	Parameter Settings: Genetic Programming	219

A.2	Parameter Settings: Pulse Coupled Neural Network	220
A.3	Cross Validation Results	221
B	Test results	222
B.1	Sella landmark	223
B.2	Nasion	226
B.3	A-point landmark	228
B.4	Incisal upper incisal landmark	229
B.5	Menton landmark	231
B.6	Nose landmark	233

List of Figures

1	Schematic of the process for extracting features within the search area of an X-ray	xxi
1.1	Cephalometric landmarks	2
1.2	Classifier for determining if the position is a landmark or background	4
1.3	Examples illustrating the correct and incorrect position of the incisal upper incisor landmark	5
2.1	Object detection	12
2.2	A Guideline for measuring performance using a ROC curve	16
2.3	A map of artificial intelligence	18
2.4	Representing a function as LISP S-expression and tree	23
2.5	Representation of tree-based, linear-based and graph-based structures	24
2.6	Schematic of the evolutionary process	26
2.7	Genetic operators	29
2.8	Shapes used for calculating features	35
2.9	The mapping of binary and multiple classes to a decision	39
2.10	A block diagram of the PCNN	44
2.11	Definition of cephalometric landmarks	47
3.1	A schematic of the heuristic used to define a search area	57
3.2	A schematic of an X-ray that has defined several search areas	58
3.3	Illustration of the soft tissue contrast	60
3.4	Schematic of the process for pre-calculating features	61

4.1	Three different landmarks of varying detection difficulty	64
4.2	Schematic of the basic methodology	65
4.3	Traversing of the input window across an image	67
4.4	Illustration for dividing an input window into square sub-regions	67
4.5	Strategy for predicting the position of a landmark	71
4.6	Comparison of average fitness between a 1×1 pixel and 2×2 pixel sub-regions	74
4.7	A sample program for locating the nose landmark	76
4.8	A sample program for locating the incisal upper incisor landmark	76
4.9	A sample program for locating the sella landmark	77
4.10	A selection of images showing the position of the predicted landmark on three different landmark types	78
4.11	Evolution of fitness using the domain independent approach	79
5.1	Schematic of the methodology for evolving detection programs using hand- crafted features	83
5.2	Features calculated from handcrafted shapes	85
5.3	Six different landmarks of varying detection difficulty	86
5.4	Evolved detection program for locating four types of landmarks	91
5.5	A selection of images showing the position of the predicted landmark on five different landmark types	92
5.6	Fitness graphs of programs applied to each pixel location	94
5.7	Detection performance of programs applied to training and testing data . . .	95
5.8	Illustration that compares the difference between evaluating each pixel and every second pixel	97
5.9	Fitness graphs of programs applied to every second pixel	99
5.10	Detection rate of programs applied to training and test data	100
5.11	Example of an ambiguous image with the corresponding output from a detec- tion program	105
5.12	Definition of the uncertain region	108
5.13	The effect of the <i>uncertain region</i> on detection rate and false alarm rate . . .	109

5.14	A comparison of fitness score for three function sets	117
5.15	An evolved linear detection program for locating the sella landmark	120
5.16	Frequency that terminal coefficients occur in linear programs	121
5.17	Illustration for calculating pixel based features within an input window	122
5.18	A comparison of average fitness scores for programs using features calculated from handcrafted shapes and pixels as features	124
6.1	Application of PCNN for segmenting regions of interest	130
6.2	Templates computed using the output from a PCNN	132
6.3	Schematic of the methodology for evolving detection programs using features calculated from the PCNN template and additional shapes	134
6.4	Substituting handcrafted for PCNN derived shapes	136
6.5	Calculation of features from the PCNN template and handcrafted shapes	137
6.6	A comparison of detection performance for <i>PCNN derived and handcrafted</i> <i>shapes with handcrafted shapes</i>	140
6.7	Images showing the predicted position of a landmark for four types of landmarks	141
6.8	Evolved detection programs for locating four types of landmarks	142
6.9	Features calculated from PCNN derived shapes	143
6.10	Features calculated from the PCNN derived shapes and quadrants	146
6.11	A summary of detection performance for four different feature sets.	151
6.12	A comparison of fitness scores between four different feature sets	153
6.13	An analysis of the input window size with respect to a program's detection performance	154
7.1	A graphical representation of a fitness function for optimising detection rate and program size.	158
7.2	The change in detection performance and size when using parsimony pressure	159
7.3	Feature sets that were used for analysing programs	160
7.4	Images where the nose landmarks are incorrectly identified	162
7.5	A graphical representation of a nose detection program's output when only the {+} operator is available for selection.	162

7.6	Equivalent programs for locating the nose landmark	165
7.7	A graphical representation of a nose detection program's output when only the {+, -} operators are available for selection.	167
7.8	Frequency that terminal coefficients occur in linear programs when applied to the nose landmark	169
7.9	Correlation between terminal coefficient and detection rate	170
7.10	A graphical representation of two sella detection programs' output when only the {+, -} operators are available for selection.	180
7.11	Equivalent linear programs for locating the sella landmark	181
7.12	Frequency that terminal coefficients occur in linear programs when applied to the sella landmark	183
B.1	A selection of images and their predicted positions for the sella landmark . . .	223
B.2	A detection program used to predict the position of the sella landmark	225
B.3	A selection of images and their predicted positions for the nasion landmark . .	226
B.4	A detection program used to predict the position of the nasion landmark . . .	227
B.5	A selection of images and their predicted positions for the A-point landmark . .	228
B.6	A detection program used to predict the position of the A-point landmark . .	229
B.7	A selection of images and their predicted positions for the incisal upper incisor landmark	229
B.8	A detection program used to predict the position of the incisal upper incisor landmark	230
B.9	A selection of images and their predicted positions for the menton landmark . .	231
B.10	A detection program used to predict the position of the menton landmark . .	232
B.11	A selection of images and their predicted positions for the nose landmark . . .	233
B.12	A detection program used to predict the position of the nose landmark	234

List of Tables

2.1	A confusion matrix	14
2.2	Summary of vision and image applications using genetic programming	30
2.3	Summary of function sets applied to vision and image applications	37
2.4	Fitness functions used for object detection	38
2.5	A range of GP run-time parameters that have been applied to image-related applications	40
2.6	Published detection results for automatically detecting cephalometric landmarks	54
4.1	The size of the terminal sets that are made available for use during the genetic search	68
4.2	Definition of operators used in the function set	68
4.3	Run-time parameters used during the genetic search	72
4.4	Comparison of fitness between 1×1 pixel and 2×2 pixel sub-region	73
4.5	Comparison of detection and false alarm rate between 1×1 and 2×2 sub-regions	75
5.1	Detection results using handcrafted shapes and highest output to locate the landmark	90
5.2	An investigation of detection performance for evaluating all pixels and every second pixel	98
5.3	Comparison of two methods for predicting the position of the landmark, i.e. highest output and binary output	102
5.4	Comparison of cumulative distance error and detection rate fitness functions .	112
5.5	Definition of operators	113

5.6	Definition of functions sets	114
5.7	A comparison of three function sets applied to three different types of landmarks	115
5.8	A Tukey's pairwise comparison of the detection performance of programs evolved from three function sets	116
5.9	A comparison of average detection rate from the top 10% of evolutionary runs for each function set	118
5.10	An ANOVA matrix comparing the average detection performance from the top 10% of evolutionary runs for each function set	119
5.11	Comparison of detection performance between domain independent and hand- crafted features	123
6.1	Summary of segmentation results using PCNN	131
6.2	Definition of four feature sets	134
6.3	A comparison of detection performance for programs that use <i>handcrafted</i> <i>shapes</i> with <i>PCNN + handcrafted shapes</i>	139
6.4	A comparison of detection performance for programs that use <i>handcrafted</i> <i>shapes</i> and <i>PCNN shapes</i>	145
6.5	A comparison of detection performance for programs that use <i>handcrafted</i> <i>shapes</i> and <i>PCNN derived shapes and quadrants</i>	148
6.6	Definition of four feature sets	149
6.7	A comparison of detection performance for four different feature sets	150
6.8	A Tukey's pairwise comparison of the detection performance of programs evolved from four feature sets	152
7.1	Operators that will exhibit the behaviour of linear functions	160
7.2	Frequency that a linear function has occurred for the nose landmark	164
7.3	Sample evaluations of a linear function used to locate a nose landmark	167
7.4	Sample evaluations of a linear function used to locate a sella landmark: reduced terminal set	175
7.5	Sample evaluations of a linear function used to locate a sella landmark	178

8.1	A comparison of our detection rates with results from the literature.	189
A.1	Run-time parameters used during the genetic search	219
A.2	Parameter settings used to generate the PCNN derived shapes	220
A.3	Templates computed using the output from a PCNN	220
A.4	Cross validation results for a range of landmark types	221

Glossary

Definitions

The following definitions are commonly used throughout this thesis and have been included to assist the reader.

Arity The number of arguments that are required to be given to a function.

Bloat A term given to describe the process of code growth over time.

Cephalogram Is a radiograph (also known as an X-ray) of the head, including the mandible, taken in full lateral view which is used for making cranial measurements.

Crossover Creates two programs for the new population by crossing over or swapping the sub-trees of two selected programs. The sub-tree in each program is created by randomly choosing a node.

Crossover rate The probability of creating two new individuals using the crossover operator.

Detection accuracy Is a measure of a program's ability to accurately locate the position of a landmark. Accuracy is measured as the difference between the detected landmark and the known position which is quantified in pixels.

Detection rate Is a ratio used to compare the relationship between the number of correctly detected landmarks with the total number of landmarks that are to be detected.

False alarm rate Is a ratio used to compare the relationship between the number of incorrectly detected landmarks with the total number of landmarks that are to be detected.

Features Refer to terminals.

Sub Image A heuristic based on anatomical knowledge is used to extract a smaller image from a digital X-ray, relative to a datum point. The rationale for the smaller image is to reduce the size of the search area when locating the landmark (refer to Figure 1).

Image dataset A selection of 110 digital X-ray cephalograms that are used throughout this thesis. Each image is an 8-bit greyscale image that has 256 levels of grey (refer to Figure 1).

Input window Contains a pre-defined set of shapes that are used to calculate features. The input window is traversed across each position on the sub image (refer to Figure 1).

Introns Code segments not contributing to a program's performance *or* irrelevant pieces of code that do not contribute to program fitness.

Maximum tree depth, initial The maximum depth of a tree at the initial generation.

Maximum tree depth The maximum depth of a tree after the initial generation.

Mutation Creates a new program by randomly choosing a node and introducing a new subtree into a program.

Mutation rate The probability of creating a new individual using the mutation operator.

Parsimony pressure is a popular bloat-control technique used to combat bloat in genetic programming. A size penalty is added to the fitness function.

Population size The number of individuals, or programs, in a given population.

Operators Are nodes with children that correspond to functions that are available in the function set.

Elitism Copies an individual from the current generation into the next generation, with the aim of allowing the fittest individuals to survive into the next generation.

Elitism rate The probability of creating a new individual using the reproduction operator.

Sensitivity An operating characteristic that measures the ability of a test to detect an object when it is present.

Shape The input window is divided into a set of shapes. Each shape is composed of a number of pixels which are used to calculate feature values. Each shape has two features, i.e. mean and standard deviation (refer to Figure 1).

Specificity An operating characteristic that measures the ability of a test to exclude an object when it is not present.

Terminals Are variables that are always leaves in the parse tree. In the context of using

genetic programming in object detection, terminals correspond to image features. Terminals are commonly referred to as features throughout this thesis and vice-versa.

Test set Is a collection of examples which were never used, or unseen, during training.

Training set Is a collection of examples that are used for learning a model during training

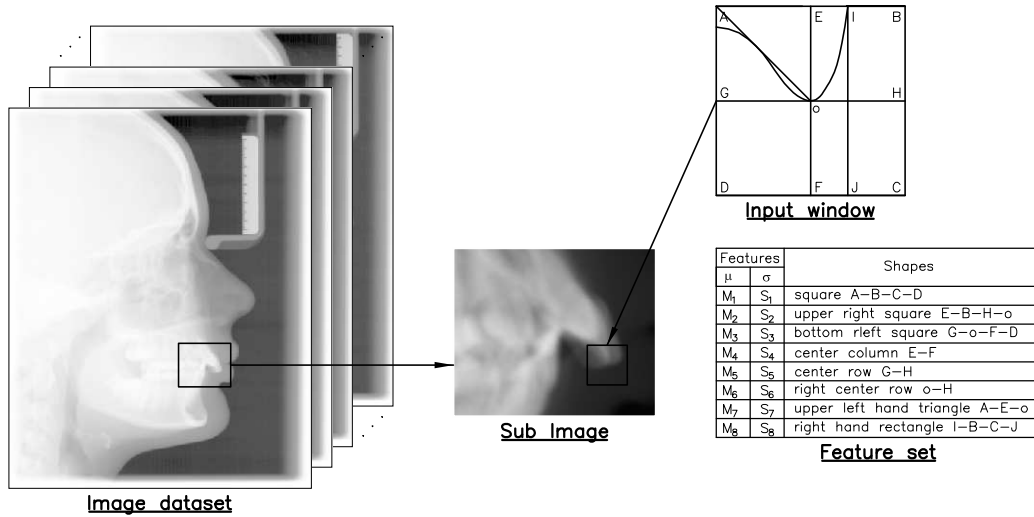


Figure 1: Schematic of the process for extracting features within the search area of an X-ray.

Acronyms and Abbreviations

ANOVA	analysis of variance
BNF	Backus-Naur Form
CPU	central processing unit
DR	detection rate
e.g.	<i>exempli gratia</i> , for example
et al.	<i>et alii</i> , and others
ERS	earth resource satellite
FAR	false alarm rate
FN	false negatives
FP	false positives
FSM	finite state machines
FSM(GP)	finite state machines with embedded genetic programs
GHz	gigahertz
GP	genetic program, genetic programming
i.e.	<i>id est</i> , that is
IR	infrared
IRLS	infrared line scan
LISP	list processing
LUT	look up table
min	maximum
max	minimum
MLP	multi-layered perceptron
NN	neural network
NMR	nuclear magnetic resonance
PCNN	pulse coupled neural network
ROC	receiver operating characteristic
RoI	region of interest
RS	remote sensing

SAR	synthetic aperture radar
SNR	signal-to-noise ratio
SS	square size
TN	true negatives
TP	true positives
w.r.t.	with respect to

Symbols

Chapter 1

P_i	Pixel intensity value of pixel i
-------	------------------------------------

Chapter 2

d	Maximum tree depth, initial - the maximum tree depth allowed in the initial generation
D	Maximum depth - the maximum tree depth allowed for programs
DR	Detection rate - a ratio expressed as the number of objects correctly located within the training/test set divided by the total number of actual objects within the training/test set
FAR	False alarm rate - a ratio expressed as the number of objects incorrectly located within the training/test set divided by the total number of objects in the training/test set
FN	False negative - the number of positive examples incorrectly classified
FP	False positive - the number of negative examples incorrectly classified
FP_i	The number of objects incorrectly classified in image i
f_i	Represents objective i
G	Maximum generations - used as a termination criterion to end the evolutionary process
M	Population size - the number of individuals in the population
n	The number of images in the training/test set
N_i	The number of objects in image i
TP	True positive - the number of positive examples correctly classified
N_p	The number of positives in the dataset
N_n	The number of negatives in the dataset
P_R	Probability of selecting a reproduction operator
P_C	Probability of selecting a crossover operator
P_M	Probability of selecting a mutation operator
TN	True negative - the number of negative examples correctly classified
TP_i	The number of objects correctly located in image i
w_i	A weight given to objective i

F_{jk}	Feeding compartment
L_{jk}	Linking compartment
S_{jk}	Input stimulus
U_{jk}	Internal state of the neuron
V_F	Magnitude scaling term for feeding
V_L	Magnitude scaling term for linking
V_T	Magnitude scaling term for threshold
Y_{kl}	Input pulse – output of neurons from a previous iteration
α_F	Decay term for feeding
α_L	Decay term for linking
α_T	Decay term for threshold
β	Linking strength

Chapter 3

c	A scaling constant, such that the maximum output value is 255 (assuming an 8-bit greyscale image)
$F(i, j)$	A pixel intensity value at position i, j in image F
$G(i, j)$	A pixel intensity value at position i, j in transformed image G
<i>Height</i>	The height of an X-ray which is expressed in pixels
<i>SS</i>	Square size - the dimensions of the input window which is expressed in pixels
<i>Width</i>	The width of an X-ray which is expressed in pixels
μ_x	The mean distance w.r.t. the x-axis which is calculated using the distance between the known landmark and the datum point from images within the training set
μ_y	The mean distance w.r.t. the y-axis which is calculated using the distance between the known landmark and the datum point from images within the training set
σ_x	The standard deviation w.r.t. the x-axis which is calculated using the distance between the known landmark and the datum point from images within the training set
σ_y	The standard deviation w.r.t. the y-axis which is calculated using the distance between the known landmark and the datum point from images within the training set

Chapter 4

A	A constant reflecting the relative importance of false alarm rate w.r.t. detection rate
B	A constant reflecting the relative importance of detection rate w.r.t. false alarm rate
d	Maximum tree depth, initial - the maximum tree depth allowed in the initial generation
D	Maximum tree depth - the maximum tree depth allowed for programs
DR	Detection rate - a ratio expressed as the number of objects correctly located within the training/test set divided by the total number of actual objects within the training/test set
FAR	False alarm rate - a ratio expressed as the number of objects incorrectly located within the training/test set divided by the total number of objects in the training/test set
<i>Fitness</i>	A measure of how well an individual performs within its environment
G	Maximum generations - used as a termination criterion to end the evolutionary process
H_0	Null hypothesis
H_1	Alternative hypothesis
INT_MAX	The maximum value for an object of type int
M	Population size - the number of individuals in the population
P_C	Probability of selecting a crossover operator
P_R	Probability of selecting a reproduction operator
P_M	Probability of selecting a mutation operator
P_i	The average pixel intensity value of a sub-region labelled P_i , where i represents the i th sub-region
SS	Square size - the dimensions of the input window which is expressed in pixels
α	The probability of rejecting the null hypothesis when it is true
$\mu_{1 \times 1}$	The mean fitness score when the input window is divided into 1×1 sub-regions
$\mu_{2 \times 2}$	The mean fitness score when the input window is divided into 2×2 sub-regions

Chapter 5

A	A constant reflecting the relative importance of false alarm rate w.r.t. detection rate
B	A constant reflecting the relative importance of detection rate w.r.t. false alarm rate
C	A constant
DR	Detection rate - a ratio expressed as the number of objects correctly located within the training/test set divided by the total number of actual objects within the training/test set
FAR	False alarm rate - a ratio expressed as the number of objects incorrectly located within the training/test set divided by the total number of objects in the training/test set
<i>Fitness</i>	A measure of how well an individual performs within its environment
INT_MAX	The maximum value for an object of type int
M_i and M_i	The calculated mean of grey level intensities within a pre-defined shape
<i>Output</i>	A real value returned by the computed program
$P_{Highest}$	The program's highest output in the image
P_{ij}	The program's output at position i, j in the image
P_{Lowest}	The program's lowest output in the image
S_i and S_i	The calculated standard deviation of grey level intensities within a pre-defined shape
<i>Threshold</i>	A lower limit to verify that other positions in the image do not have an output which is similar to the predicted position
(x_i, y_i)	The landmark's true position
(X_i, Y_i)	The landmark's predicted position
α_i	A coefficient of M_i
β_i	A coefficient of S_i
$\Delta rate$	Detection rate minus false alarm rate
ϵ	Cumulative distance error - the euclidean distance between the landmark's true position and the landmark's predicted position

Chapter 6

$A_k(i, j)$	The binary output from the PCNN
DR	Detection rate - a ratio expressed as the number of objects correctly located within the training/test set divided by the total number of actual objects within the training/test set
$Fitness$	A measure of how well an individual performs within its environment
I_n	Image cutout
M_i and M_i	The calculated mean of grey level intensities within a pre-defined shape
n	The number of images in the training set
S_i and S_i	The calculated standard deviation of grey level intensities within a pre-defined shape
$Template$	The average output using the binary output from the PCNN, $A_k(i, j)$
and	
$Template(i, j)$	
V_F	magnitude scaling term for feeding
V_L	magnitude scaling term for linking
V_T	magnitude scaling term for threshold
α_F	decay term for feeding
α_L	decay term for linking
α_T	decay term for threshold
β	Linking strength

Chapter 7

<i>DR</i>	Detection rate - a ratio expressed as the number of objects correctly located within the training/test set divided by the total number of actual objects within the training/test set
<i>Fitness</i>	A measure of how well an individual performs within its environment
M_i and M_i	The calculated mean of grey level intensities within a pre-defined shape
<i>Output</i>	A real value returned by the computed program
<i>Program Size</i>	The number of nodes in the program's tree
S_i and S_i	The calculated standard deviation of grey level intensities within a pre-defined shape
α_i	A coefficient of M_i
β_i	A coefficient of S_i
ϵ	Distance error - the euclidean distance between the landmark's true position and the landmark's predicted position

Symbols

\times	True position
\otimes	True position with allowable tolerance
\times	Predicted position
\times	False alarm
\square	Input window

Chapter 1

Introduction

1.1 Introduction

Advances and affordability in digital radiographic imaging have seen a requirement by orthodontists to automate the cephalometric analysis. A cephalometric analysis is composed from a defined set of landmarks or points, used to determine linear and angular relationships, that are located in both bony and soft tissue on a craniofacial X-ray. The cephalometric analysis is specific to an orthodontist for determining measurements and ratios based on the coordinates of the landmarks. Figure 1.1 shows an image indicating several types of cephalometric landmarks. The outcome of the analysis determines the type of treatment required for the patient. Currently the process of landmark identification is manually intensive and it can take an experienced orthodontist thirty minutes to analyse one X-ray.

Automating the cephalometric analysis is in the context of using a detection program to automatically locate landmarks, accurately enough for a cephalometric analysis, with no manual intervention. Rakosi [111] suggests an error of 2 mm is accurate enough for locating cephalometric landmarks. The error is defined as the difference between the position of the landmark automatically identified and the position if located by an experienced orthodontist.

The research that is addressed in this thesis is to develop a new framework using image processing techniques combined with machine learning to automatically locate landmarks from a lateral head X-ray. A learning approach is well suited to the problem because of the diversity of biological shapes that exist within a population. An advantage of using a

learning approach is that it potentially allows a practitioner to re-train or refine the position of a landmark.

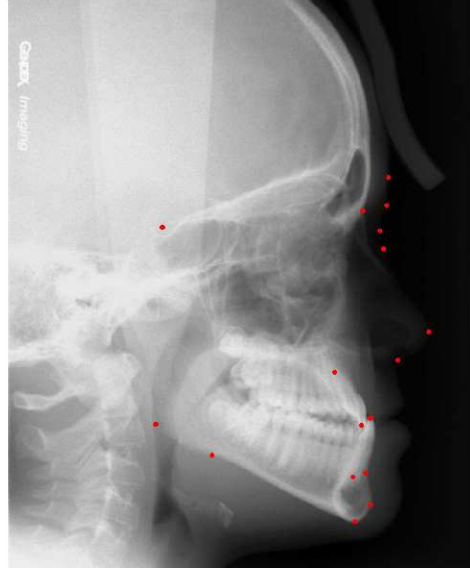


Figure 1.1: Cephalometric landmarks

A learning method that has shown promise for object detection problems is genetic programming. Genetic programming is an evolutionary search strategy from the evolutionary computation family, a field that uses mechanisms similar to biological evolution, for solving computational problems. The evolutionary search strategy is directed by increasing the likelihood that fitter programs partake in the evolutionary process, a process that is analogous to natural evolution. Evolutionary computation is an emerging area of research that has recently been applied to computer vision problems. The genetic programming method has shown potential to locate landmarks because, when applied to other problems of similar difficulty, the detection results have been promising and in some cases outperformed other learning paradigms such as neural networks [59, 166].

The overall aim is to develop an automated system to perform a cephalometric analysis. The intended purpose is to improve the efficiency of the treatment plan by reducing analysis time and allowing the orthodontist to focus on other work. Potential cost savings and potential improved diagnosis would offset any cost incurred by purchasing such a system.

Scope The aim of the work presented in this thesis is to develop a method to automatically identify the co-ordinates of cephalometric landmarks in digital X-rays that is accurate enough for a cephalometric analysis. The landmarks listed in Table 2.11 (p. 47) are located in both bony and soft tissue. Previous attempts by [20, 63] to automatically locate soft tissue landmarks have been reasonably successful, however, test results are considerably lower for landmarks located on bony tissue; in both cases, the test results are based on relatively small datasets. The work presented in this thesis will use a selection of landmarks that exhibit a range of detection difficulty (i.e. easy to hard) for the purpose of determining the likelihood that the proposed method will work on the entire list of landmarks in Table 2.11.

Realistically the proposed method will not locate landmarks in all grades of digital images, as the learning method will only be trained to locate landmarks for a predefined set of conditions such as signal-to-noise ratios. For example, if the method is trained to locate landmarks on digital radiographic imaging equipment then more than likely the method will not work when presented with digitised film X-rays. This ensures that the work presented in this thesis is clearly defined and that an umbrella is not created in an attempt to solve all cephalometric issues. Although the conditions for locating landmarks are predefined, there is no reason that a general method would not be able to learn how to locate a new type of landmark.

Domain Independent Approach A learning method that has shown promise for solving object detection problems is research conducted by Zhang et al. [164, 168] who presented a domain independent approach using genetic programming and pixels as features for classifying and locating the centres of coins. Zhang et al. subsequently applied the method to a difficult detection problem for detecting haemorrhages and microaneurisms in retina images. The detection performance of solutions from genetic programming was superior to other learning paradigms that included a neural network. The method by Zhang et al. uses a multi-class classifier, however, in the context of our work each landmark type will be treated as a separate detection problem. An advantage of this approach is that the computational requirements of a detector program are relatively inexpensive compared to other techniques such as those that use wavelet transforms.

The initial research in this thesis will investigate whether the domain independent approach described by Zhang et al. is able to locate cephalometric landmarks to a sufficient degree of accuracy for a cephalometric analysis. The later research will look at ways to modify and improve the basic approach. An outline of the domain independent approach for object detection is now given.

The location of the object of interest is found by traversing an input window across the image and evaluating a program at each pixel location. The program will take as input the values of the pixels in the input window or pixel statistics (such as mean and standard deviation) of simple shapes [8, 58, 59, 141, 165, 166]. The program will output a number greater than zero when centred on an object of interest or a number less than zero otherwise.

The desired program is obtained by an evolutionary process which uses a training set of marked up images and a measure of detection rate and false alarm rate as the fitness. The performance of each program within the population is measured and sorted according to fitness. A new population of programs is generated probabilistically by selecting programs to partake in genetic operations. The process for finding each type of landmark will be treated as a separate object detection problem.

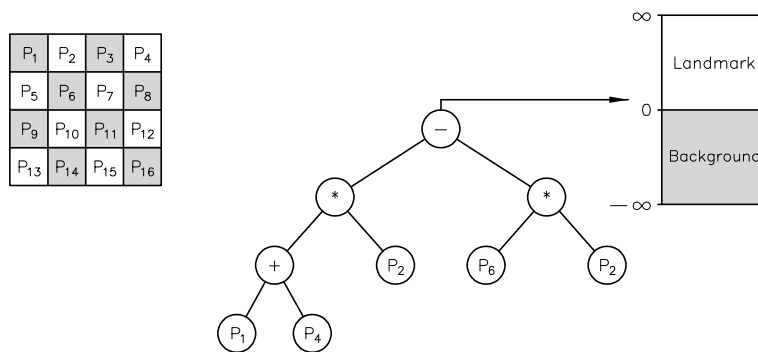


Figure 1.2: Classification strategy that uses the output of the program to determine if the pixel is a landmark or background. P_i represents the intensity value of pixel i

The images in Figure 1.3 illustrate four examples where the sweeping process that was described above was used to search for the incisal upper incisor landmark. The images are indicative of the large biological variability amongst four different patients. When the input window is applied to the images in Figure 1.3a-c the landmark was correctly found. The landmark was not correctly found in the example shown in Figure 1.3d. This is shown by the

disparity between the input window centred on the predicted position and the cross.

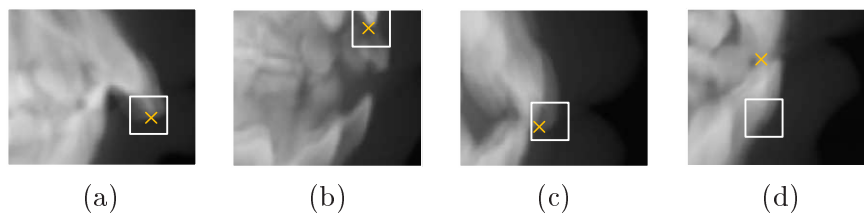


Figure 1.3: Examples illustrating the correct and incorrect position of the incisal upper incisor landmark. The input window is centred on the predicted position of the landmark and the cross indicates the actual location found by the orthodontist.

1.2 Goals of the thesis

The overall goal of the work presented in this thesis is to develop object detection programs for locating cephalometric landmarks using the genetic programming paradigm. The following questions will be explored and address issues associated with using genetic programming on landmark detection problems for a range of landmark types classified from easy to difficult.

1.2.1 Research questions

1. **Can the domain independent approach using pixels as features and genetic programming be used for landmark detection?**

The strategy that we have described above as the domain independent approach will determine if pixel based features are able to locate a range of landmarks from easy to hard to the desired level of accuracy for a cephalometric analysis. Formulating fitness is domain dependent, for example having false alarms may be acceptable in one problem (mammograms) and unacceptable in other problems (target detection). We investigate a formulation that suits the landmark problem.

2. **How can the domain independent approach be modified and extended to give better detection performance?**

This part of the investigation will address a number of fundamental issues for improving the detection of landmarks in complex images. This question will address:

- Can detection accuracy be improved by manually partitioning the input window

into a set of shapes, idiosyncratic to the landmark, and using pixel statistics (mean and standard deviation) of the shapes instead of pixel intensities.

- How can fitness be formulated to reduce the false alarm rate?
- Can a fitness function that uses detection error, defined as the euclidean distance between the predicted position and the true position, produce accurate detection programs?
- What operators should be incorporated into the function set so that programs are given the best chance of achieving a successful solution?

3. Can handcrafting of shapes be replaced by learning the shapes from examples and will this increase detection accuracy?

It is expected that handcrafted shapes used to compute features will be able to produce better detection programs compared to using pixels as features from the domain independent approach. However, the problem with determining handcrafted shapes is that knowledge of a landmark is required in order to determine suitable shapes and so the technique becomes a semi-automated approach. In this part of the investigation our aim is to develop a method that is able to automatically generate a set of shapes using the output from a pulse coupled neural network (PCNN) and determine if the detection programs produced are comparable to handcrafted shapes. The PCNN is a method that has shown some promise of segmenting regions of interest (ROI) from complex scenes. This part of the investigation will determine whether learned shapes are able to give better detection performance compared to the handcrafted shapes.

4. Are there any underlying algorithms that are learnt during the evolutionary process?

The aim of this part of the investigation is to determine whether the successful detection programs use some ad-hoc patterns in the training data or whether some understandable algorithm has been evolved.

1.3 Contributions

The contributions of this thesis are:

1. A framework for genetic programming applied to landmark detection, a difficult real world computer vision problem.

This work has delivered a feasible method for detecting cephalometric landmarks based on GP. Most GP applications in computer vision use simple or artificial images. However, our work has shown that GP can also be applied successfully to detect landmarks in complex real-world images.

2. Presentation for improving the GP framework

The work presented in Chapter 5 clearly demonstrated the benefit of reduction in false alarm rate by introducing handcrafted features, fitness adjustment and the creation of new operators for improving the performance of detection programs. By placing our work within an object detection context using GP, we expect that similar problems may make use of our contribution.

Part of this work was published in:

A. Innes, V. Ciesielski, J. Mamutil, S. John, and Alan Harvey. Landmark detection for cephalometric radiology images using genetic programming. In Ruhul Sarker, Bob McKay, Mitsuo Gen, and Akira Namatame, editors, in *Proceedings of the 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, pages 125-132, Canberra, November 2002.

V. Ciesielski, A. Innes, J. Mamutil, and S. John. Landmark detection for cephalometric radiology images using genetic programming. *International Journal of Knowledge Based Intelligent Engineering Systems*, 7(3):164-171, July 2003.

A. Innes, V. Ciesielski, J. Mamutil, and S. John, Reducing False Alarms Using Genetic Programming in Object Detection, in *Proceedings of International Conference of Artificial Intelligence*, Las Vegas, USA, June 2004.

3. A method for the automatic generation of image features for landmark detection

The work presented as part of this thesis has focused upon the automation of what is typically a manual process of ‘handcrafting’ features. Specifically, we have devised a general method for achieving this automation using a pulse coupled neural network and through experimentation, have discovered that our technique to discovering features is

comparable if not exceeding in performance with handcrafted features as well as other techniques presented in the literature.

Part of this work was published in:

A. Innes, V. Ciesielski, J. Mamutil, and S. John. Landmark detection for cephalometric radiology images using pulse coupled neural networks. In Hamid Arabnia and Youngsong Mun, editors, in *Proceedings of the International Conference on Artificial Intelligence (IC-AI'02)*, volume 2, pages 511-517, Las Vegas, June 2002. CSREA Press.

A. Innes, V. Ciesielski, J. Mamutil, and S. John. Finding templates for cephalometric landmark detection using pulse coupled neural networks and genetic programming. In Hamid Arabnia and Youngsong Mun, editors, in *Proceedings of the International Conference on Imaging Science, Systems and Technology (CISST'03)*, volume II, pages 511-517, Las Vegas, June 2003. CSREA Press.

4. A methodology for understanding evolved detection programs

We have developed a methodology for understanding the evolved programs. The methodology involved finding the underlying algorithm implemented in the evolved program and establishing that it is appropriate for the particular object detection problem. This gives confidence that underlying regularities are being captured in the evolved programs, not artefacts of the training data.

Victor Ciesielski, Andrew Innes, Sabu John and John Mamutil, "Understanding evolved genetic programs for a real world object detection problem", in *Proceedings of the 8th European Conference on Genetic Programming*, Maarten Keijzer, Andrea Tettamanzi, Pierre Collet, Jano I. van Hemert and Marco Tomassini, Eds., Lausanne, Switzerland, 30 Mar. – 1 Apr. 2005, vol 3447 of Lecture Notes in Computer Science, pp. 351–360, Springer

Chapter 2

Literature review

2.1 Computer vision

The problem of automatically extracting cephalometric landmarks from digitised X-Rays is a computer vision problem. Computer vision is a large and diverse field covering many different areas, such as navigation, remote sensing, character recognition and document processing, and medical imaging. Our particular problem fits within the area of medical imaging, however, this is not to say the work described in this thesis is limited to this domain.

The term computer vision has many definitions, but probably the most appropriate, based on our problem, is given by Shapiro and Stockman [130, p. 13] who defined the goal of computer vision as being able to “*make useful decisions about real physical objects and scenes based on sensed images*”. From this statement one may be lead directly to the question of how does a computer vision practitioner make useful decisions when presented with an image? This question is addressed by creating a description or model of the object in the image. As a result, Forsyth and Ponce [49, p. 13] have further generalised the definition of computer vision, and the view of many ‘experts’, by saying “*the goal of computer vision is the construction of scene descriptions from images*”.

Solving computer vision problems has a history dating back to the 1960s [49], however, relatively few researchers explored computer vision until the 1980s [69]. The catalyst for the increased research was the availability of affordable hardware meaning that algorithms that were once infeasible were now possible on relatively inexpensive workstations. Initially

machine vision problems in industrial vision were addressed using simplified binary image processing. However, with increased computing power, machine vision has progressed to greyscale image processing [41]. Nowadays, the research focus has changed from classical pattern recognition and image processing techniques towards knowledge-based techniques whereby during the past decade there has been a strong emphasis on developing computer vision systems that exhibit learning behaviour.

Because of the increasing complexity of computer vision problems, learning is seen as the next frontier in computer vision as is evident by the number of workshops dedicated to solving computer vision using machine learning [11, 12, 147]. The key advantage of learning is the ability to deal with new situations (improve responses over time) and it is not necessary to engineer a response, in advance, to every conceivable situation.

2.1.1 Computer Vision in Medicine

Computer vision in medicine is currently a very active area of research. The areas for research in medicine include morphometry (to quantify the physical characteristics of an object), visualisation of data, improved diagnosis, automatic processing of images and content-based image retrieval [97, 109]. A reason for the increased focus is to exploit the large number of 2D and 3D digital images generated by X-ray, computer topography, magnetic resonance, ultrasound and nuclear medicine imagery devices for diagnosis and therapy [4]. The field of computer vision applied to medicine is diverse and so we shall only describe problems closely related to our domain. The following gives a brief outline of research into problems relating to object detection and classification in medical X-rays.

Research in object detection and classification problems includes extracting the left ventricle from echocardiographic images [61], segmentation of normal and abnormal livers [62], detection and diagnosis of micro-calcifications in mammograms [27, 48, 149, 156, 163] and automatic detection of tantalum markers inserted into femurs [154]. Research relating to the automatic detection of cephalometric landmarks is presented in Section 2.7.

Because of the diversity of problems and their associated difficulty in the medical domain there are a large number of papers dedicated to topics that include pre-processing, such as image enhancement, noise reduction, edge detection and segmentation, classification and

localisation of objects.

2.1.1.1 Noise in X-ray images

A significant issue with digital X-ray images in computer vision is the impact of noise on image quality, i.e. image quality degrades quickly as noise increases [51]. System noise is measured as the *signal-to-noise ratio* (SNR). The main source of noise in X-rays is the random distribution of photons over the image [99]. The noise can be reduced by increasing the number of photons used to form the image. However, medical images are noisy due to the limitations on X-ray dose [115] because organic objects are sensitive to irradiation. Therefore, there has been a reasonable amount of research into reducing noise in medical X-rays with low SNR using a pre-processing step. These pre-processing steps include the use of wavelets [46, 140] and the Richardson-Lucy algorithm [82] for de-noising images. Other traditional image processing techniques for smoothing and enhancing images include median filters [146] and Gaussian kernels [115].

2.2 Object detection

Object detection is an area of computer vision and is defined as the task of determining if the object of interest is located in the image, and if so, determining the coordinates of the object's position. The most common approach for solving this problem, as described by Astrom [3], is to develop a classifier for distinguishing between two classes, i.e. object and non-object, and applying it to the image at different positions. A review of the object detection literature suggests that some of the work is restricted to only classification, although an important aspect of object detection is the determination of the position of an object. An example of this is an object detection algorithm developed by Winkeler et al. [157] for detecting faces in images. In this particular example, each pixel position in the image is classified as either a face or non-face, however, this approach does not identify the region of a face in an image.

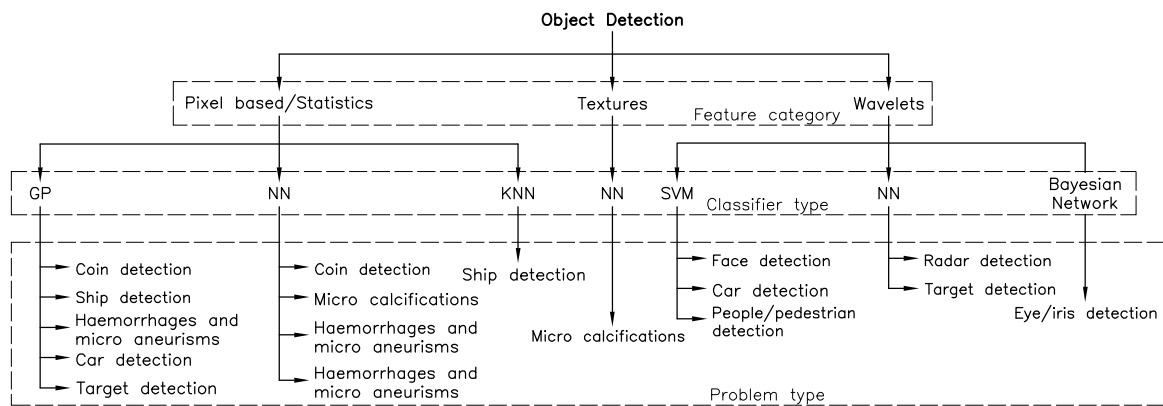


Figure 2.1: An example of features and classifiers used for solving object detection problems.

Generally speaking, most research performed on object detection problems involves two main steps: feature extraction and classification as shown in Figure 2.1. Key features are chosen during the feature extraction stage, with the goal to assist in discriminating between object and non-object, and used as inputs into the classifier. A classifier is chosen and developed, which uses the extracted features to discriminate between classes ‘object’ and ‘non-object’. It is not unusual for pre-processing, such as normalising or resizing an image, to occur prior to feature extraction. Depending on the object detection task, an image may contain multiple classes. The most common approach for solving object detection problems, even if the problem contains multiple classes, is to use a binary classifier, i.e. the classifier is designed to distinguish between object and non-object. If the problem involves multiple classes then it is not uncommon to have a different classifier for each class. However, Zhang et al. investigated a multiple class classifier for locating haemorrhages and micro aneurisms in retinal images [164, 166, 168].

As mentioned on page 10, using machine learning to develop a classifier is perceived as a promising approach for solving difficult computer vision problems. Difficult computer vision problems include applications in the medical domain because of the diversity in anatomy. Often a precise detection algorithm is required with a correct balance between false alarm rate and detection rate. This is important as images are often subject to subtle changes in greyscale, noise and background clutter. These issues cause difficulty and hence, require an approach that will have suitable features combined with a learning methodology for creating

a suitable function that will produce a robust object detection function. Both the types of features extracted and the classifier chosen are domain dependent, however, Zhang et al. proposed a method [164, 168] that claims to be domain independent.

2.2.1 Performance measures

In measuring the performance of object detection systems the most common measures of performance are (1) detection rate, which we wish to maximise and (2) false alarm rate, which we wish to minimise. Detection rate and false alarm rate are determined using Equations 2.1 and 2.2 respectively, however, it is not uncommon to express either of the terms as a percentage.

Detection Rate (DR) refers to the number of objects correctly located by the system, known as *true positives* (TP), as a ratio of the total number of objects, N , in a dataset containing n images. The detection rate is a value between 0 and 1.

$$DR = \frac{\sum_{i=1}^n TP_i}{\sum_{i=1}^n N_i} \quad (2.1)$$

where,

TP	The number of positive examples correctly classified
N	The number of objects in the image
n	The number of images in the training/test set

False Alarm Rate (FAR) refers to the number of objects incorrectly located, known as *false positives* (FP), as a ratio of the total number of objects. Unlike detection rate, the false alarm rate can be a value greater than 1, however, ideally the system will produce a false alarm rate with a value of 0, i.e. no false alarms.

$$FAR = \frac{\sum_{i=1}^n FP_i}{\sum_{i=1}^n N_i} \quad (2.2)$$

where,

FP	The number of negative examples incorrectly classified
----	--

In the confusion matrix, a concept originally borrowed from medical diagnosis, true negatives (TN) are the number of negative examples correctly classified and false positives (FP) are the number of negative examples incorrectly classified as positive. False negatives (FN) are the number of positive examples incorrectly classified as negative while true positives (TP) are the number of positive examples correctly classified [23]. To better understand the terms ‘true positive’ and ‘false positive’ and their relationship to the predicted outcome, refer to Table 2.1.

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Table 2.1: A confusion matrix illustrating the relationship between actual and predicted classifications performed by a classification system.

The ultimate goal of object detection is to achieve a detection rate of 100% and false alarm rate of 0%. However, achieving the ultimate goal for difficult detection problems is sometimes not possible and so a trade-off between detection rate and false alarm rate is required. Determining a trade-off is problem dependent, for example producing false alarms during medical diagnosis is seen as acceptable since the practitioner would perform a further investigation as part of the treatment. However, a false alarm on an automatic targeting system, where there is no human input, will cause the system to fire the weapon. These examples demonstrate the reason why it is important to determine the balance between detection rate and false alarm rate when measuring the performance of a system based on the consequences of the decision.

In object detection, the background artefacts often significantly outweigh the number of objects in the image, potentially resulting in a highly skewed data set. This may cause an object detection algorithm to be biased towards detecting background artefacts as objects, which is undesirable. Unfortunately, a highly skewed dataset is something that is not uncommon in real world problems [47]. In the presence of highly skewed datasets, the detector will be biased towards either poor detection performance or inadvertently detecting non-objects.

The detection rate and false alarm rate may be treated as a multi-objective optimisation

problem. The following gives an overview of multi-objective approaches commonly used for measuring the performance of object detection and classification performance.

2.2.1.1 Weighted sum approach

The weighted sum approach adds the multiple objectives together using different weights, w_i , for each objective, f_i . The value of each weight represents the relative importance of each objective [28]. The approach of combining the objectives into a single function means the multi-objective problem is transformed into a scalar optimisation problem. This approach has been used to measure the performance of learning algorithms including neural networks, genetic algorithms and genetic programming, for object detection problems [168].

$$\min \sum_{i=1}^k w_i f_i \quad (2.3)$$

There are other approaches, similar to the weighted sum approach, where the multi-objective problem is transformed into a scalar optimisation problem [8, 58, 167]. These are referred to as ‘aggregating functions’ because the objectives are combined to produce a single fitness function [65]. An advantage of aggregating functions is that they are easily adapted to learning algorithms such as genetic algorithms and genetic programming since they are a scalar optimisation technique [28]. Aggregating functions are generally more computationally efficient than other multi-objective techniques [65]. A disadvantage of this approach is that a priori information is required to determine the weights in order to evaluate the performance of the classifier [77]. If the assignment of weights is not ideal then one of the objectives will become dominant. For example, a bias exists when the number of objects to be detected is much smaller than the total number of non-objects within an image. If the weights are not balanced to account for the bias between objects and non-objects then more than likely the measure of performance will become dominated by an objective.

2.2.1.2 Receiver operating characteristic curve

The receiver operating characteristic (ROC) curve, commonly referred as ROC curve, originated from the evaluation of radar operators that was adopted for the diagnosis of tests

followed by the machine learning community [129]. An ROC analysis is seen as an alternative technique to evaluate how well classifiers perform given a distribution of two classes [13]. The ROC curve is a graphical representation of the trade off between true positive and false positive rates as a function of varying classification threshold [129]. Alternatively, ROC curves are expressed as a trade-off between sensitivity and specificity as illustrated in Figure 2.2. ROC curves have been used to measure the performance of object detection systems that include neural networks and support vector machines [48, 98, 103, 131].

$$\text{Sensitivity} = \frac{TP}{N_p}, \text{ where } N_p \text{ is the number of positives in the dataset} \quad (2.4)$$

$$\text{Specificity} = 1 - \frac{FP}{N_n}, \text{ where } N_n \text{ is the number of negatives in the dataset} \quad (2.5)$$

The area under the curve, as shown in Figure 2.2, measures the probability of correct classification [54]. ROC curves present an attractive way of measuring the performance of machine learning algorithms [16] and have been applied as a fitness measure for genetic algorithms [128]. The approach is seen as advantageous when choosing an optimal point on the curve. The point lying on the convex hull of the ROC curve is chosen as the optimal classifier/detector [23] as apposed to manually choosing a classification threshold.

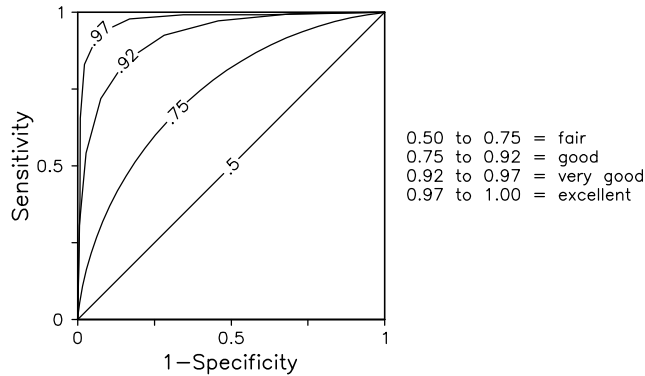


Figure 2.2: A guideline for determining performance of an algorithm based on the area under the ROC curve.

Agarwal et al. in [1] suggest another approach to ROC curves for object detection problems. The distinction between the two approaches being an alternative x-axis measure, i.e

$1 - \textit{Specificity}$ is replaced with $1 - \textit{Precision}$. Agarwal et al. in [1] justify the difference in approach for object detection problems because the number of negatives in the dataset is not known and so specificity cannot be calculated.

$$\textit{Precision} = \frac{TP}{TP + FP} \quad (2.6)$$

2.2.1.3 Multi-objective optimisation using a Pareto-optimal front

A new area receiving a lot of interest for evaluating the performance of classification problems is multi-objective optimisation using a Pareto-optimal front to find non-dominated solutions. The Pareto front is a collection of solutions that have no superior and are referred to as non-dominated solutions [65]. Solutions are said to be non-dominated if the solutions do not perform better with respect to both objectives [36]. The Pareto-optimal set corresponds to points on the ROC curve [77]. A single solution is selected from those solutions along the Pareto-optimal front.

An advantage of this approach over the aggregated functions, as described in 2.2.1.1, is that the ambiguity is removed with regards to preference of the objectives [77]. Kupinski et al. [77] demonstrated that the Pareto-optimal front, optimised by a genetic algorithm, was comparable to or better than the ROC curves for a given dataset and classifier. This approach is not restricted to two classes and the generation of the ROC curves can be performed within a single task.

2.3 Machine learning

For complex problems it is often too difficult to encode the necessary behaviours and intelligence to solve such problems. Therefore, it can be more feasible to implement a machine learning algorithm so that the desired behaviour of the system can be learned. One goal of machine learning is to program computers using example data to solve a given problem. Conceptually, machine learning can be viewed as a search defined by some underlying representation (e.g. linear functions, logical descriptions, decision trees and neural networks).

Machine learning, an area that overlaps with statistics, is a subset of artificial intelligence

(refer to Figure 2.3) relating to the application of learning methodologies that allow computers to learn. An overall definition of learning, as given by Witten et al. [158], is “*things learn when they change their behaviour in a way that makes them perform better in the future*”. One form of machine learning involves learning from training data. The goal of such an approach is to apply knowledge gained from the training stage to unseen cases.

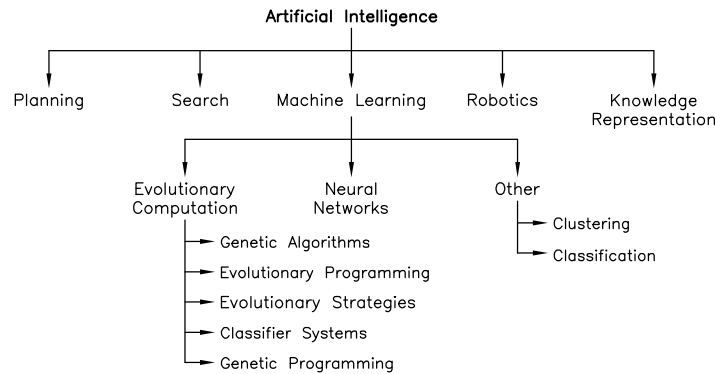


Figure 2.3: A map of artificial intelligence and its relationship to evolutionary computation.

2.3.1 Types of learning

The methods for learning from data can be categorised into the following types of learning: *supervised* and *unsupervised*. The following gives a brief outline for each of these types of learning.

Supervised learning: This is the most common learning category. Supervised learning uses training data that consists of inputs and their associated outputs to develop knowledge or rules that are able to predict the output associated with unseen input or test data.

Unsupervised learning: This is commonly associated with cluster analysis algorithms. The most significant difference with supervised learning is there is no a priori output associated with unsupervised learning.

For both supervised and unsupervised learning we want to learn a function, $y = f(x)$, where x is an input and y is the output. However, supervised learning implies that a set of (x, y) pairs are given, whereas in unsupervised learning only a set of (x) are given.

2.3.2 Current research with machine learning

The following is an outline of issues associated with learning from data.

- Deciding what type of algorithm will give the best approximation to the function, i.e. what is the best way to represent the knowledge.
- How many training examples are sufficient to learn a concept.
- Is the algorithm scalable with respect to increasing the number of training examples, features or the number of classes.
- How to control overfitting of the learned function (discussed in Section 2.5.4).

2.3.3 Estimating error

The goal of learning an object detector/classifier from sample data is to successfully predict when presented with new data. The most common measure of success is the classifier's error rate (refer to Equation 2.7), however, a more precise measure is the true error rate. The true error rate is statistically defined as the error rate of the learnt object detector when applied to a large number of new cases – as the number of test cases increase the error rate will give a better estimate of true error rate. However, in many real world problems and specifically our problem, the number of samples in the dataset is relatively small. Therefore, if we only have a finite selection of data then how do we best estimate true error rate?

$$\text{Error rate} = \frac{\text{number of errors}}{\text{number of cases}} \quad (2.7)$$

A common approach that is used for both learning and measuring performance when a small set of examples are made available from an infinitely large population is to divide the samples into two groups. The set of examples are also referred to as the dataset. The group used for learning is known as the *training set* and the other group used to measure performance is known as the *test set*. It is important that the examples selected for the two groups are randomly selected from the dataset. The *Training Set* is a collection of examples that are used for learning a model during training. The *Test Set* is a collection of examples

which were never used, or unseen, during training. The test set is used for measuring the performance, or generalisation, of the final model that was learnt as a result of training. In general, the training set is used for learning the model, while a test set is used for measuring the performance of that model.

When selecting the size of the groups, how should the number of examples be proportioned to training and test? The following is an outline of common methods that are used to provide an estimate of error when only a small number of examples are available in the dataset. The estimate is often used as a means of comparing between different approaches to learning the problem. Based on the review from the literature, a less common approach to sampling data is the bootstrapping method [155].

2.3.4 Holdout

The holdout method is a single application of training and test sets which is typically used when a large number of examples are available in the dataset. In order to have sufficient samples to learn, it is not unusual for the training set to be larger than the test set. For example, the training and test set may be proportioned 2/3 and 1/3 of the total dataset respectively.

2.3.5 Cross-validation

A method known as leave-one-out consists of $(n - 1)$ samples for training and applying the remaining sample as a test, where n is the number of samples in the dataset. This process is repeated n times, with training and testing occurring on a different sample. This method provides a good approximation of true error rate, however, the method is computationally expensive because it is repeated n times.

A less computationally expensive method is the k -fold cross-validation. This method randomly divides the dataset into k test partitions. The train and test process is repeated k times and each time using a different test partition for test. An advantage of using cross-validation is that all of the available samples are used for testing, and a large proportion of samples are available for learning. Typically, leave-one-out is preferred over k -fold cross-validation when the dataset consists of 30 or fewer samples [125].

2.4 Evolutionary Computation

Evolutionary computation is a powerful search strategy based on biological evolution, for solving optimisation and other problems. The principal idea is that individuals from a population are allowed to generate offspring by means of mutation, mating, and other genetic operators. The fitness of an individual is based on the how well the individual solves the problem. The fitter individuals are allowed to survive and participate in future generations in a process analogous to natural selection, and they in turn generate their own offspring whereby the whole process iterates until a desired solution to the problem is reached.

There are several well known algorithms based on this process. These include genetic algorithms, evolutionary programming, evolutionary strategies, learning classifier systems and genetic programming.

Evolutionary strategies differ from genetic algorithms because they only deal with real-coded problems, whereas genetic algorithms can solve complex combinatorial problems, or mixed valued problems. Evolutionary strategies also provide the facility to self-adapt their control parameters, such as mutation rate which can assist in escaping from local minima. Both genetic algorithms and evolutionary strategies have borrowed features from each other and over recent years the distinction between the two has become blurred.

Typically genetic algorithms and evolutionary strategies use vectors to represent individuals, whereas genetic programming uses tree or stack based structures to represent computer programs, unlike evolutionary programming, which is related to genetic programming but the representation of the program is a state-machine. Learning classifier systems use a population of binary rules from which a genetic algorithm alters and selects the best rules. The utility of a rule in this approach is decided by a reinforcement learning procedure, instead of a measure of fitness.

All of these approaches have the basic evolutionary principles in common; they use a population of solutions, the solutions are perturbed in some manner to generate offspring, and the fitter offspring are selected for the next generation.

More recently other population based approaches have been developed such as particle swarm optimisation, differential evolution, cultural algorithms, artificial immune system ap-

proaches and ant colony optimisation.

2.5 Genetic Programming

Genetic programming encompasses a family of evolutionary algorithms popularised by Koza [73] from 1989. This seminal paper describes a hierarchical genetic algorithm that allows a program to be evolved. Such a program is most easily expressed as a LISP (LISt Processing) S-expression. Koza's [73] inspiration for developing a method that uses LISP S-expressions is that many problems in artificial intelligence can be thought of as executing a procedure for performing a task that could easily be expressed as a LISP program.

As with other evolutionary algorithms, the genetic programming method shares similarities with the Darwinian principle of survival of the fittest. Initially, a population of programs, or individuals, are created and ranked according to a fitness measure and depending on the success of these programs, the programs will then be allowed to participate in reproduction with other programs to produce a new generation. The idea is that over time the fitness of programs will improve where only the fittest will survive. This is akin to the Darwinian principle of natural selection.

The major difference between genetic programming and conventional genetic algorithms is the representation of the problem. A genetic algorithm typically uses a fixed length string where each bit is assigned a meaning. However, in its original formulation a genetic program is represented as a variable length LISP S-expression that can be interpreted as a program. Banzhaf et al. [7] state that due to the flexible nature of GP, it is theoretically possible to evolve any solution that can currently be produced by conventional machine learning mechanisms.

Representation of solutions A solution in genetic programming is commonly represented as a LISP S-expression that can also be depicted as a tree-based structure. An example of a LISP S-expression with the corresponding tree is shown in Figure 2.4. The functions and terminals are the two fundamental elements used for the creating a tree. The terminals correspond to leaves, nodes without branches, that represent a variable or a constant value.

The functions are nodes with children that correspond to operators and functions that are available in the function set. The arity of a function is the number of arguments (children), or inputs, required to be given to that function.

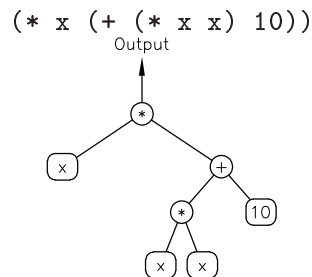


Figure 2.4: Representing a function, $x(x^2 + 10)$, as a LISP S-expression (above) with the corresponding tree (below).

Other representations of a genetic program Generally speaking, the most popular approach for structuring a genetic program, based on published literature, has been in the form of a tree-based structure. However, some of the other approaches reported in the literature for structuring a genetic program include a linear-based structure [6, 17, 30], a graph-based structure [96, 143], a linear-tree structure [70], a linear-graph structure [71] and grammatical evolution [102].

Linear-based structure The program of a linear-based structure consists of four parts: the header, body, footer and return instruction. A , B , C are registers and RO is the register which is used to return the output at the end of the program. The header and footer do not partake in the evolutionary process. Programs using a linear-based structure are represented as binary strings which are manipulated and executed without interpretation using a normal genetic algorithm. An example of a linear-based program is shown in Figure 2.5 along with the equivalent tree-based program.

Graph-based structure A program of a graph-based structure consists of N nodes in a directed graph with up to N arcs going out from each node. Each node consists of an action part and a branch-decision part. There are also special nodes that include *start* and *end* nodes that indicate the start and end of the program, and special nodes may also be

in the form of subprogram or library calling nodes. An example of a graph-based program is shown in Figure 2.5. A method similar to the graph-based structure is cartesian genetic programming by Miller and Thomson [96] that considers a grid of nodes that are addressed in a cartesian coordinate system.

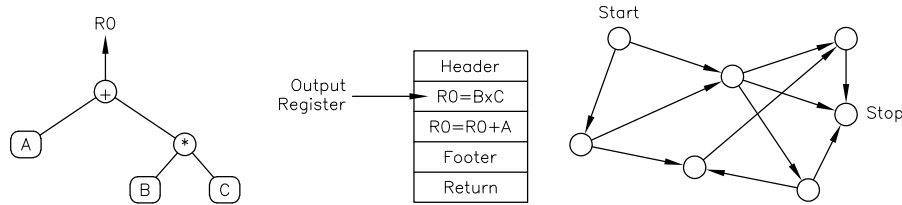


Figure 2.5: Representation of a tree-based (left), linear-based (middle) and graph-based (right) structures used in genetic programming.

Grammatical evolution An approach in which a genetic algorithm is used to evolve programs. The individuals in the population are binary strings that undergo binary selection, crossover and mutation. A program is generated from an individual by decoding the individual into a sequence of applications of rules from BNF grammar.

2.5.1 Outline of Tree-Based Genetic Programming

The following description outlines the evolutionary process for tree-based genetic programming. The process is initiated by generating a population of random programs created from functions and terminals that are made available for selection as part of the evolutionary process. New populations of programs are created using genetic operations that are analogous to evolution. At the end of each generation the performance of each program is measured according to a fitness metric. This iterative process is continued until some termination criterion is satisfied.

Each phase of the genetic program framework is explained as follows:

1. Generate an initial random population of programs created from a selection of functions and terminals.
2. The evolutionary process iteratively steps through the following:

- (a) Measure the performance of each individual program against a fitness metric and rank each program according to fitness.
 - (b) Determine if the termination criterion is satisfied. If a termination criteria has been satisfied then proceed to step (3), otherwise, proceed to step (c) - creating a new generation.
 - (c) A new population of programs is generated by probabilistically selecting individual programs from the current population to partake in genetic operations. Better individuals have a higher probability of selection. P_R , P_C and P_M are the probability that an individual(s) will be created using reproduction, crossover or mutation respectively.
 - i. Reproduction: Copy an individual program to the new population, typically if the program is fit. This may also be referred to as elitism.
 - ii. Crossover: Creates two programs for the new population by crossing over randomly chosen parts from two selected programs.
 - iii. Mutation: Creates a program by mutating or introducing a randomly chosen part of a selected program.
3. At the end of the evolutionary run, the best program is typically selected as the outcome of that run.

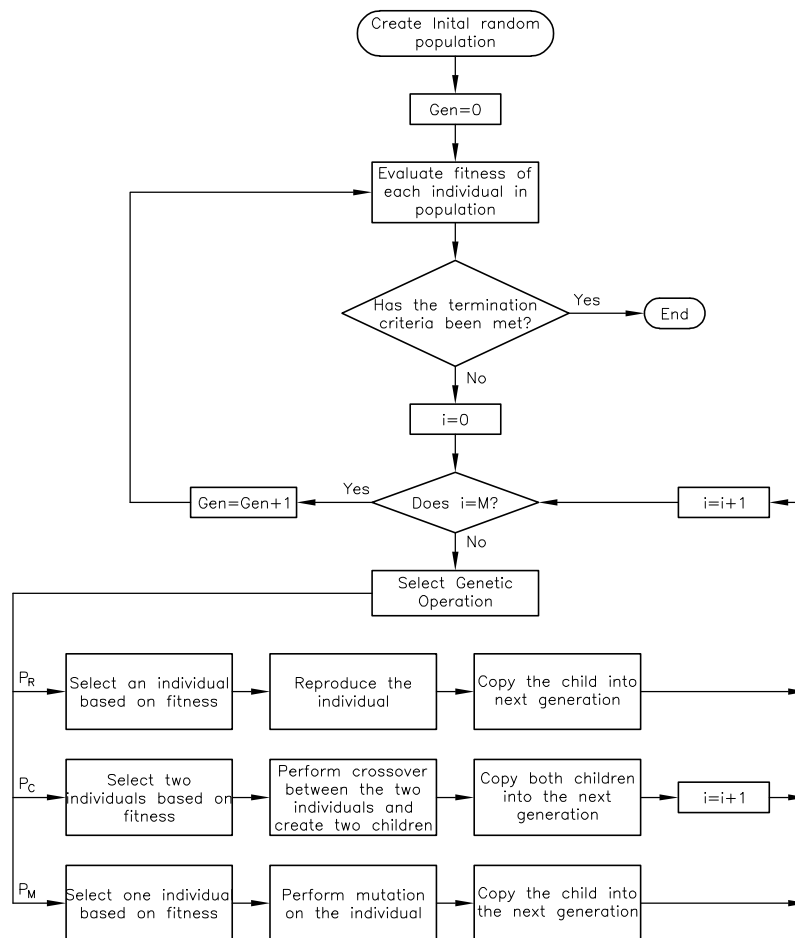


Figure 2.6: This schematic outlines the process for an evolutionary run.

Population initialisation and genetic operators will be explained in greater detail.

2.5.1.1 Population Initialisation

Population initialisation is the term given when programs are created in the initial population. The majority of researchers in GP use the *ramped half-and-half* method outlined by Koza in [74]. Other work includes the *uniform* method by Böhm et al. [14] who claims a superior performance to the ramped half-and-half initialisation method. Luke and Panait [91] compared five major tree generation algorithms on two problems and concluded that there was no difference in performance between the algorithms. However, results on a third problem indicate that the uniform method has an inferior performance to the ramped half-and-half method.

The *ramped half-and-half method* creates a set of trees for the initial population. The process by which these trees are created is that given a maximum and minimum depth, $M/(maximum - minimum + 1)$ trees are created at each depth. Half the trees at each depth are created using the full method and the other half using the grow method. The *full method* always creates a full tree to the computed tree depth. The *grow method* however, creates randomly shaped trees that do not exceed the computed tree depth. It is not uncommon for the initial population to be restricted to a smaller maximum depth compared with programs that are evolved during the evolutionary search.

The *uniform method* takes a single pre-defined tree-size, and guarantees that it will create a tree chosen uniformly of that tree size.

2.5.1.2 Genetic Operations

In genetic programming there are three type of genetic operators used for creating individuals for the new generation. These are reproduction, crossover and mutation. The choice of operator is determined probabilistically and generally speaking $P_C \gg P_M$ and $P_R \approx 0 - 0.1$.

Reproduction is the copying of an individual from the current generation into the next generation. If a small number of the best individuals are copied the process is called elitism. Elitism guarantees monotonic improvement in fitness.

Crossover creates two programs for the new population by crossing over or swapping sub-trees of two selected programs (refer to Figure 2.7a). The offspring are created by randomly choosing a node from each parent program and swapping the sub-trees between the nodes (refer to Figure 2.7a).

Mutation creates a new program by randomly choosing a node and introducing a new sub-tree into the program at that node (refer to 2.7b).

The theory for justifying the probabilities for crossover and mutation in genetic algorithms does not apply to genetic programming because the tree-based genome is significantly different to the vector-based genome [92]. Since theory for choosing probabilities for crossover and mutation is lacking and research by Luke et al. [92] concluded “why one is preferable to the other is dependent on domain and parameter settings”, our decision for crossover and mutation probabilities will be based on literature in genetic programming applied to image-related applications (refer to Table 2.5).

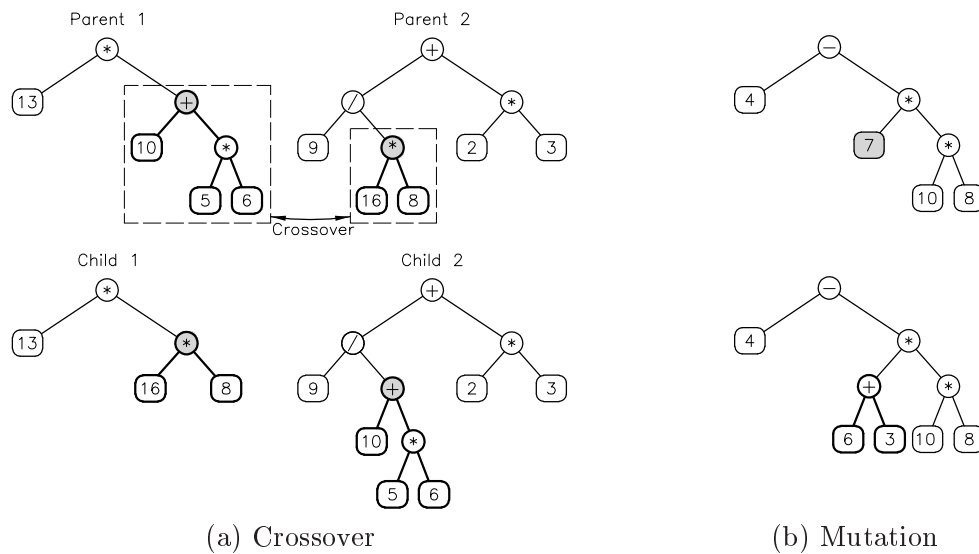


Figure 2.7: Genetic operators: Crossover and mutation

Five major steps that require consideration prior to applying genetic programming to a problem are outlined by Koza [75]. These steps are:

1. Terminal set
2. Function set
3. Fitness measure
4. Parameters
5. Termination criteria

Most of these steps are specific to a particular problem domain. Each step will be addressed in Section 2.5.2 by focusing on literature relating to genetic programming applied to image-related applications.

2.5.2 Vision and image applications related to GP

This section is a survey of literature on genetic programming related to object detection and other applications in the image domain. The literature has been divided into sections relevant to a problem domain and sorted in order of relevance to our specific problem. The most relevant problem domain is object detection followed by image classification and image

processing. A summary of the literature is shown in Table 2.2. It is worth noting that generally object detection is an extension of classification.

Domain	Application	Source	Year
Detection	Detection of vehicles in IRLS images	Howard et al. [58, 60]	2006, 1999
	Detection of simple objects	Roberts et al. [119]	2004
	Object detection in retinal images	Zhang et al. [164, 165, 166]	2003, 2000, 1999
	Detection of ships in SAR images	Benson et al. [8, 9]	2000
	Detection of ships in SAR images	Howard et al. [59]	1999
	Target detection	Tackett [141]	1993
Image classification	Mineral classification	Ross et al. [123, 124]	2005, 2002
	Image texture feature extraction	Lam et al. [79]	2004
	Texture classification	Song et al. [133, 135]	2003, 2002
	Text/picture classification	Agnelli et al. [2]	2002
	Digit recognition	Teredesai et al. [144]	2002
	Spectral imagery	De Falco et al. [33]	2002
	Class'n of hyper-spectral imagery	Rauss et al. [114]	2000
	Facial recognition	Winkeler et al. [157]	1997
	Classification of SAR images	Daida et al. [31, 32]	1996, 1996
	Classification of brain tumours	Gray et al. [53]	1996
	Class'n of remote sensing imagery	Riolo et al. [116]	1995
	Object classification	Teller [142]	1995
Image processing	Color constancy	Ebner [38]	2006
	Mathematical morphology	Quintana et al. [110]	2006
	Impulse noise filter	Petrović et al. [106]	2005
	ROI extraction	Bhanu et al. [10]	2004
	Text Segmentation	Rivero et al. [117]	2004
	Segmentation	Lin et al. [84]	2002
	Thresholding	Rosin [121]	2001
	Edge detection	Ross et al. [122]	2000
	Edge detection	Lucier et al. [88]	1998
	Edge detection	Harris et al. [56]	1996
	Segmentation	Poli [107, 108]	1996
Other	Low level feature extraction	Trujillo et al. [148]	2006
	Autonomous robot vision	Martin [95, 93, 94]	2006, 2002, 2001
	Orientation detection	Roberts et al. [120]	2000
	Sparse optical flow	Ebner et al. [39]	1999
	Moravec operator	Ebner et al. [37]	1998

Table 2.2: Summary of genetic programming literature in the vision and image domain.

2.5.2.1 GP applied to object detection problems

This section is a survey of literature using genetic programming for the purpose of locating small objects in complex images. Zhang et al. [165, 168] described a domain independent

approach using genetic programming to detect the location of multiple class objects. Zhang et al. define domain independent as being able to apply the same approach to any problem and the approach will work unchanged. The approach utilises features that are composed of: (a) raw pixels directly or (b) easily computed pixel statics such as the mean and variance of pixels within generic shapes. These features are not specific to any type of problem and are referred to as domain independent features.

The work was tested on a suite of object detection problems ranging from easy, synthetic images, medium, heads and tails of Australian coins, to a more difficult problem of detecting haemorrhages and micro-aneurisms in retinal images. Zhang et al. [166] found that using genetic programming as a method greatly reduced the number of false alarms in images compared to a neural network. The detection rate using the genetic programming method was also superior.

Howard et al. [59] used genetic programming to evolve a detector that can automatically detect ships in synthetic aperture radar SAR images. Results obtained by Howard et al. [59] compared favourably when benchmarked with previous work on the same problem using a self-organising Kohonen neural network and a multi-layered perceptron neural network. Benson [8, 9] used finite state machines with embedded genetic programs FSM(GP) to perform the task of automatic target detection and applied to the same problem as Howard et al. [59]. A comparison of test results using figure of merits (FOM) as the measure was demonstrated favourable for the FSM(GP) when compared with Howard et al. [59].

Howard et al. [58, 60] modifies the multi-stage method using genetic programming from [59] to detect vehicles in infrared line scan (IRLS) images. A comparison of detection performance using simple and textural statistics in the second stage detectors concluded that that the textural statistics were marginally superior to simple statistics.

Tackett [141] applied genetic programming for classifying targets/non-targets in IR imagery. Two experiments using different terminal sets were performed using GP for training a classifier. The genetic programming method was compared to a neural network and binary tree classifier and in both cases the genetic programming method produced fewer false alarms.

Roberts et al. [119] coevolved both feature extraction and object detection using genetic programming to detect simple objects in artificial and natural images. Although the approach

was only applied to detect relatively simple objects, Roberts et al. stated that the algorithm was able to automatically choose features thought to be appropriate rather than requiring a human designer to manually chose features.

2.5.2.2 GP applied to image classification problems

This section is a survey of literature using genetic programming for the purpose of classification. Agnelli et al. [2] applied genetic programming for classifying documents into one of two classes, i.e. text segments and picture segments. The evolved classifier was applied to a large data set and good accuracy was achieved. Agnelli et al. on p. 308 noted that a reason for using GP was because the symbolic nature make the solutions easier to understand compared to “neural networks and most classifiers”. This may allow an expert to gain insight in the domain. Song et al. [136] presented two classification methods using genetic programming for classifying Brodatz textures. The results demonstrated that genetic programming was able to classify textures, and showed that the dynamic range selection method of classification not only had a higher accuracy but also converged at a faster rate compared to the static range selection.

Preliminary work presented by De Falco [33] applied GP to the classification of spectral pattern recognition. Results were reported as “positive and encouraging”. Ross et al. [123] used genetic programming for classifying minerals from hyperspectral images. This work is closely related to the work by Rauss et al. [114] that evolved classifiers to detect grass in spectral images. The work illustrated that a classifier could be evolved for accurately detecting the existence of a particular mineral. Rauss et al. [114] used genetic programming as a tool for classifying spectral imagery. More specifically the aim was to classify grass from the spectral images. It was reported that classification was not ideal when presented with new examples during testing. Daida et al. [31] used genetic programming to extract ridge and rubble features in multi-year ice from earth resonance satellite (ERS) synthetic aperature radar (SAR) data. The author reported that the results are excellent and compare favourably with a manually interpreted ERS SAR data product.

Gray et al. [53] compared genetic programming to a neural network for classifying tumors from a nuclear magnetic resonance spectra of biopsy extracts. A comparison of classification

accuracy for the genetic programming and neural network approach showed a classification accuracy of 80% and 90% respectively. A finding by Gray et al. was that the evolutionary process found simple programs that were as competitive at classifying as more complicated solutions.

Riolo et al. [116] proposed genetic programming for the purpose of classifying satellite remote sensing imagery. The goal is to predict whether a pixel represents water or not, based on the information from the spectral bands. The results, although preliminary, showed that the method was able to discover simple relationships that could correctly predict >98% for training and testing data.

2.5.2.3 GP applied to image processing problems

This section is an outline of literature relating to the use of genetic programming for image processing. This area of research is less relevant to our problem, i.e. classification and localisation of landmarks, and therefore only a brief outline will be given to describe where genetic programming has been used to evolve programs for image processing techniques. These include edge detection, segmentation and thresholding.

Lucier et al. [88] used genetic programming to evolve an edge detection program, however, the test set was limited to three ‘toy’ images of varying difficulty. However, the Ross et al. [122] work was applied to a real world problem of detecting grain edges in petrographic images. The results demonstrated that an evolved edge detector was able to perform better at locating fine grain edges than an edge extraction procedure that required ten steps to extract the edges. As a comparison to the GP approach, a neural network was learnt using the same training data. The results from the neural network approach were considered inferior, with the authors concluding that a more careful approach to the selection of data needed to be undertaken. Harris et al. [56] used GP to replicate Canny’s Gaussian first-derivative approximation, however, the work was only applied to a one dimensional signal.

Lin et al. [84] used genetic programming to find composite operators to extract regions of interest from an image that could also be applied to similar images. Poli [107, 108] used GP, and a combination of simple terminals, functions and fitness functions, to evolve a program to segment regions of interest in medical images. The GP approach was able to outperform

neural networks. Poli suggested that GP could be used as an approach to create image analysis tools much more powerful than those currently used in image processing. Rosin [121] used genetic programming to detect the presence of landslides in multi-temporal aerial images. A multi-temporal analysis deals with the detection of changes in pairs of images acquired in the same geographical area at different times [18].

2.5.2.4 Terminal set

In the context of using genetic programming in object detection problems, terminals generally correspond to image features. An important component of genetic programming is determining what are useful features for solving a problem. However, this is one of the difficulties of solving real world problems when domain knowledge is not available. If the feature set is not sufficient to express a solution, then GP is unable to solve the problem. However, containing too many extraneous or redundant features in the terminal set causes the efficiency of GP to decrease caused by futile searching in the higher-dimensional search space. Koza in [74] demonstrated on a symbolic regression problem that extraneous features reduce the probability of success. Ok et al. in [101] proposed an adaptive mutation based on terminal weighting for finding relevant features from a terminal set. However, although the results were promising, the method was only applied to a symbolic regression problem and not extended to real world problems. Therefore choosing relevant features is a careful selection process that seeks to minimise extraneous features.

Although some work uses image pixels directly, most work in GP applied to object detection problems have used a terminal set composed of features calculated using *simple statistics* applied to pixel values [8, 58, 59, 60, 141, 164, 165, 166, 168]. Generally speaking, the input window was divided into various shapes similar to those shown in Figure 2.8 and the mean and standard deviation of pixel values calculated for each shape. However, Zhang et al. [165], Howard et al. [58] and Tackett [141] used features described as rotational invariant statistics, textural statistics and moment and intensity based features respectively. In each of these cases it was demonstrated that simple statistics had superior performance by producing fewer false alarms. Tackett hypothesised that the non-linear nature of GP combined with simple statistics may be discovering features better suited to the problem than human-synthesised

features. Another advantage of using simple statistics is the speed of processing because of the low computational overhead.

In addition to image features, Zhang et al. [164, 166, 168] also used a terminal which generates a random number in the range $[0, 255]$; the range corresponding to the number of grey levels in the image.

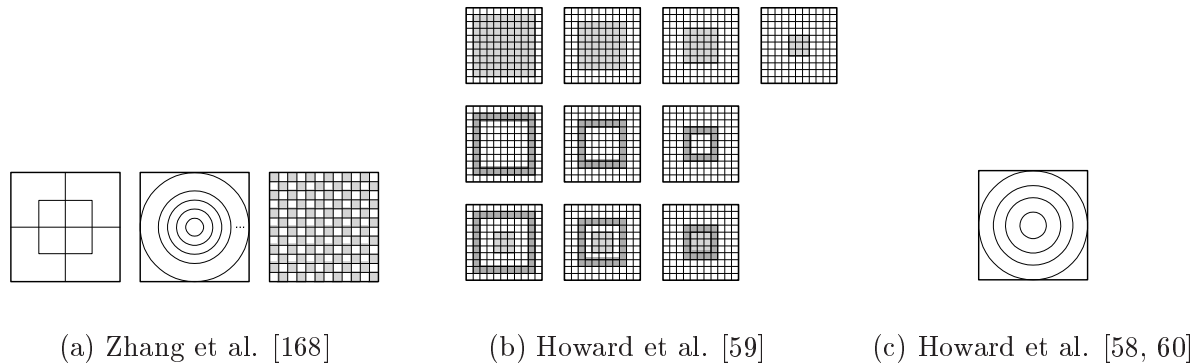


Figure 2.8: Shapes used for calculating features.

2.5.2.5 Function set

One of the problems with genetic programming is choosing a set of functions that is able to solve the problem. Work performed by Koza [74] indicated that genetic programming is unable to solve the problem if the function does not contain the necessary operators, however if the function set includes irrelevant operators then the performance will be degraded. Work by Wang et al. [152] supports this hypothesis based on the results from three different problems. They found that if too many operators are included in the function set then this may degrade performance. This is because the extra operators lead to an increase in search space.

Wang [153] experimented with various function sets for solving a sequence induction problem and two symbolic regression problems. The resulting experimentation on the problems found two common characteristics with the best performing function set. The investigation revealed that the function set contained operators similar to the target function and the function set contained the smallest number of operators. Although the first point is an interesting observation, the difficulty with many real world problems is that we do not know

what operators are required prior to solving the problem.

Soule et al. [138] contradicts the findings by Wang [153] that the best function set contains operators found in the target function. The aim of Soule et al. work was to evolve a function for the problem in Equation 2.8. They found that a smaller function set $\{+, -, \times, /, \sqrt{\quad}\}$ was able to outperform a function set containing the exact operators $\{+, -, \times, /, \sqrt{\quad}, \tan\}$ required to replicate Equation 2.8, although this conclusion is based on the average fitness and not the optimal solution. They concluded by saying “that better information regarding how to choose function sets could significantly improved GP performance” [p. 190]. So as an initial investigation to our object detection problem we are guided by the operators that have been used in object detection and image classification problems.

$$\sin(x) = \frac{\tan(x)}{\sqrt{1 + \tan^2(x)}} \quad (2.8)$$

One thing common in the literature is that the function set should use the smallest or minimalistic approach to choosing operators for solving the problem. This places an emphasis for choosing the correct operators for solving the problem in order to minimise the search space. Zhang [164] also demonstrated that using additional operators in the function set does not improve detection rate and may also reduce the rate of convergence when training programs to detect objects in difficult images.

Table 2.3 is a summary of operators that are used for object detection and classification problems in image-related applications. The most common set of functions used by genetic programming for vision and image-related problems are the $+$, $-$, \times and $/$ operators. Other operators that appear frequently are boolean and (min and max) operators, and less cited operators include trigonometric operators (sine, cosine, tangent) and exponential operators. It is hard to see how periodic functions such as sine and cosine assist with creating a better classifier since data in vision and image-related applications are generally not periodic. The only application where a periodic function may be useful is evolving a program for removing periodic noise.

Domain	Source	Application	+, -, ×, /	Boolean	min, max	Other
Detection	Zhang et al. [165]	Object detection	✓	✓		
	Zhang et al. [168] (a)	Object detection	✓			
	(b)	Object detection	✓			✓
	Zhang et al. [166]	Object detection	✓			
	Benson [8]	Det'n of ships in SAR images	✓		✓	✓
	Howard et al. [59]	Det'n of ships in SAR images	✓		✓	
	Howard et al. [58, 60]	Det'n of vehicles in IRLS images	✓		✓	
Tackett [141]	Target detection	✓	✓			
Image classification	Agnelli et al. [2]	Texture/picture classification	✓	✓		✓
	Teredesai et al. [144]	Digit classification	✓			✓
	Song et al. [136, 135]	Texture classification	✓	✓		
	Ross et al. [123]	Mineral classification	✓		✓	
	Rauss et al. [114]	Class'n of spectral imagery	✓			
	Winkeler et al. [157]	Facial recognition	✓			
	Daida et al. [31, 32]	Classification of SAR imagery	✓			
	Gray et al. [53]	Classification of brain tumors	✓	✓		✓
	Riolo et al. [116]	Classification of RS imagery	✓	✓		
	Teller et al. [142]	Object classification	✓		✓	

Table 2.3: Summary of operators made available as part of the evolutionary process for vision and image applications.

2.5.2.6 Performance measures using genetic programming

In determining a performance measure for solving problems in genetic programming, as with any machine learning algorithm, it is important to ensure the goals of the problem have been captured. A general overview was given on how to measure the performance of object detection systems in Section 2.2.1. As mentioned previously, the performance of a program in genetic programming is measured according to a fitness criteria and the population of programs are ranked according to the measured fitness. As part of the evolutionary process the fittest programs are reproduced and copied so that they can partake in the next generation.

In the context of object detection in image-related applications, fitness is measured using a combination of true positives and false positives, i.e. the number of objects correctly and incorrectly detected respectively. Several variations of fitness functions used to measure the performance of object detection systems are shown in Table 2.4. Each of these fitness measures can be described as aggregating functions because the objectives are combined to produce a single function. The advantages and disadvantages of aggregating functions are described in Section 2.2.1.1.

Source	Application	Fitness function
Zhang et al. [165]	Object detection	$\alpha FR + \beta(1 - DR) + \delta FAA$
Zhang et al. [166, 168]	Object detection	$\alpha FR + \beta(1 - DR)$
Benson et al. [8, 9]	Det'n of ships in SAR images	$\frac{\alpha TP}{FP + N_t} + \frac{\beta TN}{FN + N_o}$
Howard [59]	Det'n of ships in SAR images	$\frac{\sum_{\text{hits}} (5 - \text{target grade})}{\sum_{\text{targets}} (5 - \text{target grade} + FP)} - 1$
Howard [58, 60]	Det'n of vehicles in IRLS images	$\frac{\alpha TP}{n_s + \beta FP}$

Table 2.4: Fitness functions used for measuring the performance of programs for object detection problems.

Computer vision problems using genetic programming for classification

In Section 2.2 we described object detection as the task of classification and localisation. This section gives a summary of literature relating to classifiers in genetic programming for computer vision related problems, such as object detection and image processing. The most common approach to classification in genetic programming is to use a representation similar to Figure 2.4 on page 23. In this case, the output of the evolved program is a real number that needs to be mapped to a decision. In the context of classification or object detection, the real number is converted into a decision about the class or object respectively.

The majority of genetic programming literature relating to image classification uses a boundary as a decision point that is used to differentiate between two classes, i.e. object/background, target/clutter, edge/non-edge, picture/text, etc. The classification of problems into two classes using GP is relatively simple where the most common approach is to use zero as the decision point between the two classes [2, 88, 107, 122, 141]. This is illustrated in Figure 2.9. For example, when a program is applied to an instance of data, the output is computed returning a value and the class is chosen depending on whether the value is negative or positive. An exception to this is work by Teredesai et al. [144] that defines an interval between the two classes as uncertain (refer to Figure 2.9). However, the problem with this approach is the output returned by the program can be orders of magnitude greater than the defined interval and so the decision between class boundary can still be black and white. Song et al. [136] compared dynamic range with static range selection to classify textures

and found dynamic range selection to have a higher detection accuracy and solutions tend to converge in less generations.

When a problem has more than two classes then multiple binary classifiers have been used in parallel and a heuristic is used to distinguish between the different classes [114, 123, 142, 144]. An exception to this is work by Zhang et al. [164, 165, 166] who used the output returned from the program as per the binary classifier, however, in this instance the object classes were divided into a discrete number of steps each representing a separate class (refer to Figure 2.9). A foreseeable issue with the Zhang et al. approach is that a different set of features may be required for locating objects in separate classes. Whilst it is possible to combine many features using genetic programming, using this approach may make the search unnecessarily large when evolving a detection program to locate many different types of objects.

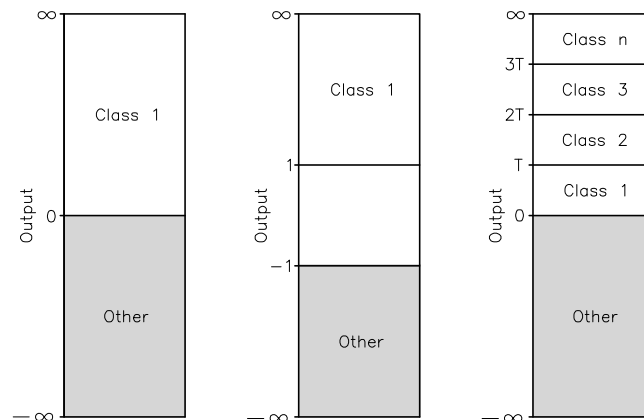


Figure 2.9: The mapping of binary and multiple classes to a decision using three concepts of class decision boundaries. Zero boundary (left), dealing with indecision (middle) and multiple class classifier (right).

2.5.3 Parameters

Table 2.5 is a summary of genetic programming run-time parameter values by [2, 8, 33, 58, 114, 123, 136, 135, 144, 157, 165, 166, 168] which have been used by genetic programming to solve image-related applications. The run-time parameters are used by genetic programming during training.

Population size, M , is the number of individuals in the population, maximum generations,

G , is one of the termination criteria to end the evolutionary process, maximum depth, D , is the maximum tree depth allowed for programs and the initial maximum tree depth, d , is the maximum tree depth allowed in the initial generation. Reproduction, P_R , crossover, P_C , and mutation, P_M , are the probabilities of selecting a particular genetic operation to create new individual(s). For a definition of the genetic operations refer to Section 2.5.1.2.

Parameters	Range
Population Size, M	100-5000
Maximum generation, G	100-2000
Maximum depth, D	8-20
Initial maximum depth, d	4-6
Probability of:	
Reproduction, P_R	0-0.10
Crossover, P_C	0.65-0.95
Mutation, P_M	0-0.25

Table 2.5: A range of genetic programming run-time parameter values that have been applied to image-related applications.

2.5.3.1 Termination criteria

An evolutionary run requires a set of criteria for deciding when the evolutionary process should be terminated. The termination criteria consist of either satisfying success criteria or the evolutionary process reaching a predefined number of generations at which point the evolutionary run will be terminated. The success of a program can be easily measured and compared against the goals of the problem, however, choosing how many generations that constitutes an evolutionary run is more complicated. Deciding when to terminate an evolutionary run is difficult because if a run is stopped too early the practitioner may not have taken best advantage of the evolutionary search. However, if the evolutionary run is allowed to continue, the search may have prematurely converged and become stuck at a sub-optimal solution resulting in diminishing returns if the evolutionary process is allowed to continue. Therefore if the goal is to achieve a solution of a certain standard then it may be more efficient to terminate and start another run.

A method popularised by Koza [75] for determining when a run should be terminated uses the *cumulative probability of success*. However, this assumes either the discovery of an optimal solution or defining what constitutes a successful solution. Luke [90] presented a

method for comparing the quality of solutions from a long single run with multiple shorter runs. The method was applied to a problem in three domains and Luke concluded that it makes more sense to do multiple shorter runs, m runs of n/m generations, than one long run of m generations. There are other criteria for determining termination criteria such as measure of diversity or the fitness reaching a plateau [55].

2.5.4 Some issues with using genetic programming

Some of the issues relating to customising genetic programming to solve problems in the image-related domain have been discussed above. The following are some other issues that are known in the genetic programming domain:

- *Over fitting* is an occurrence where the performance on the training examples still increases while the performance on unseen data becomes worse. A way to avoid overfitting is to use a large training data set, however, it is not uncommon for the dataset to contain a finite number of samples. As a way of visually assessing whether over fitting has occurred, Langdon et al. in [80] plotted training versus test performance using the best individual from each evolutionary run. If the points are scattered about the diagonal line then little over fitting has occurred. An example of a training versus test performance graph is shown in Figure 5.7.
- *Premature convergence* occurs when the population converges to a suboptimal solution. Banzhaf et al. in [7] suggest that improving the diversity of programs within a population is the key to reducing premature convergence.
- *Problem representation* is an issue for difficult real world problems where there is no *a priori* knowledge of the types of features and terminals required to solve the problem. Selecting a suitable combination of features and terminals is akin to possessing the correct ingredients to bake a cake. We stated in Sections 2.5.2.4 and 2.5.2.5 that if the terminal or feature sets are not sufficient then GP is unable to solve the problem, however, if the sets contain too many extraneous terminals/features then the performance of GP will decrease.

- *Scaling problem difficulty* is an issue in genetic programming because as the size and complexity increases so does the size of the search space and the time taken to find an optimal solution. Luke in [89] states that this makes the solution vulnerable to bloat. Gustafson et al. in [55] discussed the relationship between problem difficulty and code growth and stated that the increased rate of code growth is induced by the higher selection pressure and less genetic diversity.
- *Bloat*, or code growth, is a term given to describe the process of code growing over time [35]. Code growth in genetic programming is caused by the variable length representation. Soule et al. in [137] stated that most code growth consists of code that does not contribute to a program's performance. Code segments not contributing to a program's performance are commonly referred to as introns. Research on the usefulness of bloat is mixed, however, there is a large amount of literature dealing with the issue of reducing bloat. Reasons given why bloat should be prevented are: larger programs may be indicative of over fitting [81, 141]; bloat forces the evolutionary process into stagnation [5]; and the processing of extraneous code adds to the computation time [35]. The two most common approaches to reduce bloat are: limiting the tree depth of a program; or incorporating program size as part of the fitness measure (this is commonly referred as parsimony pressure).
- *Program understandability* is an issue in genetic programming because of the difficulty of interpreting the function of evolved programs. Prior to implementing the program in a real system, it is necessary for engineers and the like to have understanding of the functioning of the system. The following section is a summary of previous research performed to improve the understandability of programs generated by the genetic programming paradigm.

2.5.4.1 Improving program comprehensibility

Most programs discovered using genetic programming are treated as a black box, i.e. GP is run and the learnt program, or 'best individual', is blindly applied to unseen data without any understanding of the learnt program. Although genetic programming works relatively well on

many types of problems the generated programs are often unintelligible. Generally speaking, genetic programming literature is very thin when it comes to providing any explanation about the learnt programs.

Ideally, the goal is to discover a set of programs that is able to solve the problem whilst at the same time being comprehensible to the user. Previous work using genetic programming has been applied to discover comprehensible programs in real world problems [15, 66, 100]. Ngan et al. in [100] were able to discover additional knowledge, beyond some general knowledge already known, for two real world problems. Song et al. [134, p. 2099] determined that the generated texture classifiers for binary textures were not ad-hoc, and in fact behaved “as template matchers and frequency analysers”. However, when genetic programming was trained to classify complex greyscale textures the generated texture classifiers were more difficult to understand. An advantage of discovering knowledge in the learnt program is to provide insight and allow a better understanding of how the problem is solved.

Bojarczuk et al. in [15] state that most genetic programming literature associates comprehensibility with small programs - the likely reason is when a program increases beyond a certain size the comprehensibility decreases. As a result, research for discovering comprehensible programs for difficult real problems has been limited because of their association with problem complexity that is related to program size. Johnson et al. in [66] associates the increase in program complexity, as a result of bloat, with the reduction in interpretability. We previously stated that one of the common approaches to combat bloat is to apply parsimony pressure, which has subsequently been applied to increase program understandability [34, 57, 66, 78, 134]. Lai in [78] demonstrated that using parsimony pressure makes the programs easier to understand compared to using no parsimony pressure.

2.6 Pulse Coupled Neural Networks

Section 2.5.2.4 contained a discussion about the advantages of dividing an input window into various shapes and then computing simple statistics to be used as features compared to using complex features, e.g. Wavelets. However, deciding how to divide the input window into a set of shapes that will give GP the best opportunity to evolve a program is difficult. We

believe that extracting regions of interest in noisy and low contrast images will improve the detectability of landmarks located within cluttered images.

2.6.1 The PCNN model

A technique that has shown potential to extract regions of interest is the Pulse Coupled Neural Network (PCNN). The PCNN model in [68] is a modification of the Eckhorn linking field network [40]. The Eckhorn model was biologically inspired from a cat's visual cortex and modified to be used as an image processing algorithm [67]. In essence, the PCNN model is a digital simulation of the cat's visual cortex. The PCNN model generates a sequence of binary images that contain segments and edges by iterating Equations 2.9-2.13. Waldemark *et al.* in [150, p. 241] describe the output from the PCNN model as a series of binary images, where "each binary image contains different sets of segments from the original image".

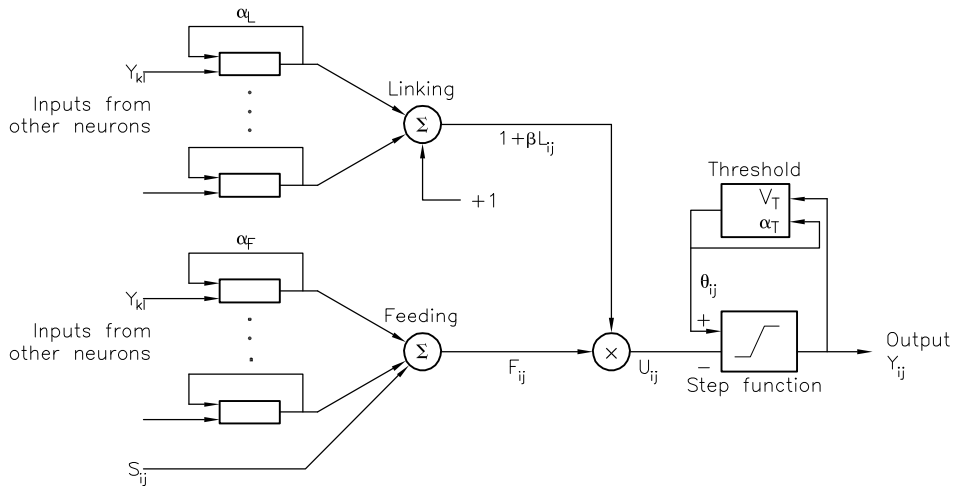


Figure 2.10: A block diagram of the PCNN.

The block diagram of the PCNN model shown in Figure 2.10 contains two main compartments: the feeding, $F_{ij}[n]$, and linking, $L_{ij}[n]$, compartments. Each of these compartments communicates with neighbouring neurons through synaptic weights of M and W respectively. Each compartment retains its previous state but with a decay factor and only the feeding compartment receives an input stimulus, S_{ij} , i.e. the pixel intensity at location (i, j) . The values for the feeding and linking compartments are computed using Equations 2.9 and 2.10 respectively. $Y_{kl}[n-1]$ are the outputs of neurons from a previous iteration. The constants

α_F , α_L and α_T are decay terms for feeding, linking and threshold respectively, and V_F , V_L and V_T are magnitude scaling terms for feeding, linking and threshold respectively.

$$F_{ij}[n] = e^{\alpha_F \delta_n} F_{ij}[n-1] + S_{ij} + V_F \sum_{kl} M_{ijkl} Y_{kl}[n-1] \quad (2.9)$$

$$L_{ij}[n] = e^{\alpha_L \delta_n} L_{ij}[n-1] + V_L \sum_{kl} W_{ijkl} Y_{kl}[n-1] \quad (2.10)$$

$$U_{ij}[n] = F_{ij}[n] \{1 + \beta L_{ij}[n]\} \quad (2.11)$$

$$Y_{ij}[n] = \begin{cases} 1 & \text{if } U_{ij}[n] > \Theta_{ij}[n-1] \\ 0 & \text{Otherwise} \end{cases} \quad (2.12)$$

$$\Theta_{ij}[n] = e^{\alpha_T \delta_n} \Theta_{ij}[n-1] + V_T Y_{ij}[n] \quad (2.13)$$

The two compartments are then combined to create an internal state of the neuron, $U_{ij}[n]$, which is controlled by the linking strength, β – refer to Equation 2.11. At this point, the internal state of the neuron is then compared to a dynamic threshold, $\Theta_{ij}[n]$, to produce an output, $Y_{ij}[n]$ – refer to Equation 2.13. The output is either 0 or 1 which produces a binary output at location (i, j) at iteration n . The threshold is dynamic by the fact that when the neuron fires, the threshold increases its value and then decays until the neuron fires again – refer to Equation 2.13. The series of equations are iteratively computed from Equation 2.9 through to Equation 2.13 which is stopped at the discretion of the user.

2.6.2 PCNNs applied to image-related applications

Studies into the feasibility of the PCNN applied to image-related applications have been diverse, covering a range of domains that includes military and medical imagery. The PCNN has been applied to images requiring smoothing, segmenting and feature extraction. However, in the context of our problem we will only be discussing PCNN literature relevant to segmenting regions of interest.

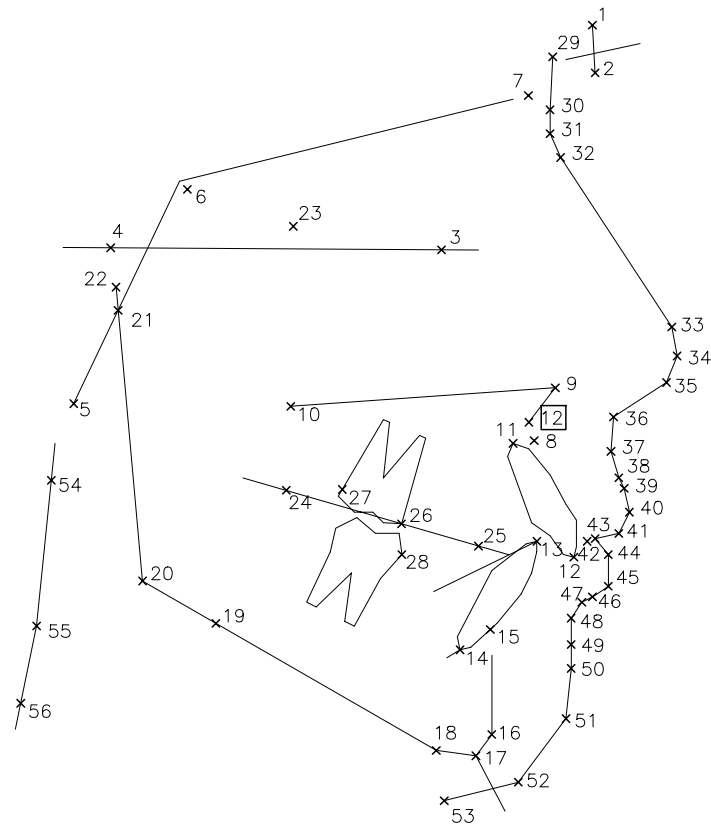
Preliminary research of PCNNs in medical imaging has produced promising results when locating regions of interest in the following areas: segmentation of brain structure and abdomen structure in MRIs [72]; segmentation of lungs in pulmonary scintigraphic images [72];

identification of the left ventricular endocardial border in echocardiographic images [159]; segmentation in mammogram imagery [67, 86] and isolating arteries from veins in retinal images [67]. The above examples demonstrated that the PCNN was able to segment regions of interest in real-world medical images. With the exception of the aforementioned medical examples, the PCNN also produced promising results when segmenting regions of interest in images having poor signal-to-noise ratios. For example, the PCNN was used for defining region boundaries and discriminating specific regions of interest in spectral images [29] and Ranganath et al. in [76, 112] reported perfect segmentation of a simple problem even when the intensity ranges overlap.

However, an issue with the PCNN is that the general consensus by [21, 67, 72, 76, 139, 151, 150] is that for automatic segmentation to be viable for a large range of images, further research into a better understanding of the set parameters and the parameter relationships is required. Waldemark et al. in [151] suggested that feedback could allow the parameters to be dynamically altered giving a higher attention to regions of interest. Preliminary research by Ranganath et al. [113] uses a semi-intelligent method to adjust the parameters. Firstly a PCNN is used to suppress noise followed by a second PCNN to segment the image. During each iteration of the segmentation stage, a control module deletes regions from further processing that do not meet certain criteria based on prior knowledge of the region.

2.7 Automatic cephalometric landmarking

Cephalometric landmarks are a set of craniofacial points of interest that are used by an orthodontist to determine the physical normality of the patient; this is a simple definition describing a cephalometric analysis. If the patient deviates from the pre-defined norm, the orthodontist is able to determine a treatment plan based on the linear and angular relationships of the landmarks to correct the abnormality. The type of cephalometric landmarks required are dictated by the type of cephalometric analysis that is performed. Figure 2.11 is an illustration of 52 landmarks and their spatial relationships. The process of locating the positions of these landmarks is time consuming and mundane and is a process orthodontists would like to automate.



- | | | |
|---------------------------|---|-------------------------|
| 1. Scale #1 | 20. Gonion | 39. Upper Lip Sup |
| 2. Scale #2 | 21. Articulare | 40. Upper Lip Mid |
| 3. Orbitale | 22. Condylion | 41. Upper Lip Inf |
| 4. Porion | 23. PterygoMaxillary Fissure Apex (PTV) | 42. Upper Lip Height |
| 5. Basion | 24. Post Occlusal Plane | 43. Lower Lip Height |
| 6. Sella | 25. Ant Occlusal Plane | 44. Lower Lip Sup |
| 7. Nasion | 26. Mesial Upper 6 | 45. Lower Lip Mid |
| 8. A Point | 27. Distal Upper 6 | 46. Lower Lip Inf |
| 9. Ant Nasal Spine | 28. Mesial Lower 6 | 47. Mental Fold Sup |
| 10. Post Nasal Spine | 29. Forehead | 48. Mental Fold Mid |
| 11. Apex Upper Incisor | 30. Nasion Sup | 49. Mental Fold Inf |
| 12. Incisal Upper Incisor | 31. Nasion Mid | 50. Chin Sup |
| 13. Incisal Lower Incisor | 32. Nasion Inf | 51. Chin Mid |
| 14. Apex Lower Incisor | 33. Nose Sup | 52. Chin Inf |
| 15. B Point | 34. Nose Mid | 53. Soft Menton |
| 16. Pogonion | 35. Nose Inf | 54. Odontoid Sup |
| 17. Gnathion | 36. Philtrum Sup | 55. Odontoid Inf |
| 18. Menton | 37. Philtrum Mid | 56. Cervical Vertebra 4 |
| 19. Md Plane Tangent | 38. Philtrum Inf | |

Figure 2.11: Definition of cephalometric landmarks

2.7.1 Previous work in cephalometric landmarking

Traditionally a cephalometric analysis was performed by manually tracing points on a lateral cephalometric film X-ray. However, more recently a semi-automated approach has been developed which allows an orthodontist to mark the positions of landmarks on a digitised film using a computer system and mouse. After the relevant landmarks have been entered the computer system performs the cephalometric analysis. As a natural progression to the semi-automated approach and whilst not a new idea, an automated approach to locating cephalometric landmarks was first proposed by Hussain et al. [64] in 1985.

To date, no fewer than twenty independent researchers have attempted to automatically locate cephalometric landmarks in one form or another. Research into automating the cephalometric analysis can be categorised into the following two sections: *traditional computer vision* and *machine learning*. Prior to 1990, the focus of research was to locate landmarks using image processing techniques in conjunction with handcrafting anatomical knowledge (non-machine learning) [83, 104, 145, 146]. However, post-1990 research has focused on using machine learning algorithms for locating landmarks [19, 20, 24, 25, 63, 87]. Although the more recent research has produced promising results, landmarking thus far has been unsuccessful for reasons including: poor detection accuracy, a lack of algorithmic robustness and small test sets.

2.7.1.1 Traditional computer vision

The following historical background provides an outline of the significant research, in terms of results, for automatically detecting landmarks with a critical discussion of deficiencies. The following discussion is based on literature that uses a non-machine learning methodology or handcrafting for landmark detection.

Levy-Mandel et al. [83] proposed a knowledge-based line-following method that accounts for changes in biological shapes. The *a priori* knowledge for each landmark was encoded into algorithms. The lines were extracted using a set of image processing operations. The test images used were from a very stringent selection of X-rays digitised to 256×256 pixels and 256 grey-levels, i.e. head carefully positioned, no filled cavities or missing teeth. The results

claim a success of 23 from 36 landmarks, however, the size of the test set was not given. It was also claimed that 13 of the landmarks were not found because the lines were not present on the digitised X-ray.

Parthasarathy et al. [104] proposed to automatically locate nine landmarks based on *a priori* knowledge of human facial structure. The original cephalogram was first digitised to 480×512 pixels and a four-level resolution pyramid was created to improve the efficiency of the search. The resolution pyramid works on the lowest resolution for locating features of interest and moves to higher resolutions to refine the search. The algorithm subsequently applies a series of digital image processing and feature recognition techniques to enhance the images. At this point, the landmark is located using a set of handcrafted rules based on *a priori* to track the facial structure's profile which is relevant to the detection of the landmark. The results for nine landmarks and a test set of five images, on average 58% were located within an error of 2 mm.

Tong et al. [145, 146] presented an extension to Parthasarathy et al. [104] by locating an additional seventeen landmarks. The seventeen landmarks are located in both bony structure (nine landmarks) and soft tissue (eight landmarks). The initial pre-processing steps are per Parthasarathy et al., i.e. a resolution pyramid is created from a digitised image and further filtering techniques are applied to trace the bony structure of the jaw. The soft tissue profile is found by applying filtering techniques with *a priori* knowledge of the skull anatomy. The algorithm uses the soft tissue profile and features inside the skull to determine new landmarks as well as previous landmarks to compute regions of interest for further region enhancement. The test set consisted of the five cephalograms as per Parthasarathy et al. and digitised to 512×464 pixels and 256 grey-levels. The results for the seventeen points, on average 76% were located within an error of 2 mm. The method had trouble locating the Porion and Gonion landmarks in each case. The focus of this work seemed to centre around locating landmarks with accuracy being a minor objective.

2.7.1.2 Machine learning

The following provides an outline of significant research, in terms of results, for automatically detecting landmarks and critically discussing deficiencies where applicable. The following discussion is based on literature that uses a machine learning methodology for landmark detection.

Cardillo et al. [20] presented an algorithm based on sub-image matching to locate twenty landmarks using a dataset of 40 images (512×490 and 256 grey-levels). The algorithm is based on greyscale mathematical morphology with a statistical approach to learn the structuring elements and their origins' probability distributions. A learning approach was used to overcome subtle changes in the facial structure. Cardillo et al. on p. 287 states that the landmark's detection performance "steadily increases from 60% at the start of training to a level of approximately 85% after 40 X-rays". Based on the test results the work seems very promising, however, it is unclear whether the test data used to determine the test results is independent of the training data. This is based on the number of images at the completion of training which seem to exhaust the entire dataset.

Chen et al. [24, 25] combined a multilayer perceptron and genetic algorithm (GA) to extract specific feature areas containing the landmark. The multilayer perceptron was used as an approximation to the genetic algorithm's fitness function. However, no results were reported stating the performance of the system.

Hutton et al. [63] proposed active shape models (ASMs) to locate sixteen cephalometric landmarks. The ASM uses a template of the spatial relationships between the important structures to help search the image for features of interest. The model was established from a training set of hand-annotated images that was subsequently applied to unseen test data. Sixty-three randomly selected cephalograms were tested using the leave-one-out method. The results for sixteen landmarks and a test set of sixty-three images, on average gave 35% accuracy within an error of 2 mm. Hutton et al. concluded that the current implementation of the method did not give sufficient accuracy for a cephalometric analysis but suggested that

improvements are possible.

El-Feghi et al. [44] used a multi-layered perceptron (MLP) for automatically locating cephalometric landmarks. This work has been the most in-depth investigation for the following reasons: the size of the dataset was 134 images and the system was applied to twenty types of landmarks. This approach differs significantly from previous work because the detection system is based on the initial positions of four points that are located using a simple heuristic. The four points are then used to compute additional features to form a feature vector that represent size, rotation and the offset of the skull. The features are then used as inputs into the MLP and the corresponding outputs represent the co-ordinates of the landmark. The average detection results, when compared to Cardillo et al. [20], were in most cases superior, however, no statistical test was used to confirm whether the differences were statistically significant. Rather than using a MLP, El-Feghi in [43, 45] used an alternative approach in the form of a fuzzy neural network and partial least squares regression in [42].

However, the test set is not completely independent from the training set and so the results are optimistic. This claim is based on how El-Feghi et al. selected their training and test sets and their method of biasing the data. As a method of reducing the number of training samples, a K-means clustering algorithm was used to form 55 clusters from the 189 samples. Clustering is based on a similarity measure between the computed feature vectors. A sample from each cluster was then used for training with the remaining samples used for testing. As a result, there will be a good chance that there will be a high similarity between the computed feature vectors in both the training and the remaining test samples.

Chakrabartty et al. [22] demonstrated the performance of project principal-edge distribution features with a support vector machine classifier to automatically locate eight types of cephalometric landmarks. The project principal-edge distribution features attempts to capture information of an image by modelling its edge distribution along different principal directions or orientations. Although the results are based on a small selection of landmarks the detection performance appears to be promising by demonstrating an accuracy of more than 95%.

Giordano et al. [52] used cellular neural networks to locate eight types of landmarks and claimed that the approach is versatile enough to be used for the detection of landmarks that are located on both edges and regions (e.g. the sella landmark), however, this claim appears to be based on a small number of, and pre-selected, test cases.

Yue et al. [161, 162] used classical image processing techniques and a pattern matching algorithm to locate an initial set of twelve reference landmarks. These landmarks are then used to divide the craniofacial shape into ten independent regions according to anatomical knowledge. For each region, the principal component analysis is used to statistically characterise shape variations and the gray profile to derive a modified active shape model for localisation. This modified active shape model can be applied to test data to locate feature points, and with the assistance of a priori knowledge, the landmarks can be detected. The detection results indicate a significant improvement over the original active shape model approach that was proposed by Hutton et al. in [63]

Rueda et al. [127] used active appearance models, with pre-processing for homogenisation, to automatically locate 28 types of landmarks. The detection performance (<2 mm) for each of the landmarks appears to be significantly less than the results presented by Chakrabartty et al. in [22].

2.7.2 A critical review of automated cephalometric landmarking

In this section we will review some initial attempts to automate cephalometric landmarking. The results from the approaches that do not use machine learning are based on a relatively small number of, and in some cases pre-selected, test cases. An issue with these approaches is one can only assume the rules were encoded using all the available images and then the performance measured – nothing was stated otherwise to contradict this claim. This is akin to learning a set of rules using a machine learning algorithm and then testing on the training data. Another issue is that because the systems were validated using a very small selection of images from a large population, it would be fair to assume that if tested on a large dataset of images then the reported detection accuracies would be significantly reduced, i.e. the system

would not be robust enough to cater for the biological variability of the different landmarks.

Other approaches for developing classifiers for solving complex problems use machine learning algorithms. Although some of the detection accuracies for the machine learning algorithms are not as high as those obtained by the non-machine learning approaches, the results are based on a larger dataset of images. Generally speaking the machine learning approaches seem to perform better than the handcrafted techniques when factoring in the size of the dataset. A possible reason may be that a more complex set of rules can be discovered by a machine learning algorithm. It is conceivable that the complexity of the rule set is able to cater for the biological variability of the different landmarks. Such a rule set would be difficult for a human to discover. It is also important to mention that the results using the machine learning algorithms, in most cases, are based on test data that is independent of the training data. However, in some reports it is not clear whether the detection system has been tested completely independently of the training data [20, 44].

In the literature, several approaches have been proposed and Table 2.6 is a chronological summary of notable results.

Currently, a central database does not exist for a dataset of cephalometric images and as a result, researchers are developing their approaches to automatically detect cephalometric landmarks in isolation. It is not uncommon for researchers to compare their approach to previous work. This is quite unfair for several reasons that include: the sizes of the datasets are never the same, the resolution of the images are different; no statistical test comparing test results is performed; and the comparison is always performed on a different set of images. As a result, the comparison should only be used as an indication and not to make a potentially fallacious claim that one approach is better than another.

	Parthasarathy [104] 1989		Tong [145] 1990		Cheng [26] 1991	Cardillo [19, 20] 1994		Forsyth [50] 1996	Rudolph [126] 1998	Liu [87] 2000	Hutton [63] 2000
Test set size	5		5		10	40		10	14	10	63
Number of Landmarks	9		17		13	20		19	15	13	16
Landmark	DR	Error	DR	Error	Error	DR	Error	Error	Error	Error	Error
Orbitale			80	1.3±0.5	2.8	40	1.1±1.7	6.0	2.5±3.8	5.3±4.1	5.5±3.4
Porion			0	5.2±3.8	2.1	89	0.6±3.4	0.0	5.7±4.9	2.4±2.1	7.3±6.5
Basion								2.8			
Sella	100	1.4±0.4			1.1	53	1.4±1.5	0.6	5.1±3.4	0.9±0.5	5.5±6.8
Nasion	80	1.8±0.5			1.9	83	0.9±1.4	1.0	2.6±2.2	2.3±1.1	5.6±3.9
A Point			100	0.8±0.6	2.8	77	1.4±1.7	0.4	2.3±2.6	4.3±1.6	3.3±2.4
Anterior Nasal Spine	40	2.4±1.1			1.9	68	1.1±2.4	3.3	2.6±3.1	2.9±1.1	3.8±2.2
Posterior Nasal Spine	60	2.2±1.1				86	0.3±0.4	4.5			5.0±4.1
Apex Upper Incisor			60	1.7±0.8		79	1.4±1.7	2.1	2.2±3.0		2.9±2.6
Incisal Upper Incisor			80	1.1±0.8	0.5	76	2.4±3.8	0.4	2.0±2.0	2.4±2.0	2.9±3.8
Incisal Lower Incisor	60	2.1±1.3			4.9	64	2.1±2.3	0.7	2.5±2.5	2.9±1.0	3.1±2.3
Apex Lower Incisor	60	1.5±1.1				89	0.6±1.2	1.2	2.7±3.0		3.9±2.7
B Point	20	3.3±0.9			2.6	71	0.5±0.9	1.0	1.9±2.1	3.7±1.6	2.6±2.7
Pogonion	60	1.9±1.2			2.1	97	0.4±0.7	0.4	1.9±2.3	2.5±1.1	2.7±3.4
Gnathion					1.4	100	0.4±0.6	0.9		1.7±0.9	2.7±3.4
Menton	40	2.0±0.7			0.2	78	1.2±3.2	0.4	3.1±3.5	1.9±0.6	2.7±3.6
Gonion			20	2.7±1.0	0.9	61	1.2±3.5	0.6		4.5±3.1	5.8±6.0
Posterior Occlusal Plane						71	1.1±1.6				
Anterior Occlusal Plane						48	3.5±4.4				
Nose Mid						94	0.1±0.2				
Upper Lip Mid			100	0.5±0.3							
Lower Lip Mid			100	0.3±0.3							
Chin Mid			100	0.4±0.4		91	0.3±1.8				
<u>Additional Landmarks</u>											
X1			60	2.7±2.2							
X2			80	1.3±1.0							
X3			40	2.2±1.4							
X4			100	0.6±0.5							
X5			100	0.3±0.4							
X6			100	0.7±0.7							
X7			100	0.3±0.3							
X11			80	1.0±1.1							
Bolton Point								1.7			
Glabella								0.4			
TMJ									5.1±4.3		
Mand. Notch									4.3±3.9		

Table 2.6: Published detection results for automatically detecting cephalometric landmarks¹. DR denotes detection rate, and error (mm) is defined as the distance between the expert and the landmark located by the system.

¹Yamakawa et al. [160], 1999, achieved a detection rate of 72.7% for the Menton landmark.

	El-Feghi [43] 2002	Chakrabartty [22] 2003		El-Feghi [44] 2003	El-Feghi [45] 2004	El-Feghi [42] 2004	Giordano [52] 2005	Rueda [127] 2006		Yue [162] 2006
Test set size	30	40		134	200	100	26	96		86
Number of Landmarks	15	8		20	20	20	8	28		12
Landmark	DR	DR	DR	DR	DR	DR	DR	DR	Error	DR
Orbitale		99	89	74	74	72		57	2.1±1.2	
Porion								18	3.7±2.1	
Basion										
Sella	100	87	79	77	77	83		39	2.3±1.3	76
Nasion	100	85	87	100	91	77	81	56	2.3±1.8	86
A Point				94	94	69	73	68	2.0±1.4	
Anterior Nasal Spine		98	89	92	92	75		55	2.1±1.3	79
Posterior Nasal Spine		96	81	100	100	83		37	2.7±1.4	83
Apex Upper Incisor				100	98	74				
Incisal Upper Incisor	97.7			100	100	77	92			90
Incisal Lower Incisor				88	79	84	81			
Apex Lower Incisor				100	100	87				
B Point	100			85	85	83	73	44	2.2±1.3	
Pogonion		98	82	100	100	82	81	57	1.8±1.1	
Gnathion	93.3			100	100	79		74	1.6±1.1	
Menton	97.7			84	84	78	92	70	1.6±1.1	98
Gonion	97.7			87	87	71		26	3.9±2.4	86
Posterior Occlusal Plane	97.7			93	87	87				
Anterior Occlusal Plane				68	68	83				
Nose Mid	100			100	100	88				
Upper Lip Mid										
Lower Lip Mid										
Chin Mid	93.3			100	100	80				
<u>Additional Landmarks</u>										
X1										
X2										
X3										
X4										
X5										
X6										
X7										
X11										
Bolton Point										
Glabella										
TMJ										
Mand. Notch										

Table 2.6 (continued)

Chapter 3

Data Preparation

3.1 What are Cephalograms?

A *cephalometric radiograph* or *cephalogram* is a radiograph (also known as an X-ray) of the head, including the mandible, taken in full lateral view which is used for making cranial measurements. The images used throughout this thesis are a selection of cephalograms provided by a practising orthodontist. Each image was digitised from film X-ray as an 8-bit greyscale image that allows for 256 pixel intensities, or grey levels, to be recorded. The resolution of each image is 2056×2588 pixels.

3.2 Defining a Search Area Using a Heuristic

Rather than searching an entire X-ray for the purpose of locating a particular landmark the search is limited to an area defined by a heuristic that is driven by anatomical knowledge. This heuristic is also based on the assumption the head always has the same orientation on the X-ray. For example, it is fair to assume that the upper lip is always going to be located below the tip of the nose. Each landmark is searched for in an area limited by the spatial relationship relative to a *datum point* previously located. It is expected that when traversing an image, the landmark will be located somewhere in this search area. Only a landmark that can be located with a high degree of confidence is used as a datum point for defining the search area for subsequent landmarks. The location of a datum point, marked as the bottom

corner of the ruler in Figure 3.1, can easily be found and in this case is used as a reference for defining the search area for the nose landmark. The search area is shown as the hatched region in Figure 3.1.

It is expected that not only will a smaller search area be a more efficient way of searching a landmark during both training and testing, but detection reliability should be improved because there will be fewer candidate positions compared with a search of the entire X-ray. As such, fewer false alarms are expected because the search is directed towards a much smaller search region.

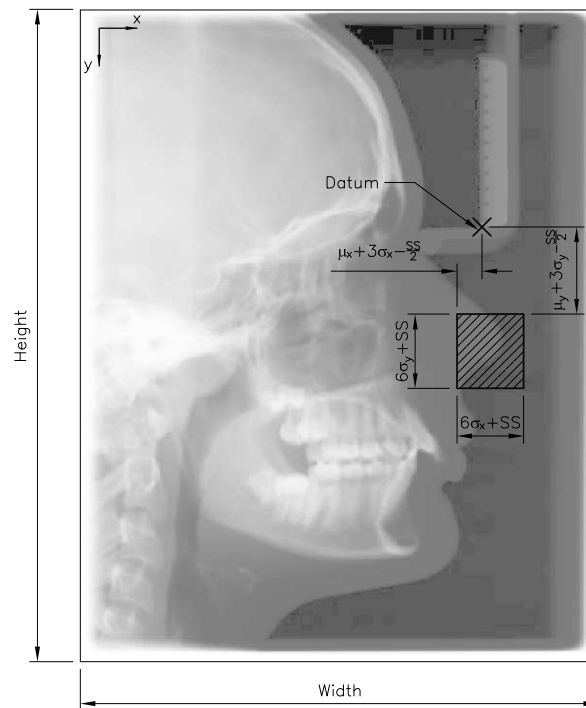


Figure 3.1: An example illustrating the search heuristic used to define the search area for locating the nose landmark.

The hatched region in Figure 3.1 is the search area that has been determined statistically relative from the datum point (bottom corner of the ruler). In this example we have chosen $\pm 3\sigma$ that gives a 99.95% chance that the landmark is located somewhere in this region. The mean, μ , and standard deviation, σ , are calculated using the distance between the known landmark and the datum point from the images within the training set in both the x and y directions.

Figure 3.2 illustrates several search areas that have been defined relative from the datum

point. The search area for locating the bottom corner of the ruler is defined from the upper right of the image while the other search areas are defined relative to the nose landmark. As expected, the size of the search area increases as the distance increases from the datum point. This increase in search area is a function of variance that is related to biological variability. So ideally a datum landmark should be located as close to the landmark as possible in order to minimise the size of the search area.

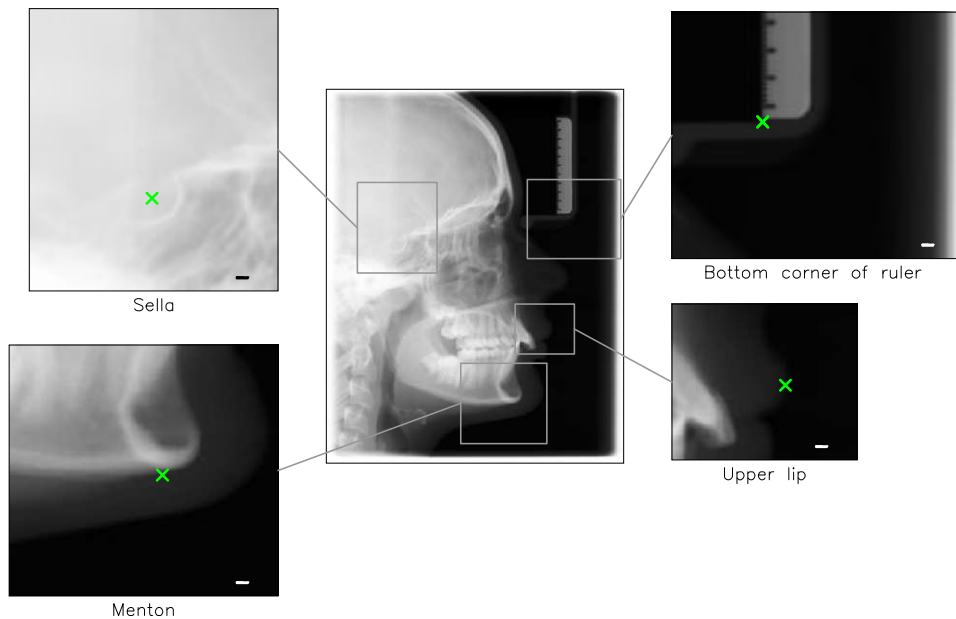


Figure 3.2: A schematic of an X-ray that has defined several search areas that encompass the bottom corner of the ruler, upper lip, menton and sella landmarks.

In the context of our work, each landmark type will be treated as a different domain and thus each landmark type will be considered as a separate detection problem. The reason for treating each type of landmark as a separate detection problem is because of the difficulty and diversity of landmarks and it is considered unrealistic that one program could locate all 56 landmarks. So the approach taken involves dividing the landmark detection problem into 56 independent sub-problems. Each of these sub-problems involves finding a specific landmark in a region of the X-ray. For each landmark, we wish to evolve a program that can be placed over a small window that gives a positive response if the window is centred within 2 mm (5 pixels) of the known position of the landmark.

The work presented in this thesis will use a selection of landmarks that exhibit a range

of detection difficulty (i.e. easy to hard) for the purpose of determining the likelihood that the proposed method will work on all 56 landmarks. Difficulties can include ambiguity where different areas in an image have similar characteristics to the landmark of interest (e.g. the upper and lower lips), and background clutter where it is difficult to discriminate between the location of the landmark and background (e.g. the sella landmark). The nose landmark is subject to minimal biological variation and is deemed relatively easy to detect compared to the sella landmark.

3.3 Image Processing

The main issues from the computer vision literature relating to digital X-ray images as discussed in Section 2.1.1.1 are noise and to a lesser extent the enhancement of objects in low contrast images. The issue of noise is related to the limitation of applying the X-ray process to organic objects. This section will give a brief overview of pre-processing techniques that have been considered for enhancing image quality prior to extracting features for use in GP. These techniques have included noise reduction, contrast enhancement, size reduction and normalisation.

The image of the nose in Figure 3.3(a) illustrates the difficulty faced when viewing soft tissue against the background. Figure 3.3(b) has been greatly enhanced by mapping Figure 3.3(a) using a logarithmic look up table (LUT). A logarithmic LUT maps the output values, G , from the logarithm of input values, F (refer to Equation 3.1). A logarithmic LUT is used to enhance pixels with low intensity values and reduce the spread of high intensity values.

$$G(i, j) = c \log(1 + F(i, j)) \quad (3.1)$$

However, preliminary experimental work (not presented in this thesis) indicated that improving the contrast of soft tissue for human perception does not correlate to better detection programs. Therefore, the contrast of the soft tissue has not been enhanced – enhancement has only been used for the purpose of improving the presentation clarity of soft tissue.

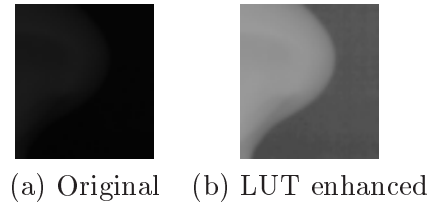


Figure 3.3: An example of two images that contain the nose landmark. The image (b) has been mapped from the original image (a) using a logarithmic LUT.

The *domain independent approach* of GP using pixels as features, as described in Section 1.1 on page 3, will most likely not scale well to large objects because of the large input window required to detect the objects. The number of features in the domain independent approach, or terminals in the context of GP, is a function of the input window dimensions. So as the input window size increases so too does the search space.

For the purpose of developing a strategy for detecting landmarks, the original images have been scaled down to 20% of the original image dimensions. The scale is a trade-off between detection accuracy, or resolution, and training/testing times. The resolution of the original X-ray was approximately 12.3 pixels/mm and scaling produces a resolution of approximately 2.5 pixels/mm. So based on an acceptable tolerance of 2 mm, the maximum error acceptable from the known position with a scaled image is 5 pixels.

Reducing the size of the images also reduces the time of an evolutionary run from 67.5 hours to 2.7 hours¹. Scaling transforms the images from a resolution of 2056×2588 pixels to 411×517 pixels. This reduces the number of genetic program evaluations during training and also reduces the effect of Gaussian noise in the image.

3.4 Pre-calculation of Feature Values

To simplify the experimental work in this thesis, we have extracted the search area as an image and pre-calculated feature values prior to starting an evolutionary run. An example of this is shown in Figure 3.4, whereby an image containing the incisal upper incisor is extracted from an X-ray within the dataset. As described in Section 3.2, images are extracted using

¹Processing time is calculated by averaging the time taken to process 100 generations for 80 evolutionary runs. An evolutionary run is based on evolving a detection program for the sella landmark. Fitness is evaluated by computing the output at every second position in the image and using the highest output to predict the position of the landmark. Processing was performed on an Intel Pentium 4 1.4 GHz CPU.

a heuristic based on anatomical knowledge relative to a previously found datum point. An input window is then traversed across each position on the image which computes features at each pixel location. Although there is no advantage in improving the accuracy of detection programs when pre-computing pixels as features prior to an evolutionary run, we have decided to pre-compute features because the process is a once-off cost that will significantly reduce training time when computing computationally expensive features. The process for pre-computing features is illustrated in Figure 3.4.

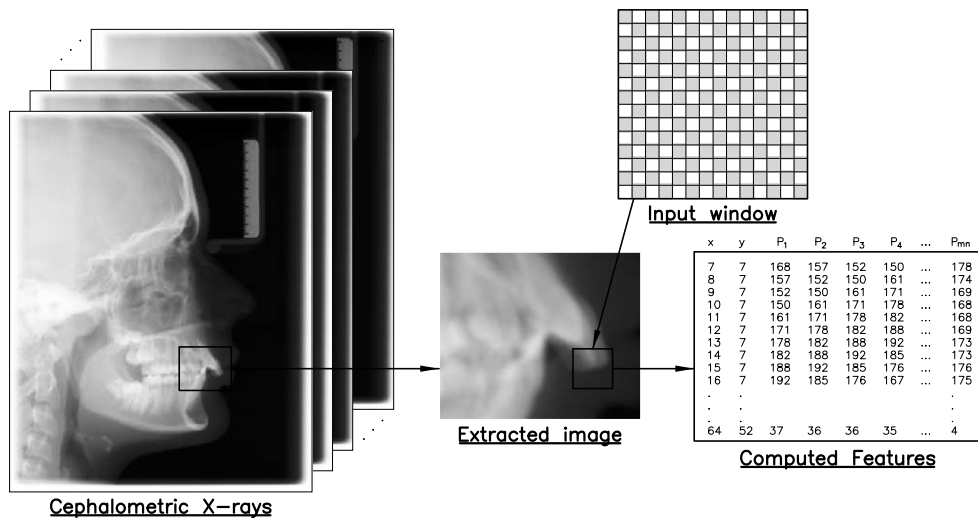


Figure 3.4: Schematic of the process for pre-calculating features within a search area of an X-ray. The search area for locating a landmark is extracted using a heuristic of anatomical knowledge relative to a datum point.

3.5 Dataset Selection

The dataset used in this thesis contains 110 images. The images in the dataset were provided by a practising orthodontist and the images have not been hand chosen and are a random selection of images that are indicative of biological variability from a population of patients. For example, the percentage of images having an overbite should be in keeping with what is expected from real-world data.

Because the dataset consists of a finite number of samples, we have to determine the best way of dividing the data set for both learning and testing the performance of the detector on unseen images. The majority of the work in this thesis uses the holdout method (refer

to Section 2.3.4 on page 20) for estimating accuracy. The holdout method reserves a part of the data set for testing that must not be used in any way during training. We have used a 3/4 and 1/4 split for the training and test sets respectively. Cross-validation of experimental work has been considered, but we have decided the size of the training set and test set from the holdout method to be acceptable for the purpose of comparing experiments. Performing a cross-validation for each experiment would also increase computational resources. A three-fold cross validation will only be performed for establishing detection results for the final method at the conclusion of the thesis. This is to ensure that the final results are based on a larger test set of images.

Chapter 4

Domain Independent Approach: Pixels as Features

4.1 Introduction

In this chapter we apply a method by Zhang et al. [164, 168] which has been successfully used for other object detection problems. The Zhang et al. method described in Section 2.5.2.1 is claimed to be domain independent, meaning that the same method will work unchanged on a range of problems.

The motivation for this chapter is to determine if the *domain independent approach* of genetic programming can be used for the problem of cephalometric landmark detection. The domain independent approach outlined in this chapter has been applied with the minimum of changes from the work performed by Zhang et al., who used genetic programming to locate and classify objects such as heads and tails of different Australian coins in large images, and haemorrhages and micro-aneurisms in retinal images. The main difference between the method proposed by Zhang et al. and the intended use of the domain independent approach is that Zhang et al. classified objects belonging to multiple classes, whereas our approach will treat each type of landmark as a separate detection problem. Another main difference is that Zhang et al. in [164, 168] uses a population size between 100-700 individuals which is increased with the difficulty of the detection problem, whereas this chapter uses a population

size of 100 individuals.

The domain independent approach will be applied to three landmarks of increasing detection difficulty. The landmarks are: the tip of the nose which is easy; the incisal upper incisor which is of medium difficulty; and the sella landmark which is very difficult. Examples of these three landmarks are shown in Figure 4.1. The objective is to locate each landmark within 2 mm, or 5 pixels, of the known location. The crosses in Figure 4.1 correspond to the actual position of the landmark. The circle centred on the cross of each image is a tolerance of 2 mm. If the predicted position of the landmark is within the allowable tolerance, the location of the landmark has been correctly found. If not, the position of the landmark is recorded as a false alarm.

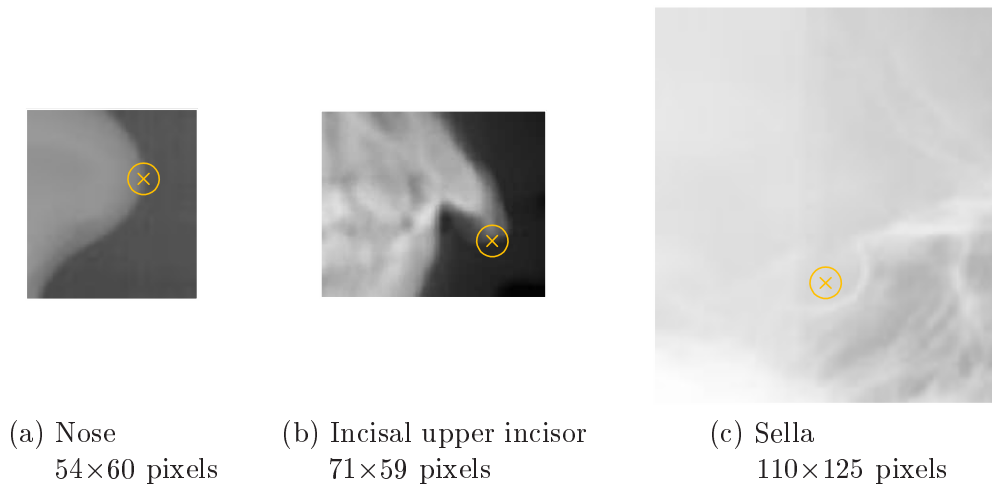


Figure 4.1: Images shown from the left contain the nose, incisal upper incisor and sella landmarks. The landmarks represent a range of detection difficulty from easy to difficult. The cross indicates the known position of the landmark. The circle is the tolerance from the known position, or allowable error, that is considered acceptable for a cephalometric analysis.

4.2 Methodology

The landmark detection approach involves applying a program to an image, in moving window fashion, to find the landmark. The success of the program is measured by the fitness function. Inputs to the evolved program will be a set of features established by partitioning the input window into an array of pixels (refer to Figure 4.4). Each feature is a pixel intensity value of

a pixel within the array. Sections 3.2 and 3.3 describe a process and the rationale for reducing the number of evaluations by extracting an image using *a priori* anatomical knowledge and scaling the image to 20% of the original size. This process reduces the number of program evaluations during training and also reduces the effect of Gaussian noise. It is worth noting that no image processing, e.g. contrast enhancement, has been performed prior to extracting the search area from each X-ray.

An outline of the domain independent approach for evolving a detector to locate landmarks is shown in Figure 4.2. The following step-by-step description gives a more in depth explanation of the methodology depicted in Figure 4.2.

1. Assemble a database of images that consists of the known positions of landmarks to be located.
2. Reserve some images as a *test set* for the purpose of measuring detection performance.
3. Determine the size of the square input window which will contain enough distinguishing information to permit the landmark to be identified.
4. Invoke an evolutionary process to generate a program that can determine whether a landmark is located within 2 mm of the actual position.
5. Apply the generated program as a moving template to the reserved test images from step 2 and obtain the positions of the landmarks. Calculate the detection rate and the false alarm rate on the test set as a measure of performance.

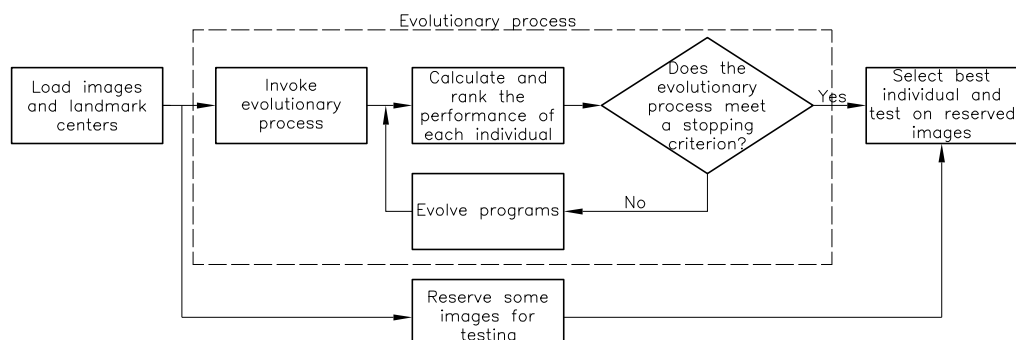


Figure 4.2: Diagram illustrating the evolutionary process for both training and evaluating the performance of a detection program.

4.3 Configuration of Genetic Programming

This section describes how genetic programming will be used for the task of landmark detection. Each section will give an outline of the main components for configuring genetic programming to the problem of landmark detection. The main components include the terminal set, function set, fitness evaluation and genetic programming parameters.

4.3.1 The Terminal Set: Pixels as Features

The terminals used in the domain independent approach are composed of a two-dimensional array of pixel values that are made available as part of the evolutionary search strategy. The array of pixels is contained within an input window of a pre-determined square size that is traversed across each position in the image. The traversing process is illustrated by moving the centre of the input window, in a scanning manner, to each pixel that is represented as a white dot as shown in Figure 4.3.

Determining the size of the input window is based upon another heuristic. This other heuristic determines how much information is required in order for the evolutionary process to generate a solution to distinguish between the landmark of interest and the background. However, choosing the size of the input window is a compromise between containing enough distinguishing information so the landmark can be found and generating a large terminal set which may contain redundant (or extraneous) features. A discussion of previous research in [74] in Section 2.5.2.4 indicated that the genetic search encounters substantial loss of efficiency when extraneous features are contained within the terminal set since the extraneous features add to the complexity of the search.

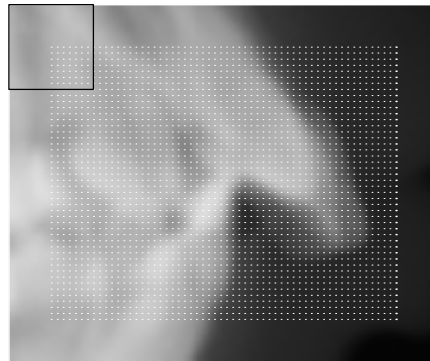


Figure 4.3: The input window shown as the black square is moved in a scanning process to each position in the image corresponding to the white dots.

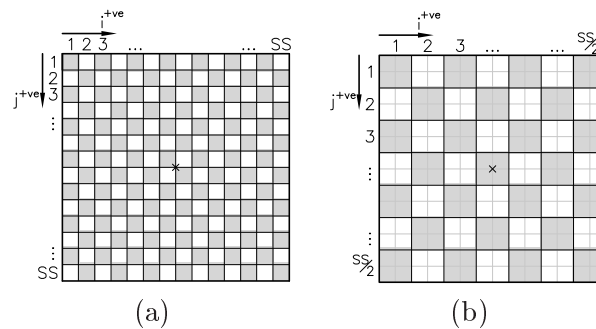


Figure 4.4: Images (a) and (b) illustrate a 14×14 pixel input window, whereby each terminal is represented by a single pixel and an average greylevel intensity of four neighbouring (2×2) pixels respectively. The input window has been divided into SS^2 and $SS^2/4$ sub-regions, centred on ‘x’ at position (x, y) on an image. These are referred to as pixel based features that represent the different terminals available for the evolutionary process.

The input window shown in Figure 4.4 illustrates the spatial relationship of terminals within an input window. These terminals are referred to as *pixel based features*. The size of both input windows in Figure 4.4, which is also referred to as square size (SS), is 14×14 pixels. Each terminal in Figures 4.4(a) and 4.4(b) is a real value represented by a single pixel and an average greylevel intensity of four neighbouring pixels respectively producing a terminal set of 196 and 49 terminals respectively. A sub-region with a size of two indirectly scales the image, which reduces the resolution by a further 50%.

Table 4.1 defines the size of the input window used for the experiments in this chapter. The second column indicates the number of features in the terminal set. The size of the terminal set is a function of the input window size and sub-region size. The size of the input

window is based on the need to contain enough relevant information to enable differentiation between the landmark of interest and the background.

Landmark	Square Size (pixels)	Size of Sub-regions		
		1×1	2×2	5×5
Nose	14	196	49	
Incisal upper incisor	14	196	49	
Sella	40	1600		64

Table 4.1: The sizes of the terminal set that are made available for use during the genetic search. The number of features in the terminal sets are a function of input window size and sub-region size.

4.3.2 The Function Set

The functions shown in Table 4.2 are the operators most commonly used by genetic programming for image-related applications as described in Section 2.5.2.5. The functions $+$, $-$, $*$, $/$ are four arithmetic operators that, when used, can allow the formation of both linear and non-linear functions. The $+$, $-$ and $*$ have their usual meanings, while ‘ $/$ ’ represents a protected division which constitutes the usual division operator, except that a divide by zero produces INT_MAX. The use of other operators used in image-related applications were discussed in Section 2.5.2.5, however, it was unclear how these could enhance the quality of solutions.

Functions		
Function	Arity	Definition
$+$	2	$a + b$
$-$	2	$a - b$
$*$	2	$a \times b$
$/$	2	$\begin{cases} \frac{a}{b} & \text{if } b \neq 0 \\ \text{else INT_MAX} \end{cases}$

Table 4.2: Definition of operators used in the function set.

4.3.3 Fitness Evaluation

The process of measuring an individual’s fitness is performed in three distinct phases. Initially the program is applied to the training data, this is akin to moving the input window across an image, which produces an output needed to predict the position of the landmark. The

method used for predicting the position of the landmark is the same as that used by Zhang et al.'s [164] and will be explained in greater detail below. The predicted positions are then compared with the actual location of the landmark and the detection and false alarm rates are then calculated.

The aim of the fitness metric presented in this section is to maximise detection rate (correct prediction) and to minimise false alarm rate (incorrect prediction). The fitness of a program is calculated by computing both detection rate and false alarm rate. The fitness is evaluated as follows:

1. Zhang et al.'s [164] approach for predicting the position of the landmark involves three separate steps. These steps are as follows:
 - (a) A program is applied as a moving input window, shown as the black square in Figure 4.5(b), across a training image with the program's output evaluated at each pixel location. The output of the program, *Output*, is a floating point number which determines whether the position in the image should be classified as a landmark or background.
 - (b) Zhang et al. uses a multi-class classification strategy for classifying objects, however, because we are evolving one program for each landmark, a classification strategy considering only two classes (landmark/background) is required. Therefore, we have chosen zero as a decision point between the two classes, which has also been commonly used by GP researchers in classification problems when differentiating between binary classes as discussed in Section 2.5.2.6. The position at each pixel location is labelled according to the classification strategy shown in Figure 4.5(a). If the *Output* is positive, the location is labelled as "landmark"; otherwise, the location is labelled as "background". The labelled positions are described as *binary outputs* within a *detection map*. An example of a detection map is shown in Figure 4.5(c) whereby the white and grey regions depict positions labelled as landmark and background respectively.
 - (c) The detection map is then used to localise the position of the landmark. The landmark's position is predicted by scanning each position within the detection

map that was labelled as “landmark” and recording its (x, y) position. However, another landmark cannot be predicted within $\frac{SS}{2}$ pixels of a landmark’s position previously predicted during the scan of the detection map. The detector shown in Figure 4.5(a) has subsequently recorded a landmark at three locations which are depicted as the green and red crosses. The green and red crosses correspond to the correct and incorrect positions respectively.

2. A comparison is made between the predicted position and the actual location of the landmark. A match (true positive) occurs when the comparison is within a set tolerance of 2 mm (5 pixels). If the comparison is not within the set tolerance, the landmark for the respective image is recorded as a false alarm.
3. The performance of a program is measured by iteratively applying steps one and two to each image in the training set and then calculating the detection rate (DR) and false alarm rate (FAR). The fitness is computed as per equation 4.1.

$$Fitness = A \times FAR + B \times (1 - DR) \quad (4.1)$$

where FAR is the false alarm rate and DR is the detection rate. A and B are constants that provide a balance between false alarm rate and detection rate. The constants are used to transform a multi-objective problem into a scalar optimisation problem as discussed in Section 2.2.1.1. Zhang et al. in [164, 166, 168] used values of 50 and 1000 for A and B respectively.

Alternatively, the fitness function can be represented as,

$$Fitness = FAR + \frac{B}{A}(1 - DR)$$

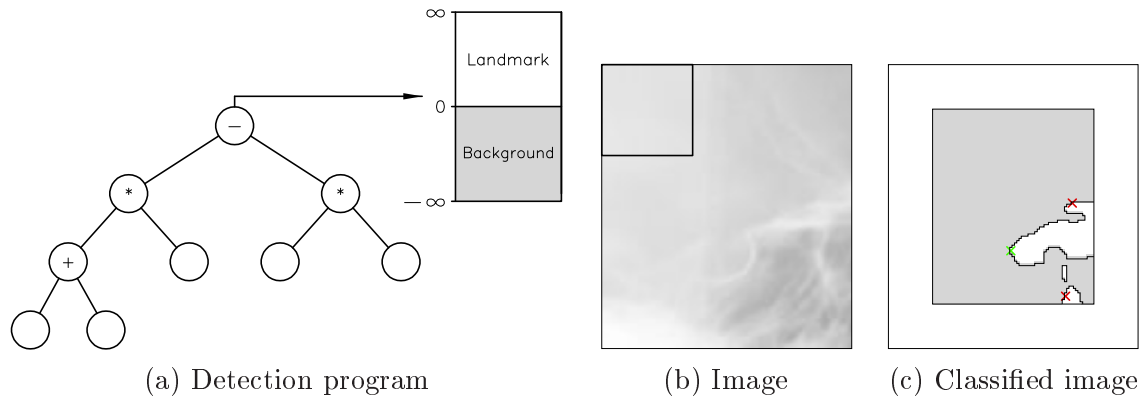


Figure 4.5: The series of images illustrate the strategy for predicting the position of a landmark. A program (a) is traversed across the image (b) at each pixel location and the program's output is calculated. The output of the program determines if the position is to be labelled a landmark or background. The labelled positions are described as the binary output in the detection map (c). The green and red crosses correspond to the correctly and incorrectly predicted positions respectively.

4.3.4 Parameters

Table 4.3 indicates the genetic programming run-time parameter values used by genetic programming during training. The parameter values are based on research by Zhang et al. in [164, 165, 166, 168] whereby genetic programming was used to solve similar types of detection problems to the cephalometric landmark detection problem discussed here. A description for each run-time parameter is given in Section 2.5.3.

4.4 Results

The results given in this section attempt to establish if the domain independent approach using genetic programming, is able to locate cephalometric landmarks accurately enough for a cephalometric analysis. To determine the efficacy of this strategy, the method is tested on three landmarks of varying levels of detection difficulty ranging from easy to hard.

The aim of the genetic search is to minimise the fitness function and achieve a fitness score of zero. The fitness function is a combination of detection rate and false alarm rate. A fitness score of zero only occurs for a program that achieves a detection rate of 100% and a false alarm rate of 0%. Each evolutionary run is terminated when either the fitness score is zero or 100 generations have been completed. The result from each experiment is based on

Parameters	
Population size, M	100
Maximum generation, G	100
Maximum depth, D	8
Initial maximum depth, d	6
Probability of:	
Reproduction, P_R	0.10
Crossover, P_C	0.70
Mutation, P_M	0.20
Probability of crossover at:	
Terminal	0.15
Function	0.85
A	50
B	1000
Terminal Set	refer to Section 4.3.1
Function Set	refer to Section 4.3.2
Tolerance (pixels)	5 (2 mm)

Table 4.3: Run-time parameters used during the genetic search for evolving detection programs for the nose, incisal upper incisor and sella landmarks.

80 evolutionary runs.

Table 4.4 shows training and test results for three different landmarks which provides a comparison of the mean fitness scores, calculated using the best individual at the end of 100 generations, for 1×1 and 2×2 pixel sub-regions. The experiments for the sella landmark using 1×1 or 2×2 pixel sub-regions and an input window square size of 40 pixels was not conducted as this would have created a terminal set of 1600 and 400 terminals respectively; in Section 2.5.2.4 we discussed that the efficiency of GP decreases when the terminal set contains too many extraneous or redundant features. The size of the terminal set was subsequently decreased by using a 5×5 pixel sub-region.

Table 4.4 shows the average fitness score for three landmarks increasing with the relative increase in detection difficulty between the three landmarks. The exact reason as to why the fitness score varies with the different landmarks will be explained in detail later. To determine if there is a difference between the two sub-region sizes, a two-sample t test will be conducted to compare if either of the two sub-region sizes, i.e. 1×1 pixel or 2×2 pixels, have come from the same population. To validate this experiment, the hypothesis will be tested on the nose and incisal upper incisor landmarks.

$$H_0 : \mu_{1 \times 1} = \mu_{2 \times 2}$$

$$H_1 : \mu_{1 \times 1} \neq \mu_{2 \times 2}$$

$$p\text{-value} \begin{cases} \leq \alpha & \text{reject } H_0 \text{ in favour of } H_1; \\ > \alpha & \text{do not reject } H_0. \end{cases}$$

		Size of Sub-regions		
		1×1	2×2	5×5
Nose	Average Fitness	57.06	55.41	
	<i>p</i> -value	0.666		
Incisal	Average Fitness	71.76	71.81	
	<i>p</i> -value	0.988		
Sella	Average Fitness			142.79
	<i>p</i> -value			

Table 4.4: Comparison of average fitness score between 1×1 pixel sub-region and 2×2 pixel sub-region for the nose and incisal upper incisor. Average fitness is calculated from the best individual's fitness score from each run for 80 evolutionary runs.

Since the *p*-value exceeds the critical 'cut-off' boundary of 0.05 for program fitness of the best individual for both landmarks, the null hypothesis is accepted indicating that the fitness of programs evolved from both sub-regions are from the same population. This indicates that both terminal sets, on average, will produce programs with the same fitness. It is reasonable to expect that if the size of the sub-region is increased beyond a certain size, the performance of programs during and at the end of the evolutionary process will decrease. This is related to a loss of pixel information as the sub-region size is increased.

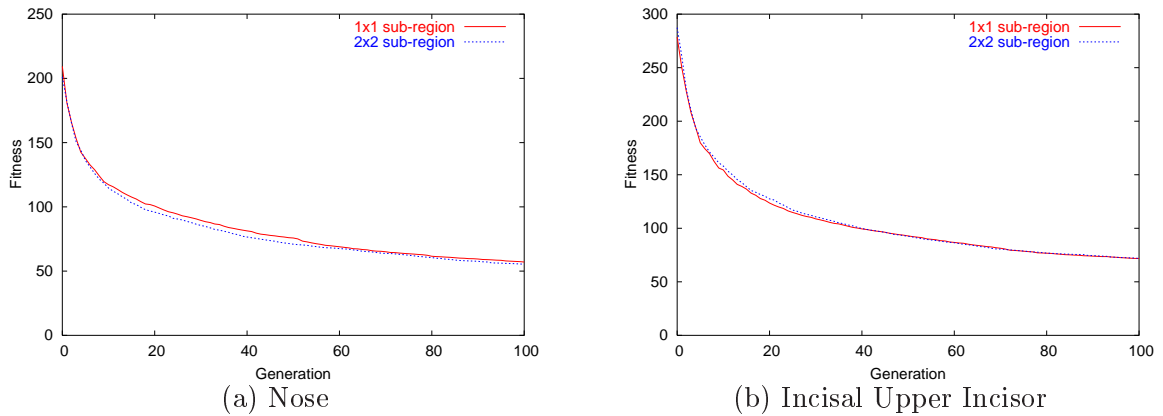


Figure 4.6: Comparison of average fitness between a 1×1 pixel and 2×2 pixel sub-region. The average fitness scores are calculated from the fitness score of the best individual at each generation for 80 evolutionary runs.

The graphs in Figure 4.6 show the average fitness calculated from the best individual's fitness score at each generation for 80 evolutionary runs. The graphs illustrate that there is negligible difference between the rate of convergence during the evolutionary search for both sub-region sizes of both landmarks.

		Training		Test	
		Sub-region size		Sub-region size	
		1×1 pixel	2×2 pixels	1×1 pixel	2×2 pixels
Nose	Average				
	detection (%)	97.32	97.82	97.64	96.81
	false alarm(%)	60.46	67.23	58.56	66.85
	Best program				
	detection (%)	100	100	96.30	92.59
	false alarm(%)	13.41	9.76	11.11	14.81
Incisal	Average				
	detection (%)	98.04	97.53	94.72	94.82
	false alarm(%)	104.37	94.22	96.11	87.50
	Best program				
	detection (%)	100	100	100	96.30
	false alarm(%)	67.47	78.31	85.19	59.26
		Sub-region size		Sub-region size	
		1×1 pixel	5×5 pixels	1×1 pixel	5×5 pixels
Sella	Average				
	detection (%)		99.34		97.41
	false alarm(%)		272.47		269.35
	Best program				
	detection (%)		100		96.30
	false alarm(%)		202.44		214.81

Table 4.5: Detection results for the nose, incisal upper incisor and sella landmarks using the pixel based features defined in Section 4.3.1. Averages are calculated using the best individual discovered from each run for 80 evolutionary runs.

Table 4.5 shows the detection and false alarm rates using the best individual at the completion of evolutionary runs. It is worth noting that no run achieved a fitness score of zero, i.e. a detection rate of 100% and a 0% false alarm rate, and so the results presented are from the best individual at the end of 100 generations. Although there is no significant difference in detection accuracies between the three types of landmarks, the false alarm rate increases significantly with the level of difficulty. This is a result of both anatomical variability and clutter found in the sella images, as evident by the images in Figure 4.10.

$(- (- (/ P_4 (+ (/ P_{14} P_{24}) P_{10})) (- (- P_{23} P_{43}) P_{36})) (+ (- (* (/ P_{28} (/ (/ P_{14} P_{11}) (+ (+ P_{20} P_{17}) P_{44}))) (/ (/ (+ (- P_{39} P_{27}) P_{13}) (/ P_{40} P_{38})) (* P_{14} P_{30}))) (- (* (- P_{23} P_2) P_8) P_{11})) (* (* (/ (/ (+ (- P_{43} P_{27}) P_{13}) (/ P_{11} P_{38})) (/ (/ P_{14} P_{24}) (+ P_{30} P_{44}))) (/ (/ (+ (- P_{43} P_{49}) P_{13}) (/ P_{11} P_{38})) P_{16})) (/ P_{16} (* P_{15} P_{30}))))))$

Figure 4.7: A sample program for locating the nose landmark, where P_i represents the average pixel intensity value of a sub-region labelled P_i and i is the i th sub-region. Note: The array of sub-regions $(P_{(0,0)} \dots P_{(n,n)})$ depicted in Section 4.4 has been converted into a vector $(P_0 \dots P_{n^2})$, where n^2 is the number of sub-regions within the input window. Fitness score = 4.878 (FAR=9.75% and DR=100%).

The program used to locate the nose landmark in Figure 4.7 was the fittest program at the end of 80 evolutionary runs. A random selection of images containing the nose landmark are shown in Figure 4.10(a-l). The detection program in Figure 4.7 is applied to the images in Figure 4.10. The position of the nose landmark for all these images are correctly detected with only one false alarm as indicated by the red cross in image (g).

$(- P_{35} (/ (/ (+ (- (- (- P_{41} P_{15}) P_{11}) P_{37}) (- (* (- P_{23} P_{15}) P_{11}) P_{16})) P_{39}) (/ P_{11} P_{16}))) (* (- P_{17} (+ (- P_{29} (/ (- (- P_{41} P_{15}) (- P_3 P_{32})) (- P_3 (- P_3 P_{32})))) P_{21})) P_{36}))$

Figure 4.8: A sample program for locating the incisal upper incisor landmark, where P_i represents the average pixel intensity value of a sub-region labelled P_i and i is the i th sub-region. Note: The array of sub-regions $(P_{(0,0)} \dots P_{(n,n)})$ depicted in Section 4.4 has been converted into a vector $(P_0 \dots P_{n^2})$, where n^2 is the number of sub-regions within the input window. Fitness score = 39.157 (FAR=78.31% and DR=100%).

$$\left(P_{35} - \frac{P_{16}(P_{41} - P_{15} - P_{16} - P_{37} + P_{11}(P_{23} - P_{15} - 1))}{P_{39}P_{11}} \right) \left(P_{17} - \left(P_{29} - \frac{(P_{41} - P_{15} - P_3 + P_{32})}{P_{32}} + P_{21} \right) \right) P_{36} \quad (4.2)$$

The program used to locate the incisal upper incisor landmark in Figure 4.8 is the fittest program at the end of 80 evolutionary runs. The equivalent formula is shown as Equation 4.2. A random selection of images containing the incisal upper incisor landmark are shown in Figure 4.10(a-j). The positions of the incisal upper incisor landmark for all these images are correctly detected. However, at least one false alarm was found in each image with the exception of image (h) and (j). This is a significant increase in false alarm rate as indicated by the number of red crosses compared to the nose landmark. Training results produced a false alarm rate and detection rate of 78.1% and 100% respectively. This indicates that,

on average, the detection program has incorrectly predicted the position of an additional phantom landmark on four out of five images.

```
(+ (- (- (- P62 (* P2 P54)) (- (* (+ (+ P6 P33) P26) (/ P16 P31)) (- P51 (+ P43
P58)))) (- (+ (+ P6 (+ P5 P27)) (+ (+ (* P35 P40) P38) P38)) (- P25 (- (+ P29
P9) P25)))) (- (- (* P24 P8) P43) (- (- (+ P6 P32) (/ P13 P30)) (* P24 P46))))
```

Figure 4.9: A sample program for locating the sella landmark, where P_i represents the average pixel intensity value of a sub-region labelled P_i and i is the i th sub-region. Note: The array of sub-regions ($P_{(0,0)} \dots P_{(n,n)}$) depicted in Section 4.4 has been converted into a vector ($P_0 \dots P_{n^2}$), where n^2 is the number of sub-regions within the input window. Fitness score = 101.22 (FAR=202.44% and DR=100%).

The program used to locate the sella landmark in Figure 4.9 is the fittest program at the end of 80 evolutionary runs. A random selection of images containing the sella landmark are shown in Figure 4.10(a-h). The position of the sella landmark for all these images are correctly detected. However, at least two false alarms were found in each image with the exception of image (b). Although the detection rates were similar between the three landmark types, the results indicate that false alarm rate increases with the level of detection difficulty.

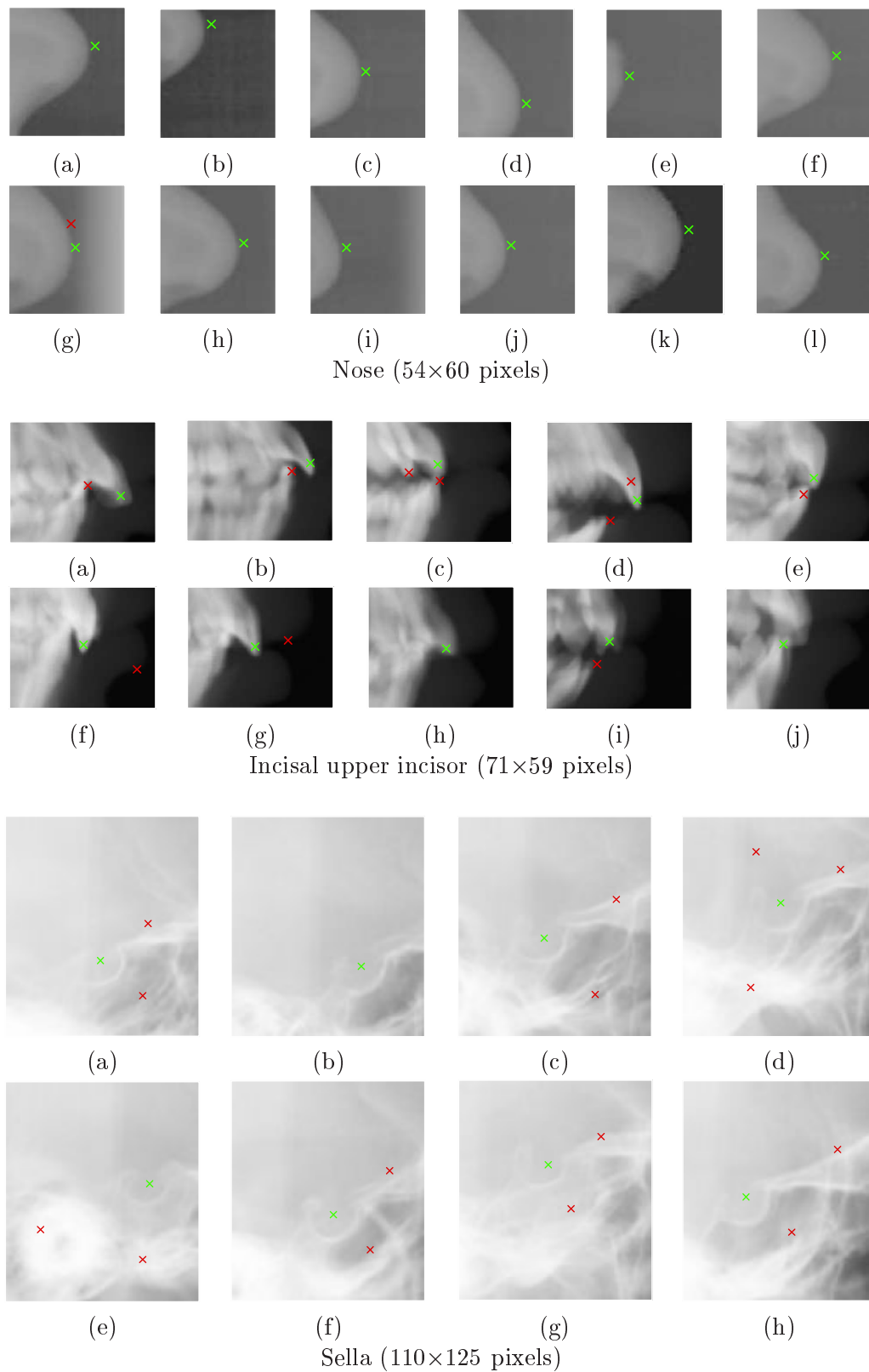


Figure 4.10: A selection of images showing the correctly found position (green cross) and incorrect position (red cross) of three landmarks of increasing detection difficulty. The landmarks, from easy to hard, are the nose, incisal upper incisor and sella landmarks.

The fitness function in Equation 4.1 is a weighted sum of *detection rate* and *false alarm rate* with the aim to minimise fitness. Figure 4.11 shows fitness graphs for terminals that have been calculated from a 2×2 pixel sub-region. The graphs are averages calculated using the best individual found at each generation for 80 evolutionary runs. The detection and false alarm rates are also illustrated. The graphs for each landmark show that almost 100% detection accuracy was achieved at the first generation. However, there was a false alarm rate of 310%, 458% and 415% for the nose, incisal upper incisor and sella landmarks respectively and at generation 100, at the point where the evolutionary process was terminated, the false alarm rate had improved to 67%, 94% and 273% respectively. A fitness score of zero was never achieved for a detection program for any of the evolutionary runs. The graphs illustrate that the fitness function rewards programs for achieving high detection rates and so then the aim of the evolutionary search becomes focused on minimising the number of false alarms.

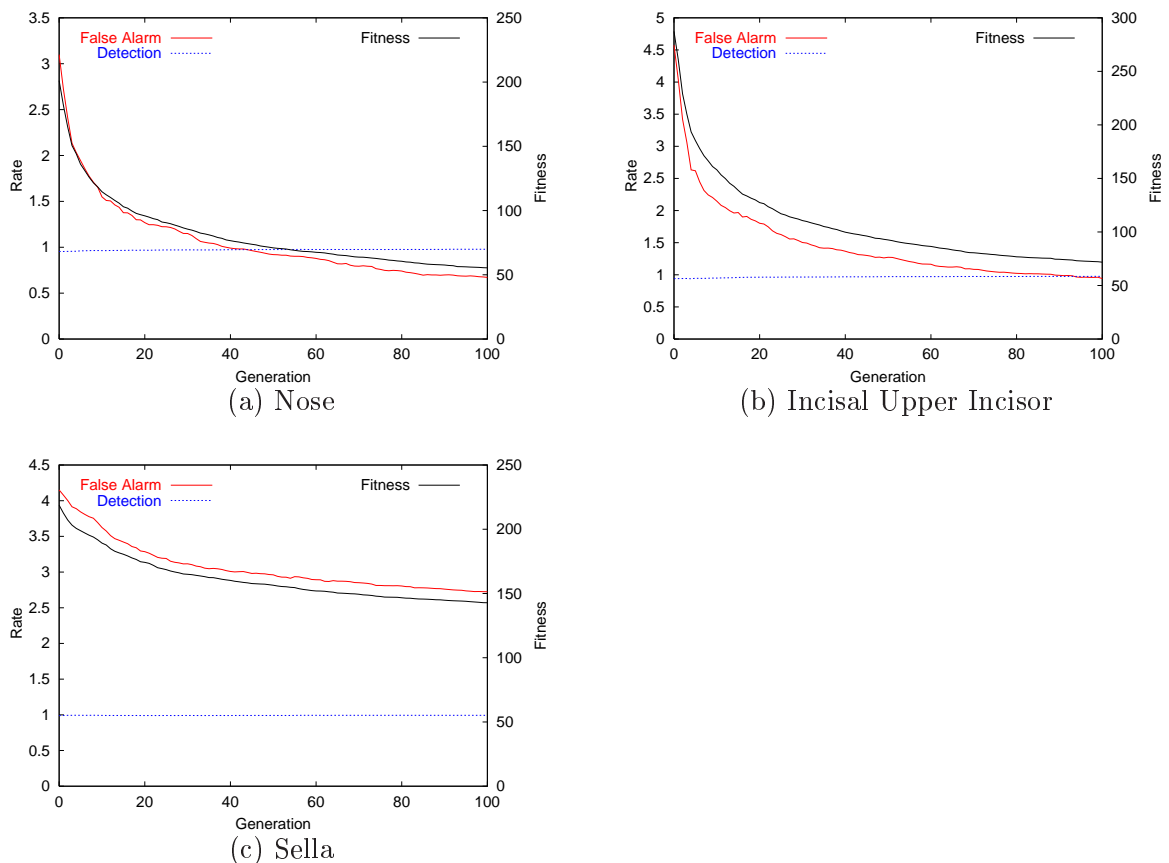


Figure 4.11: Fitness graphs for the nose, incisal upper incisor and sella landmarks. Averages are calculated using the best individual at each generation for 80 evolutionary runs.

4.5 Summary

The results demonstrate that the method by Zhang et al. [164, 168], which claims to be domain independent, were able to accurately locate simple landmarks with a low number of false alarms. Although this method was shown to be successful on simple landmarks, a large number of false alarms were found on more difficult landmarks such as the incisal upper incisor and sella landmark. False alarms are unacceptable for a cephalometric analysis and therefore further work is required to investigate if the genetic programming paradigm is able to reduce the false alarm rate. Therefore, further exploratory work of the proposed genetic programming method is required to determine if factors such as fitness evaluation or other features will reduce false alarm rate when locating difficult types of landmarks.

Chapter 5

Domain Dependent Approach: Handcrafted Shapes

5.1 Introduction

The purpose of the previous chapter was to determine if the domain independent approach of genetic programming using pixel based features was able to locate landmarks accurately enough for a cephalometric analysis. The results of the experimental work showed that the detection rates were very good but the false alarm rates were unacceptably high, especially for the harder landmarks. Therefore the purpose of this chapter is to determine how the domain independent approach can be re-formulated to reduce the false alarm rate and at the same time predict the position of the landmark accurately enough for a cephalometric analysis.

In this chapter we will provide a foundation for genetic programming that will be used in subsequent chapters. The work using the domain independent approach of genetic programming and pixel based features from Chapter 4 will be used as a benchmark for comparing performance.

5.1.1 Chapter Goals

Based on the outcome of the results from the domain independent approach using pixels as features, several questions are posed for reducing false alarm rate. In this chapter, the following research questions will be investigated.

1. Can the method of evaluating fitness be improved to reduce the false alarm rate without affecting detection performance? What is a good fitness metric that can be used as a measure of a program's performance?
2. Can replacing the pixel based features with features calculated using handcrafted shapes improve a program's detection performance?
3. Will the inclusion of other operators commonly used by genetic programming in image related applications improve the detection performance?

5.2 Methodology

The use of genetic programming for the purpose of landmark detection in this chapter is similar to the methodology described in Section 4.2. The landmark detection approach involves applying a program to an image, in moving window fashion, to locate the position of the landmark. The success of the program is determined by the fitness function. Terminals made available to the evolved program are features based on partitioning areas surrounding a landmark by *handcrafting shapes* within the input window. The shapes are intended to discriminate the landmark from background. The handcrafted shapes are shown in Figure 5.2. The features used in this chapter are calculated using the mean and standard deviation of pixel intensities within each shape.

The following step-by-step description along with Figure 5.1 is similar to the methodology in Section 4.2 with the addition of developing a set of handcrafted features.

1. Assemble a database of images with the known positions of landmarks to be located.
2. Reserve some images as a test set for the purpose of measuring detection performance.

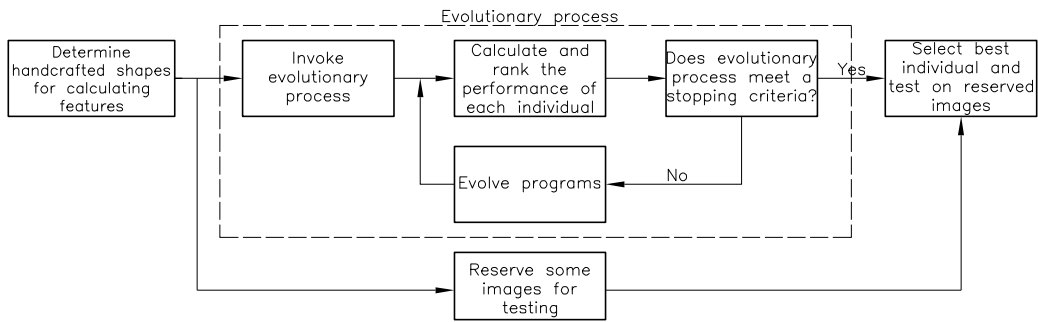


Figure 5.1: Diagram illustrating the genetic programming methodology for evolving and evaluating detection programs using handcrafted features.

3. Determine the size of the square input window centred on the landmark that will contain enough distinguishing information to permit the landmark to be identified.
4. Manually determine a set of shapes within the input window that are to be applied to the training and test images. The handcrafted shapes are specific to capturing landmark characteristics as well as discriminating against background (refer to Figure 5.2).
5. Invoke an evolutionary process to generate a program which can determine whether a landmark is located with 2 mm of the actual position.
6. Apply the generated program as a moving template to the reserved test images from step 2 and obtain the positions of the landmarks. Calculate the detection rate and the false alarm rate on the test set as the measure of performance.

The following sections describe the evolutionary process of step 5 in detail.

5.3 Genetic Programming Configuration

5.3.1 The Terminal Set

The domain independent approach using pixels as features demonstrated that the number of false alarms increases with the complexity of the detection problem. Because each type of landmark is distinct in shape, greyscale and contrast, it is expected that a set of handcrafted shapes specific to a landmark will give a better detection performance compared to pixels as

features used in the previous chapter. We anticipate that landmark specific features will be more useful for discriminating between the location of interest and background.

In Section 2.5.2.4 we reviewed the most common types of features for complex detection problems. These were calculated using simple statistics, i.e. the mean and standard deviation, of pixel values within pre-defined shapes. The features presented in this chapter correspond to the different shapes shown in Figure 5.2 with their resulting means and standard deviations calculated from each region based on grey level intensity. As described in Section 2.5.2.4 when using genetic programming in object detection problems, terminals correspond to image features. In addition to these features, a terminal that generates a random number in the range of $[0, 255]$ is included in each terminal set.

5.3.2 The Function Set

The functions $+$, $-$, \times , $/$ are four arithmetic operators used by genetic programming during training, which is identical to the set used in the previous chapter shown in Table 4.2 on page 68.

5.3.3 Genetic Programming Parameters

The genetic programming parameters to be used during training are identical to those used in the previous chapter as shown in Table 4.3 on page 72.

5.4 Variations of Fitness Evaluation

The aim of this section is to establish if the method for evaluating fitness from the domain independent approach using the genetic programming paradigm of Chapter 4 can be reformulated to improve the performance of detection programs for locating cephalometric landmarks. This investigation and subsequent investigations presented in this chapter will be tested on additional landmark types to those presented in Chapter 4. The images shown in Figure 5.3 are representative of the different landmark types presented in this chapter.

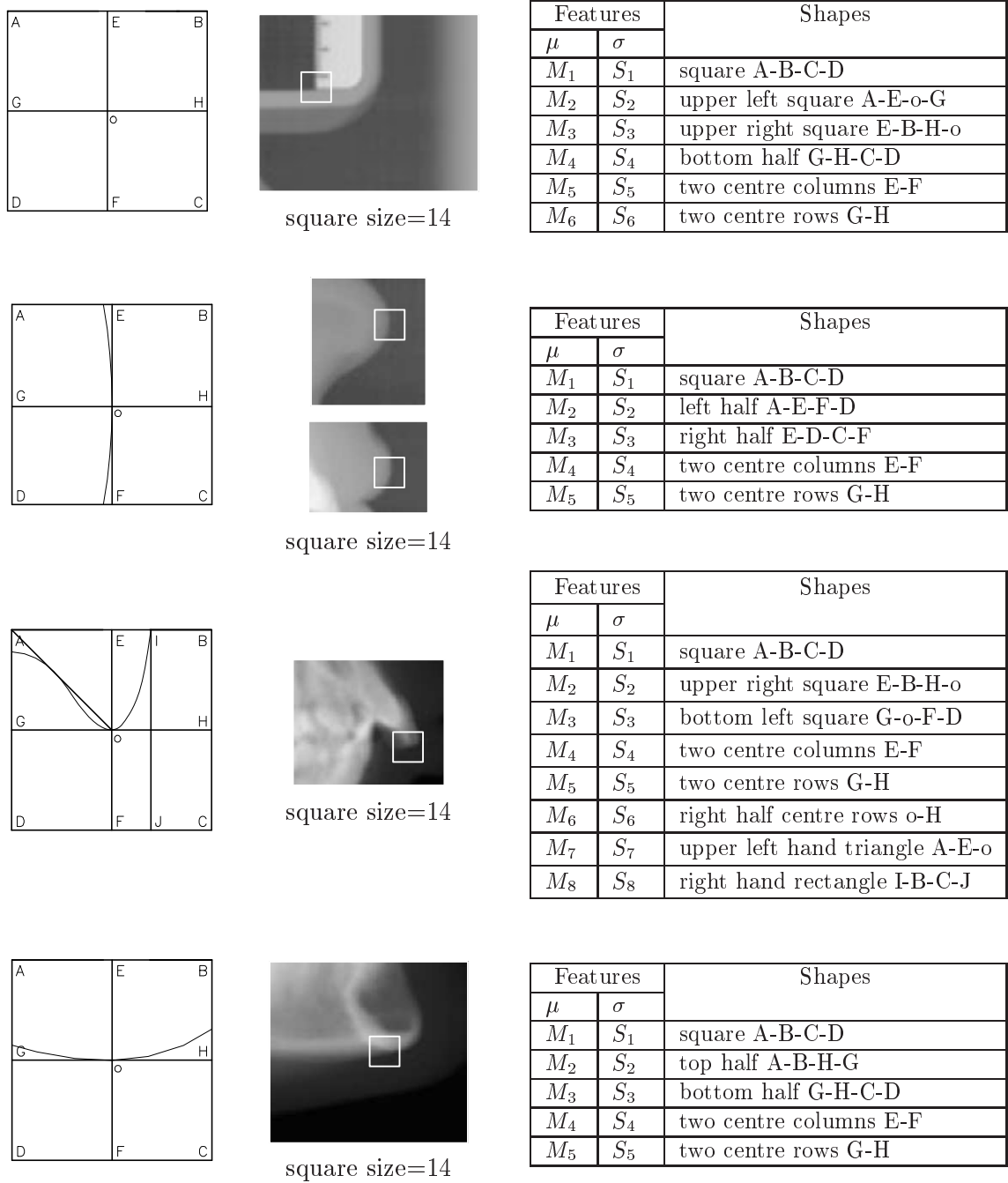


Figure 5.2: The diagrams in the left column depict the shapes used to extract the features for the bottom corner of the ruler, nose, upper lip, incisal upper incisor, menton and sella landmarks. The features consist of the mean and standard deviation calculated for each shape from grey level intensities. The corresponding pictures in the middle column depict the size of the input window – shown as the white square – relative to the image. Note: Images for the bottom corner of the ruler, nose and menton landmarks have had the contrast enhanced to improve clarity.

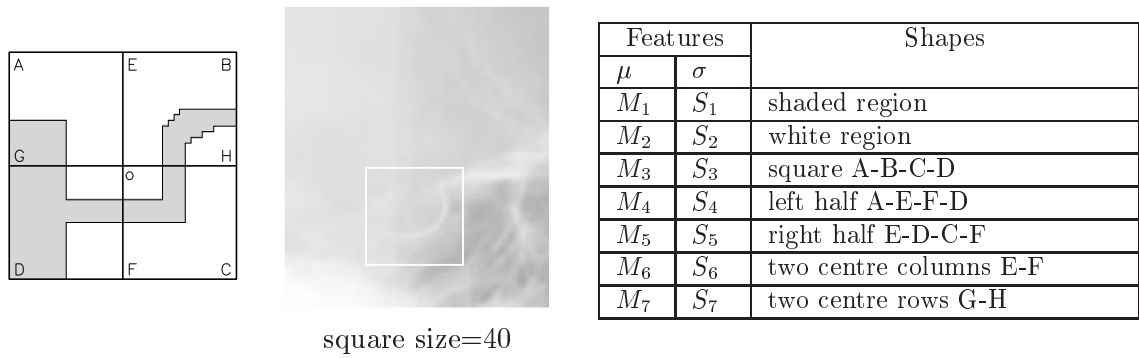


Figure 5.2 (continued)

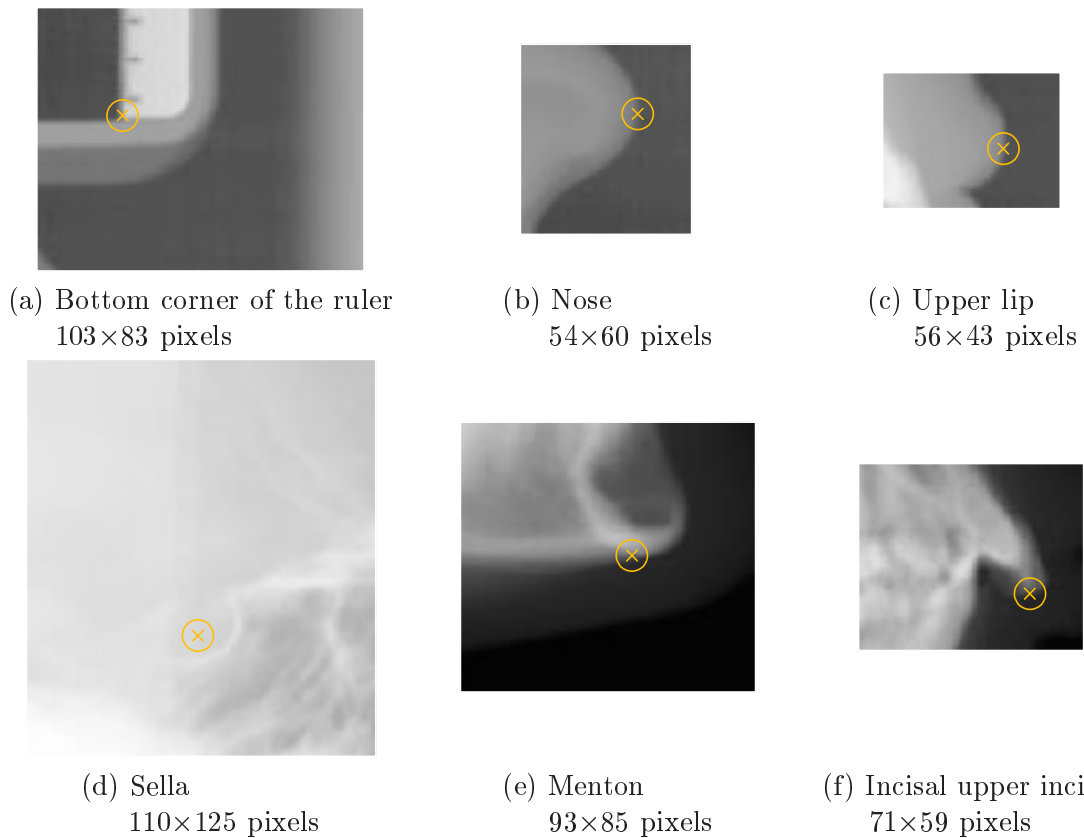


Figure 5.3: Images shown from top left in a clockwise direction contain the bottom corner of the ruler, nose, upper lip, incisal upper incisor, menton and sella landmarks. The landmarks are a range of object detection problems ranging from easy to difficult. The cross indicates the known position of the landmark.

5.4.1 Highest Output

5.4.1.1 Motivation

Results presented in this section use a fitness function that rewards programs on the basis of locating landmarks within the acceptable tolerance. Because we know there is only one landmark in the image we can consider an alternative fitness function that takes this into account. The fitness of a program is measured using only detection rate, $(\frac{\text{Total no. of objects correctly located}}{\text{Total no. of objects}})$. The reason for this will be given below. The $+$, $-$, \times , $/$ operators form the function set that allow both linear and non-linear solutions to be evolved. The function set consists of the most commonly used operators available to genetic programming for solving image related applications. The fitness is calculated as follows:

1. The program is applied as a moving window across a training image and the detection program's output, *Output*, is evaluated at each pixel location. The output of the detection program is a floating point number interpreted as the likelihood that the evaluated position from the image is a landmark centre or background. During training, the highest value of *Output* from each image is used to predict the position of the landmark. The predicted position given by the detection program is then compared with the known true location and the result for the training image is recorded as either a true positive or false alarm. However, an issue with using the highest output to predict the landmark's position is that it has become compulsory for the detection program to predict a landmark's position. Ideally we do not want a detection program to locate the position of a landmark when the detection program returns an ambiguous result, i.e. if a high output occurs in another part of the image.
2. A comparison is made between the predicted position and the known location of the landmark. A match (true positive) occurs when the comparison is within a set tolerance of 2 mm or 5 pixels. If the comparison is not within the set tolerance then the landmark for the respective image is recorded as a false alarm.
3. The performance of the programs is measured by iteratively applying steps one and two to each image in the training set and calculating detection rate (DR).

4. The fitness of a program is measured using Equation 5.1. Equation 5.1 is suitable when it is known there is only one object of interest located in an image.

$$fitness = (1 - DR), \text{ where } DR \text{ is detection rate} \quad (5.1)$$

Previous work in Chapter 4 used the output from the detection program to determine if the position of the landmark should be recorded as either a landmark or background. However, this approach produced a large number of false alarms. Therefore, in this section we use the highest output for predicting the position of the landmark with the expectation of reducing the false alarm rate. This means that only one prediction for locating the landmark will occur; the predicted position of the landmark will be either correct, i.e. within 2 mm of the actual position, or incorrect and recorded as a false alarm. As a result, the sum of detection rate and false alarm rate is one.

The rationale behind the simplified fitness metric of Equation 5.1 is based on the highest output used to predict the position of the landmark and the fitness metric of Equation 4.1 on page 70. The following derivation explains how Equation 5.1 was derived from Equation 5.2.

$$fitness = A \times FAR + B \times (1 - DR), \quad (5.2)$$

Equation 5.3 is an equivalent fitness function to Equation 5.2.

$$fitness = FAR + \frac{B}{A}(1 - DR) \quad (5.3)$$

Observation 1: When one landmark is always located within an image then $FAR = 1 - DR$

$$= \left(1 + \frac{B}{A}\right)(1 - DR)$$

Observation 2: Factors in the fitness function containing only constants, such as $\left(1 + \frac{B}{A}\right)$, can be eliminated from the equation because they have no effect when ranking programs based on fitness score.

$$fitness = (1 - DR)$$

5.4.1.2 Results

To determine whether the genetic programming approach described in Section 5.4.1 can be used to locate craniofacial landmarks, several landmarks have been chosen ranging from easy to most difficult. The different landmark types are shown in Figure 5.3 and the relative positions are shown on a tracing in Figure 2.11.

The results indicate that the genetic programming methodology described in this section has been successfully used to evolve detection programs for a number of cephalometric landmarks. The results for the methodology are presented in Table 5.1. The detection performance on the easier landmarks (nose and incisal upper incisor landmarks) was excellent and the performance on the more challenging sella landmark was also promising albeit with a considerably decreased detection performance. The reduction in detection performance of the sella landmark is caused through the landmark exhibiting a greater variation in an anatomical shape and located in areas that are subject to subtle changes of greyscale. Due to the difficulty of the sella landmark, a non square/rectangular handcrafted shape was created, as shown in Figure 5.2, to assist with improving the evolution of detection programs. While some of the landmarks are ‘easy’ to locate, it is important to note that there is a large variation in human shapes and sizes as is evident from Figure 5.5, e.g. incisal upper incisor, and that the accuracy obtained is a non trivial achievement.

		Training		Testing	
		Detection	False alarm	Detection	False alarm
Bottom corner of the ruler	Average	100	0	99.77	0.23
	Std dev.	0	-	1.08	-
	Best program	100	0	100	0
Nose	Average	99.36	0.64	97.69	2.31
	Std dev.	0.89	-	1.90	-
	Best program	100	0	100	0
Incisal	Average	94.38	5.62	89.54	10.46
	Std dev.	3.18	-	3.72	-
	Best program	98.80	1.20	92.59	7.41
Sella	Average	55.53	44.47	43.06	56.94
	Std dev.	13.23	-	14.27	-
	Best program	73.17	26.83	62.96	37.04

Table 5.1: Detection results for training and test sets based on the use of features calculated using handcrafted shapes and the highest output for predicting the position of the landmark. Results are based on a training set of 83 images and a random set of 27 test images that are independent of the training set. The averages are calculated from the best individual from each run for 80 evolutionary runs.

The best performing programs from 80 evolutionary runs are shown in Figure 5.4. The landmark positions were predicted by applying the relevant program from Figure 5.4 to the images in Figure 5.5. The cross in each image of Figure 5.5 corresponds to the predicted position of the landmark.

```
(- (- M3 (- M2 S1)) S4)
```

Bottom corner of the ruler detection program

```
(* (/ (- (- (- M2 S2) (* (/ (- (- M2 S2) S3) M1) (- M5 M4))) S3) M1) (- M5 M4))
```

Nose detection program

```
(* (* (* (+ M8 (* M7 S6)) (/ M6 (+ M8 S5))) (- 162.108 M1)) (* (* (/ (* S4 M8) (* M8 M5)) (/ S4 (+ M5 M5))) (- 162.108 M1)))
```

Incisal upper incisor detection program

```
(/ (- M5 (/ M1 S5)) (+ (+ M3 S4) (+ (+ (+ S3 (/ 59.5795 S2)) M3) (/ M1 S2))))
```

Menton detection program

```
(/ (- (- S2 (- (* (- M7 M1) 113.67) (- (/ (/ M3 226.489) (+ M7 S1)) M4))) (- (* (- M3 M1) 113.67) (- (- (- (* S6 S6) S3) M6) S2))) (* M3 (* (+ (/ S7 (+ (/ M3 226.489) M6)) (+ (* S4 S6) M4)) (* S1 (+ (+ (* S4 S4) M2) (* (/ M7 (+ M7 S1)) M7))))))
```

Sella detection program

Figure 5.4: These programs are the best individuals from 80 evolutionary runs for the bottom corner of the ruler, nose, incisal upper incisor and sella landmarks.

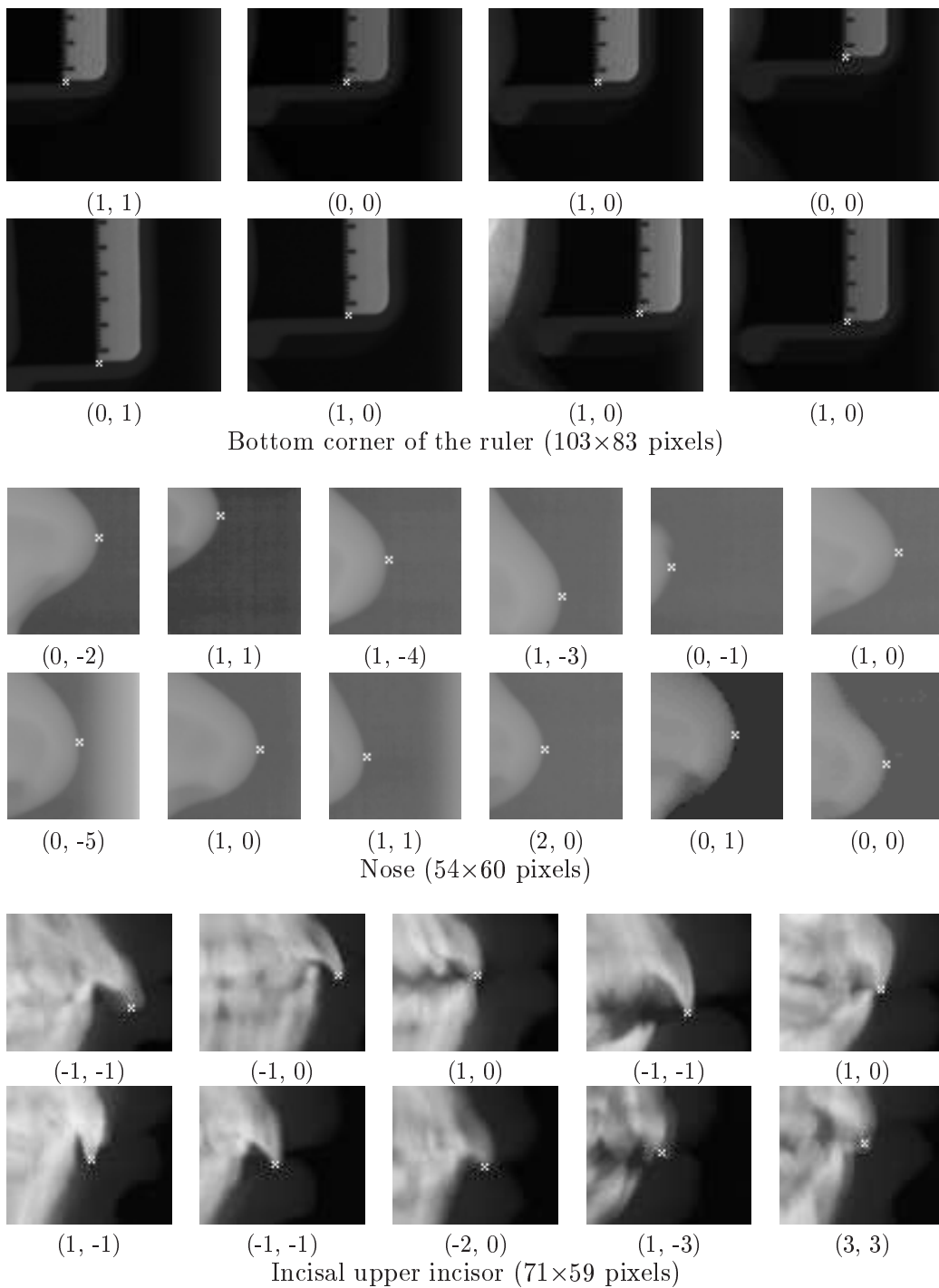


Figure 5.5: A selection of images showing the predicted position, illustrated by the cross, for five different landmark types. The different landmark types from the top to bottom rows are the bottom edge of the ruler, nose, incisal upper incisor, menton and sella landmarks. The error shown under each image is a measure of the predicted position relative to the actual position. Positional error is measured in pixels.

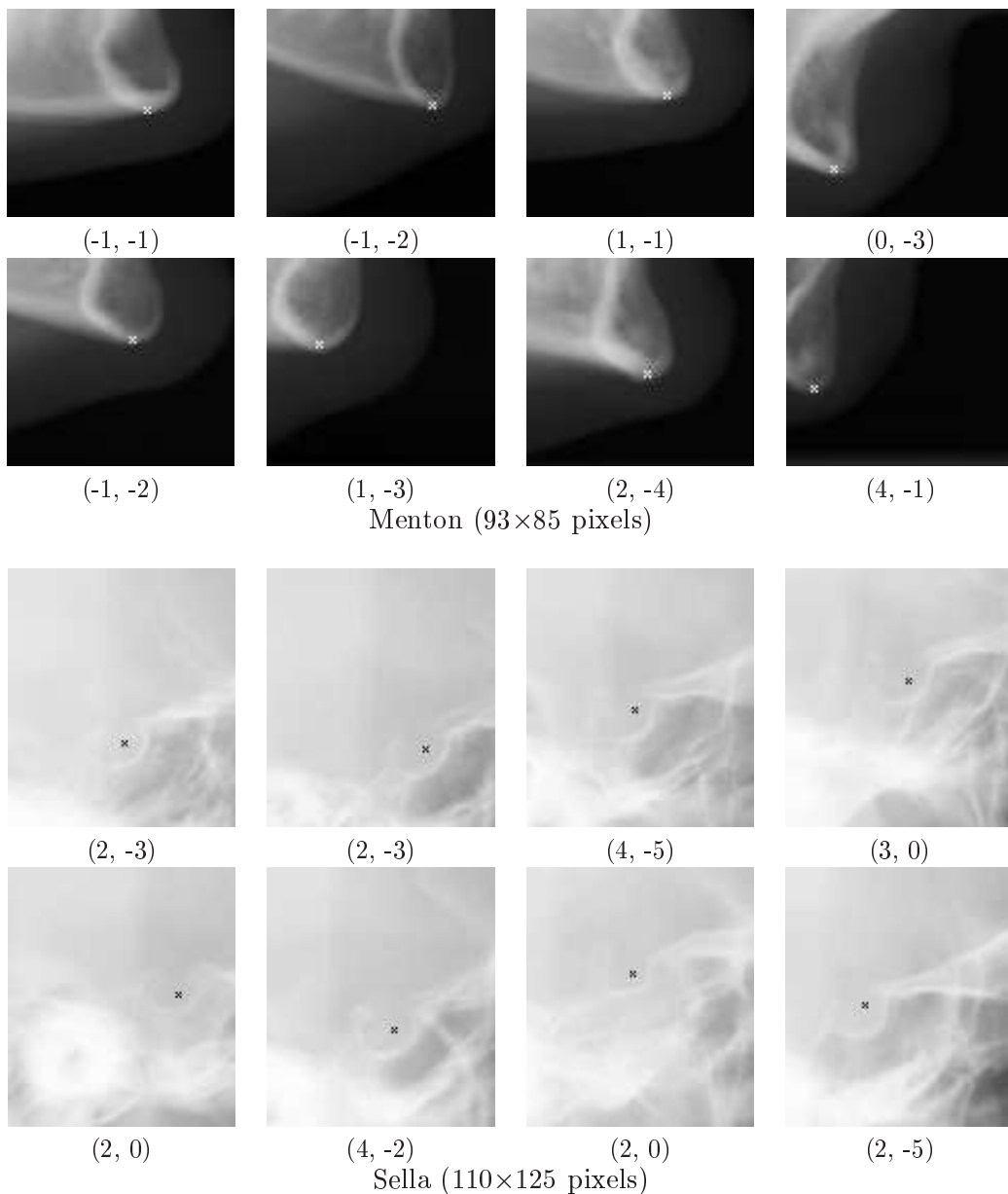


Figure 5.5 (continued)

The fitness graphs shown in Figure 5.6 illustrate the process whereby the fitness function is minimised, i.e. Equation 5.1, with the ultimate goal of achieving a fitness score of zero. As the fitness score becomes smaller, programs with a better detection performance and lower false alarm rate are produced. The evolutionary process continues until either a fitness score of zero, i.e. 100% detection rate, or the number of generations has reached 100. Detection of the bottom corner of the ruler and nose landmark achieved a 100% detection rate from

100% (80/80) and 61% (49/80) of runs respectively whilst the incisal upper incisor and sella landmarks never achieved a detection rate of 100%.

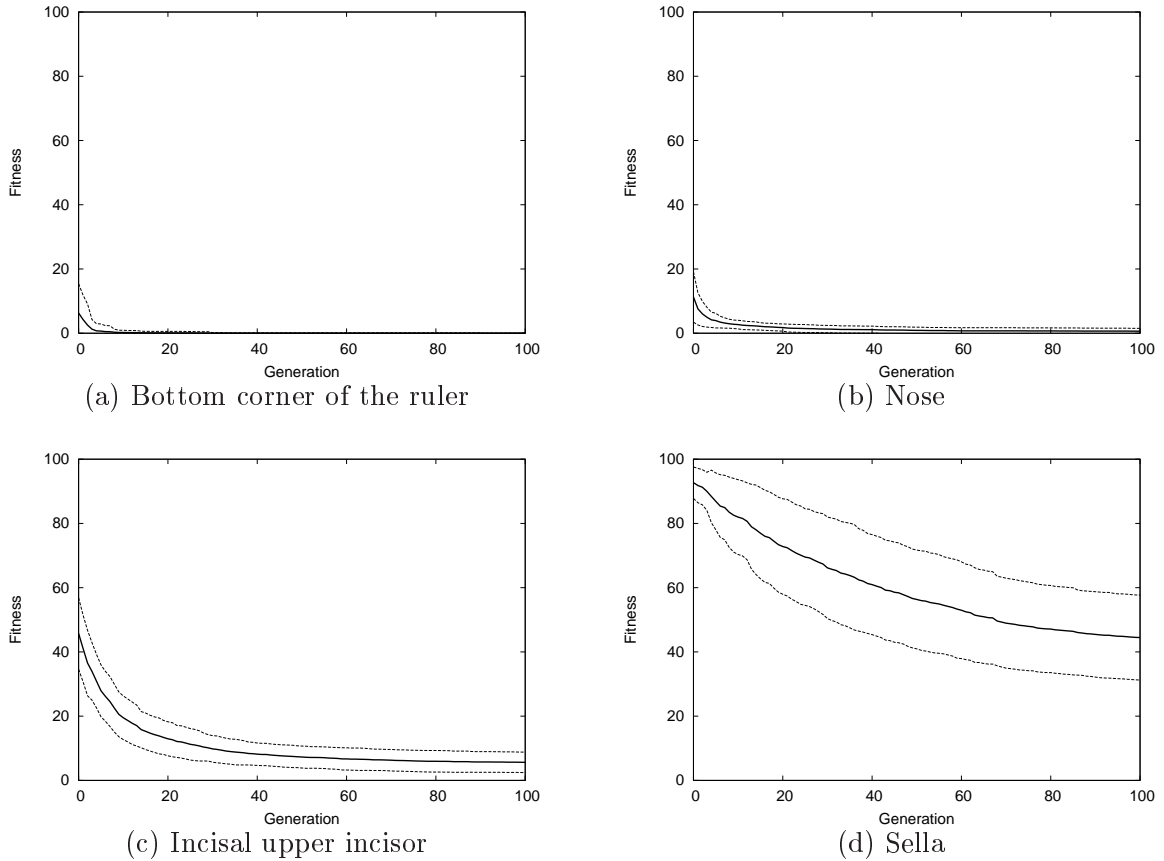


Figure 5.6: The fitness graphs are the average fitness scores of the best evolved programs that have been applied to every pixel position in an image. The average fitness score is calculated from the fitness score of the best individual at each generation for 80 evolutionary runs. The dotted lines show \pm one standard deviation from the average fitness.

Langdon et al. in [80] presented a graph for visualising the performance of programs applied to both training and test sets. The graph was used to indicate how much over fitting had occurred on training data. The graphs in Figure 5.7 determine how well the method has generalised on the training set by comparing the best performing individual from each evolutionary run and measuring the performance against the test set. The diagonal line represents neutral situations where the performance of detection programs are equal when applied to both the training and test sets. Points below the diagonal line indicate that detection programs perform better on training data.

The performance of detection programs for the bottom corner of the ruler and nose landmarks indicate minimal over training as the points in the scatter plot are evenly distributed about the diagonal line. However, the performance of detection programs for the incisal upper incisor and sella landmarks indicate over fitting as shown by the majority of points located below the diagonal line. The scatter below the diagonal line indicates a bias for performing better on the training data. The detection programs for the incisal upper incisor and sella landmarks on average perform 5% and 12% better on training than test data. The discrete steps in the scatter plot are due to the finite number of examples in the training and test sets.

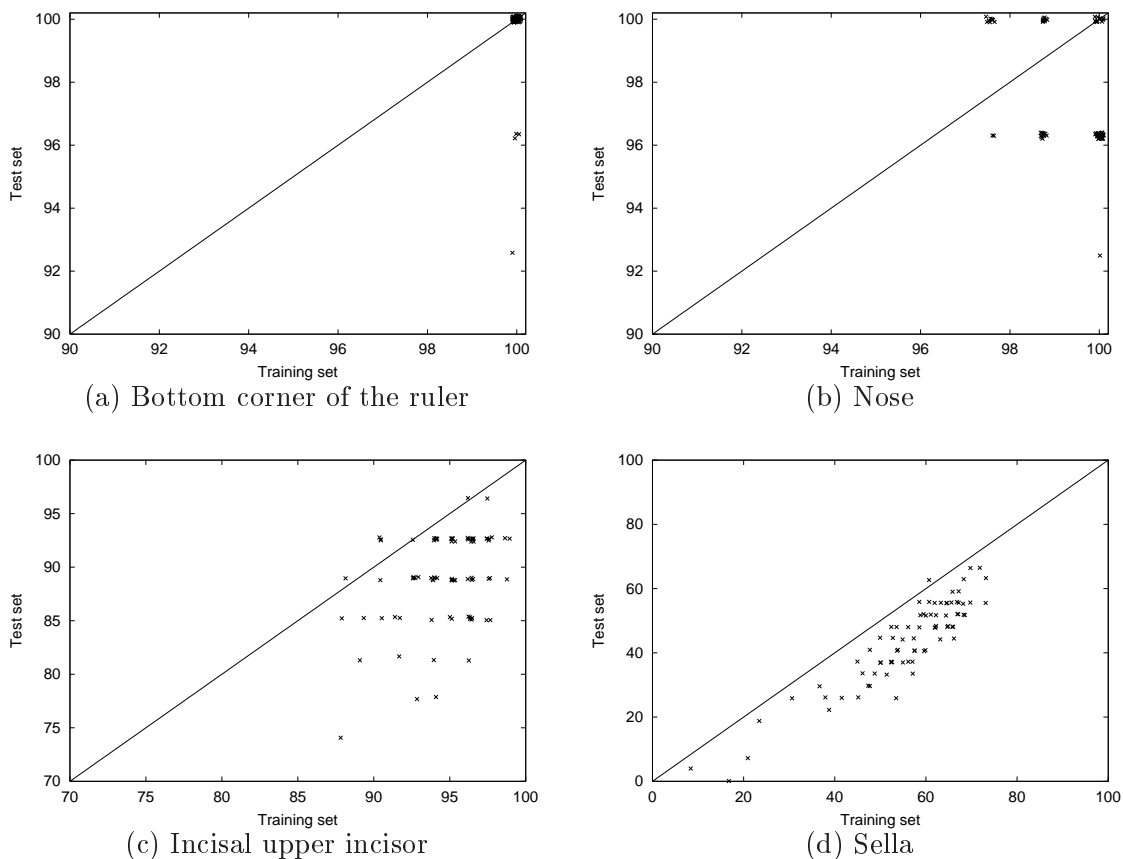


Figure 5.7: Detection rate (%) of the best program from 80 evolutionary runs applied to training and test data. Programs were evolved from features calculated using handcrafted shapes and the highest output was used for predicting the position of the landmark. The data has been jittered for the purpose of enhancing clarity. Data points located below the diagonal line indicate over fitting of data during training.

As a general rule, the size of the programs increased with the difficulty of the landmark.

This is evident by the programs that are shown for each landmark in Figure 5.4. It is not conclusive whether the additional nodes are required to improve the fitness score, or if the increase in program size is a result of introns being introduced during an evolutionary run. Introns were described in Section 2.5.4 in reference to the growth of code not contributing to a program’s performance. However, we can hypothesise based on the work of Zhang et al. in [164] that the size of the program has increased because of the problem complexity.

A drawback of the method is that run times of the evolutionary process are high with one run of 100 generations taking around $10.8 \text{ hrs} \pm 2.7 \text{ hrs}$ ¹. However, the evolutionary process is a once-only cost and applying the program to an image is very fast, taking around 0.15 seconds per image². Given the coarseness of the features used, particularly for the nose tip and incisor points, the detection accuracy achieved is surprising and suggests that with more attention to the features the approach will be successful on the more difficult landmarks.

5.4.2 Highest Output: Evaluating every second pixel position

5.4.2.1 Motivation

The main factors that influence training times during the evolutionary process are:

- The number of images in the training set
- The maximum depth of the tree
- The number of positions to be evaluated in the image

¹Processing time is calculated by averaging the time to process 100 generations for 80 evolutionary runs. An evolutionary run is based on evolving a detection program for the sella landmark. Processing was performed on an Intel Pentium 4 1.4 GHz CPU.

²Processing time is based on the time to predict the sella landmark using the highest output. Processing was performed on an Intel Pentium 4 1.4 GHz CPU.

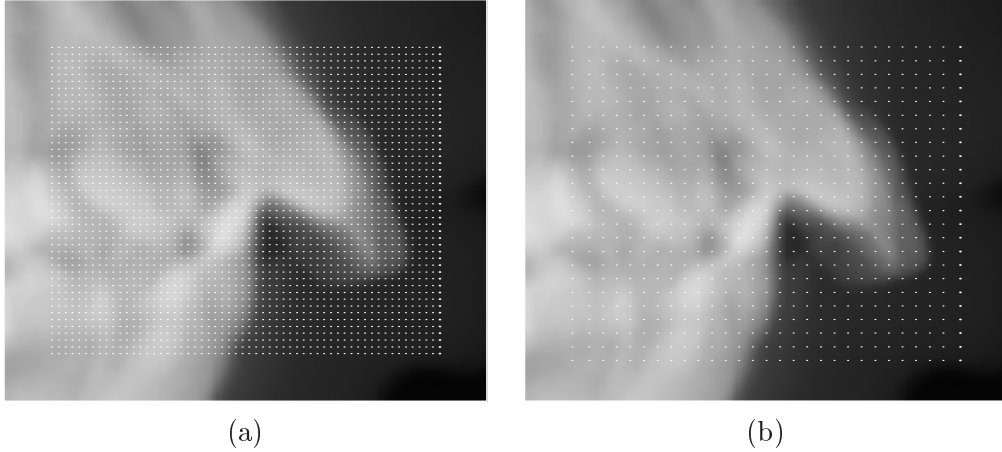


Figure 5.8: The white dots correspond to positions that are evaluated in the image. Images (a) and (b) are evaluated at each pixel position and every second pixel position respectively.

The previous section demonstrated that genetic programming worked relatively well at predicting the position of the landmark, however, the processing time of an evolutionary run is quite long. The aim of this section is to investigate if the number of pixel positions evaluated during training can be reduced without compromising the performance of detection programs. It is expected that reducing the number of evaluations will reduce training time. Figures 5.8(a) and 5.8(b) depict two images where each pixel is evaluated and every second pixel is evaluated respectively. If training is not compromised by evaluating every second pixel, then it is expected that training time will be reduced by a factor of four. A two-sample t test can be used to compare the differences in mean detection rate. Let the null hypothesis be that the detection rate of programs evaluated at each pixel position have the same mean as programs evaluated at every second pixel position. In this experiment, the hypothesis is tested on three landmarks of varying difficulty ranging from easy (nose) to difficult (sella).

$$H_0 : \mu_{\text{all}} = \mu_{\text{qtr}}$$

$$H_1 : \mu_{\text{all}} \neq \mu_{\text{qtr}}$$

$$p\text{-value} \begin{cases} \leq \alpha & \text{reject } H_0 \text{ in favour of } H_1; \\ > \alpha & \text{do not reject } H_0. \end{cases}$$

		Training		Testing	
		All	Quarter	All	Quarter
Nose	Detection rate(%)				
	Average	99.36	99.24	97.69	97.55
	Std dev.	0.89	1.05	1.90	1.95
	<i>p</i> -value	0.429		0.648	
Incisal	Detection rate(%)				
	Average	94.38	94.61	89.54	88.98
	Std dev.	3.18	2.63	3.72	4.25
	<i>p</i> -value	0.625		0.380	
Sella	Detection rate(%)				
	Average	55.53	54.73	43.06	44.40
	Std dev.	13.23	12.84	14.27	15.27
	<i>p</i> -value	0.696		0.566	

Table 5.2: An investigation to determine if the detection performance of programs is compromised by reducing the number of evaluations on training data. The table shows a comparison of detection performance for programs that have been trained on all pixel positions (All) and every second pixel position (Quarter) for the nose, incisal upper incisor and sella landmarks. The average detection rate is calculated from the detection rate of the best individual from each run for 80 evolutionary runs. The *p*-value is calculated from a two-sample *t* test to compare the mean detection rate from two independent samples.

The *p*-value exceeds an alpha level, or critical ‘cut-off’ boundary, of 0.05, for each landmark and so we accept the null hypothesis (refer to Table 5.2). This indicates that there is not enough evidence to conclude that the detection performance of programs, at the end of the evolutionary process, has changed by reducing the number of evaluations during training. Additionally, the null hypothesis is also accepted as the *p*-value exceeds an alpha level of 0.05 when comparing the detection performance of programs against test data. This indicates that there is no evidence to support the hypothesis that the detection performance of programs has been altered by reducing the data by a factor of four. Therefore, to reduce the time of an evolutionary run, subsequent experiments will be based on evaluating a program at every second pixel position in an image. Figure 5.9 depicts fitness graphs for each landmark which are similar to those shown in Figure 5.6.

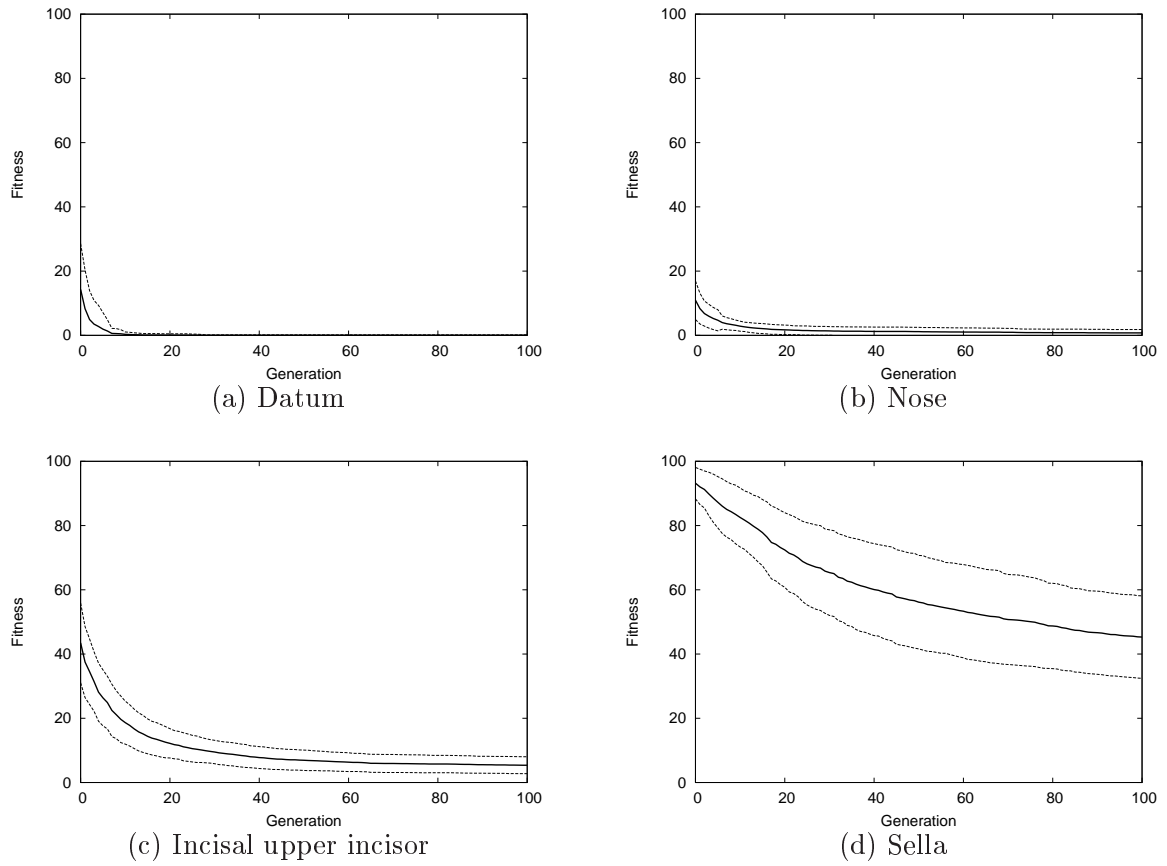


Figure 5.9: The fitness graphs are the average fitness scores of the best evolved programs that have been applied to every second position in an image. The average fitness is calculated from the fitness score of the best individual at each generation for 80 evolutionary runs. The dotted line is \pm one standard deviation from the average fitness.

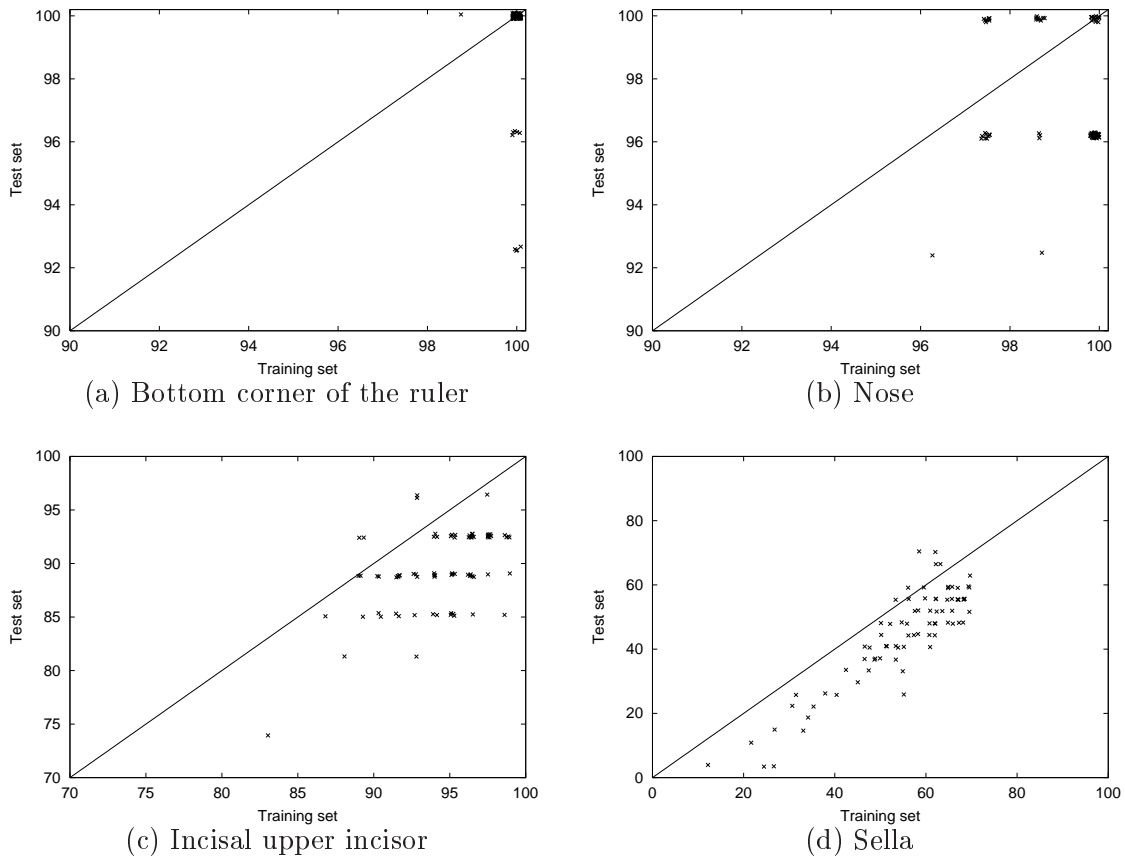


Figure 5.10: Detection rate of the best program from 80 evolutionary runs. An evolutionary run is based on applying a program to every second pixel position in the training data. Programs were evolved from features calculated using handcrafted shapes and the highest output was used for predicting the position of the landmark. The data has been jittered for the purpose of enhancing clarity. Data points located below the diagonal line indicate over fitting of training data.

5.4.3 Binary Output

5.4.3.1 Motivation

Previous work in this chapter has investigated an alternative *domain dependent* approach for evaluating fitness. The fitness is evaluated by applying a program as a moving window across an image and computing the output at each pixel location. The highest output from the image is used for predicting the position of the landmark. This process is based on the premise that only one object of interest is always located in the image. An alternative is the *domain independent* approach described in Chapter 4 that applies a program to the image

and computes the output as described above, however, in this case each location is labelled either object or background depending on the whether the program's output is positive or otherwise. The labelled positions are described as *binary outputs* in the detection map. Refer to Section 4.3.3 for additional information describing how this approach is used for locating objects.

The objective of this section is to compare the detection and false alarm rates of programs that have been evolved using the domain independent and domain dependent approach to evaluate fitness. In both cases the evolutionary process will use features calculated from the handcrafted shapes as depicted in Figure 5.2. The fitness of a program using the domain independent approach is evaluated as per Section 4.3.3. The method for evaluating fitness for the domain dependent approach is outlined in Section 5.4.1. Advantages and disadvantages will be given for both approaches.

5.4.3.2 Results

Results presented in Table 5.3 are for the nose and incisal upper incisor landmarks. The detection rate for locating the nose landmark, a simple detection problem, was similar using either the domain dependent approach (highest output) or the domain independent approach (binary output). The detection rate for the incisal upper incisor, a medium difficulty problem, using the domain dependent approach was slightly lower compared to the domain independent approach. However, the false alarm rate for both landmarks using the domain independent approach was significantly higher compared to the domain dependent approach approach. An advantage of using the highest output for predicting the position of cephalometric landmarks is that fewer false alarms are produced.

		Training set		Testing set	
		Highest	Binary	Highest	Binary
Nose	Average detection(%)	99.24	98.54	97.55	99.17
	<i>p</i> -value	0.000		0.000	
	false alarm(%)	0.76	47.33	2.45	33.75
	Best program detection(%)	100	100	100	100
	false alarm(%)	0	3.66	0	0
Incisal	Average detection(%)	94.61	97.38	88.98	94.58
	<i>p</i> -value	0.000		0.000	
	false alarm(%)	5.39	107.50	11.02	94.17
	Best program detection(%)	97.59	98.80	96.30	100
	false alarm(%)	2.41	72.29	3.70	51.85

Table 5.3: Comparison of detection programs that were evolved to predict the position of the landmark using the highest output and binary output. The averages are calculated from the best individual from each run for 80 evolutionary runs. The *p*-value is calculated from a two-sample *t* test to compare the mean detection rate from two independent samples.

5.4.3.3 Discussion

The primary objective of this work as described in Section 1.1 is to have an automated approach for locating cephalometric landmarks. However, it is expected landmarks not found will need to be manually located, while false alarms will require a practitioner to check the X-ray and re-position landmarks. The results from the previous section demonstrated the domain dependent approach produced fewer false alarms compared to the domain independent approach. However, the domain dependent approach makes only one prediction resulting in a decrease in detection rate.

Future work for increasing detection rate and reducing false alarm rate would be to apply a two stage approach for object detection. Multiple stage approaches have been applied previously by [58, 118, 141] for locating objects in large images. The first stage would be to train a classification program, similar to the method described as the domain independent approach, with the aim of minimising the number of candidate positions in the image. The second stage would then train a detection program that would only be applied to the candidate positions. The highest output from the detection program, as used in the domain dependent approach, would be used to predict the position of the landmark.

Because the complexity of the problem has been reduced using a two stage approach, it is expected that detection programs will become easier to understand. Currently programs are quite large and program complexity increases with the level of detection difficulty.

```

if Output_1 < 0 then
  background
else
  if Output_2 > highest
    record position(x, y)

```

Algorithm 1: Description of the two stage approach applied to object detection. The first stage applies a classification program to classify positions based on the output, `Output_1`. The second stage applies a detection program to each of the candidate positions, i.e. when `Output_1` is greater than zero, and the output is computed, `Output_2`. The highest output from the detection program is used to predict the position of the landmark.

5.4.4 Highest Output: Uncertain region

5.4.4.1 Motivation

It was demonstrated that a method using a domain dependent approach for locating landmarks is able to outperform the domain independent approach in terms of producing fewer false alarms. The domain dependent approach described in Section 5.4.1 uses the highest output to predict the most likely position of the landmark. However, the highest output does not always correctly predict the location of the landmark and the false alarm rate increases with image complexity. This poses the question: “Are we able to refine the domain dependent approach with the objective of reducing the false alarm rate?”

Previous work in this chapter has shown some landmarks cannot be located accurately enough for a cephalometric analysis because the images are either ambiguous in nature or located within a cluttered background. The detection performance for locating landmarks within a cluttered scene is most likely improved by making available better terminal and function sets, however, improving the detection performance of landmarks within areas of ambiguity we feel is related to how fitness is evaluated. An example of ambiguity is finding the position of the upper lip landmark that may also encompass the lower lip within the same image as shown in Figure 5.11. In this example, both the upper and lower lips are similar in

appearance.

Figure 5.11 is an image showing the upper and lower lips with the accompanying output of a reasonably successful detection program that has been evolved using the domain dependent approach outlined in Section 5.4.1. The output of the program, which is represented as the surface plot, has been superimposed on the greyscale image; each point on the wire mesh is the output of the program at a given pixel location. The objective of this program was to detect the tip of the upper lip. Figure 5.4.1 illustrates that the highest output from the detection program – refer to the left side of View A – coincides with the position of the lower lip. In this particular example, the landmark was incorrectly located and is therefore a false alarm. The second highest output – refer to the right side of View A – coincides with the correct location of the upper lip. The reason for this occurring is that the lower lip exhibits similar characteristics to the upper lip. So based on this result, are we able to reformulate the domain dependent approach using the highest output for locating landmarks so that images containing ambiguity are not classified?

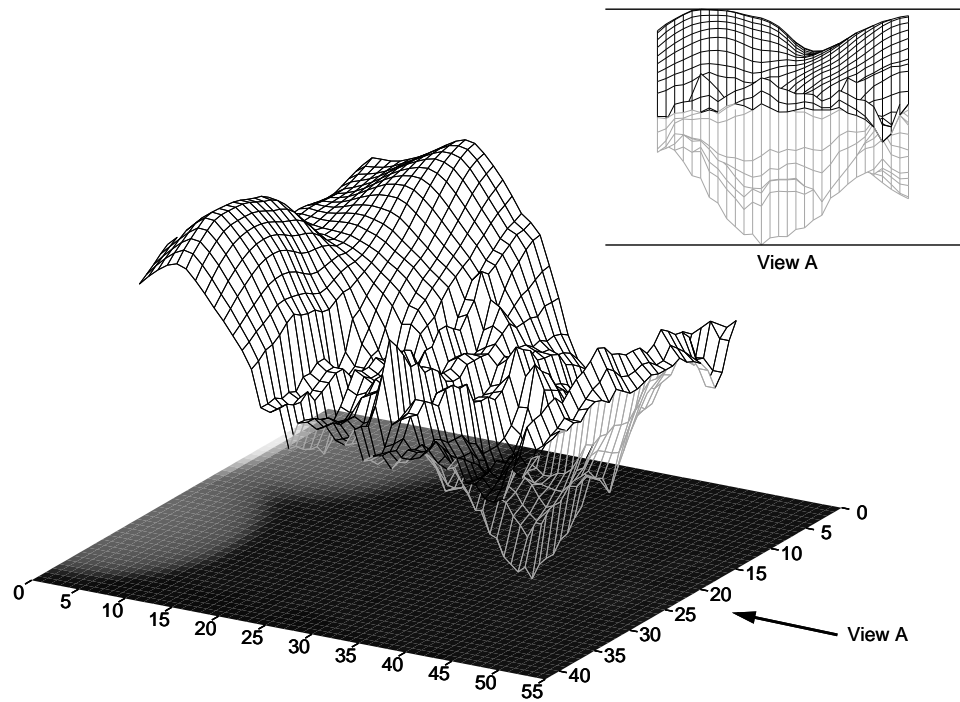


Figure 5.11: An ambiguous image that contains the desired object, the upper lip, but also contains the lower lip that is similar in appearance. The surface plot represents the output from a detection program that has been applied to the image. The highest output is located at the lower lip at position (11, 31) and the region containing the second highest output is located at the upper lip at position (20, 7). The domain dependent approach uses the highest output to predict the position of the landmark and as result the landmark will be recorded as a false alarm.

As described in Section 2.5.2.6, a common approach to using genetic programming for object detection problems has been to formulate as an object/non-object classification problem. This is the most simplistic classification model as the output of the program decides the class and often zero has been chosen as the decision boundary. An exception to this is research presented by Teredesai et al. [144] who proposed an undefined region that was used when the detector is unable to confidently make either a positive or negative decision – this was discussed in Section 2.5.2.6. However, our domain dependent approach uses the highest output for predicting the location of the landmark and we have found that this significantly reduces the number of false alarms when compared to classifying as object/non-object as per the domain independent approach. So rather than defining an arbitrary region between two classes, Teredesai et al.’s concept has been reformulated so a landmark will not be classified

when the output from a detector is within a percentage of the highest output. This is known as the *uncertain region*.

The fitness of a program during training is calculated by using detection and false alarm rates. The fitness is evaluated as follows:

1. A program is traversed across a training image and the program's output, P_{ij} , at position (i, j) is evaluated. The predicted position of the detected landmark is recorded as the location corresponding to the highest output, $P_{Highest}$.
2. A second traversal of the image verifies that the program's output at each location in the image, P_{ij} , is not similar to the predicted position, $P_{Highest}$. Each position is verified using Equation 5.4 which determines if a position has produced an output within the shaded area as shown in Figure 5.12(a). The size of the shaded area is predetermined prior to training and represents the *uncertain region* which is expressed as a percentage between the lowest, P_{Lowest} , and highest outputs, $P_{Highest}$. For example, a threshold of 10 requires the program's output at each position within the image, other than the predicted position, to be at least 10% smaller than the highest output. If the criterion of Equation 5.4 has been fulfilled then the landmark for that image is recorded as *unclassified*, i.e. the landmark's position will not be predicted for the image. Pixels located within a distance of 5 pixels (2 mm) of the predicted position are not verified because they are within an error tolerance that is acceptable for a cephalometric analysis. It is also expected that an output at these pixel locations will more than likely produce an output similar to the highest output.

$$\frac{P_{Highest} - P_{ij}}{P_{Highest} - P_{Lowest}} \times 100\% \leq Threshold \quad (5.4)$$

3. If the landmark in the image has not been *unclassified*, then a comparison is made between the landmark's predicted position and the known true location. A match, true positive, occurs when the comparison is within a tolerance of 5 pixels or 2 mm. If the comparison is not within the tolerance then the landmark for the respective image is recorded as a false alarm. The tolerance is an upper error limit that is deemed

acceptable for a cephalometric analysis. The error is defined as the Euclidean distance between the position found by the ‘system’ and an ‘expert’.

4. At the conclusion of evaluating the program for each image in the training set the detection rate, DR , and false alarm rate, FAR , are calculated.
5. The fitness is computed as per Equation 5.5.

$$fitness = A \times FAR + B (1 - DR) \quad (5.5)$$

where A and B are constant values of 50 and 1000 respectively that are also used in Section 4.3.3. The fitness function defined in Equation 5.5 is constructed so as detection rate increases and false alarm rate decreases the fitness score will approach zero.

The ultimate aim is to correctly detect the position of the landmark, however, if the landmark cannot be confidently located then ideally we would like the program not to predict the position of the landmark as opposed to producing a false alarm. The fitness function from Equation 5.1 is not used for this scenario because false alarms and unclassified landmarks would be equally awarded. The objective of the fitness function in Equation 5.5 is to reward programs that can detect landmarks and produce a small number of false alarms, and also indirectly reward programs by not locating landmarks that are within cluttered or ambiguous scenes.

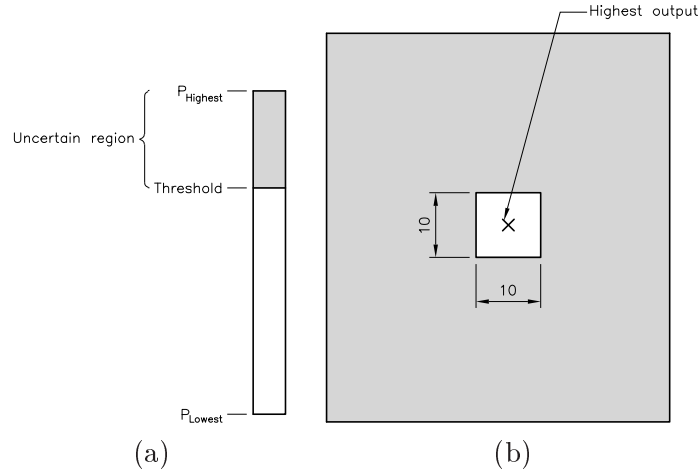


Figure 5.12: The shaded area in (a) is the *uncertain region* and is defined as a percentage between the lowest, P_{Lowest} , and highest output, $P_{Highest}$. The figure in (b) represents an image where ‘x’ represents the position of the highest output. A comparison is performed on the shaded area to determine if any outputs are within the *uncertain region*. The shaded area in (b) is bound between an error tolerance of 5 pixels around the highest output and the edge of the image.

5.4.4.2 Results

To determine the effect of the threshold on false alarm rate, the method has been tested on the upper lip and sella landmarks. The reason for selecting these landmarks is because the performance of the detection programs was not adequate. The upper lip landmark is a medium level of detectability based on the ambiguity between the upper and lower lips. The sella landmark is a more difficult landmark which is located within a cluttered background. The results for each threshold setting are based on 80 evolutionary runs.

$$\Delta rate = DR - FAR \quad (5.6)$$

The graphs in Figure 5.13 show that incorporating a threshold during training produces programs that have minimal effect on false alarm rate when detecting the upper lip, while the results for the sella landmarks show false alarm rate is significantly reduced plateauing at a threshold value of 0.1. However, the smaller graph inset of each graph shows an undesirable trend of detection rate reducing at a faster rate than false alarm rate. This is described as $\Delta rate$ and is defined in Equation 5.6. This indicates that while false alarm rate is reduced it

is to the detriment of detection rate. In other words, detection rate reduces at a faster rate than false alarm rate.

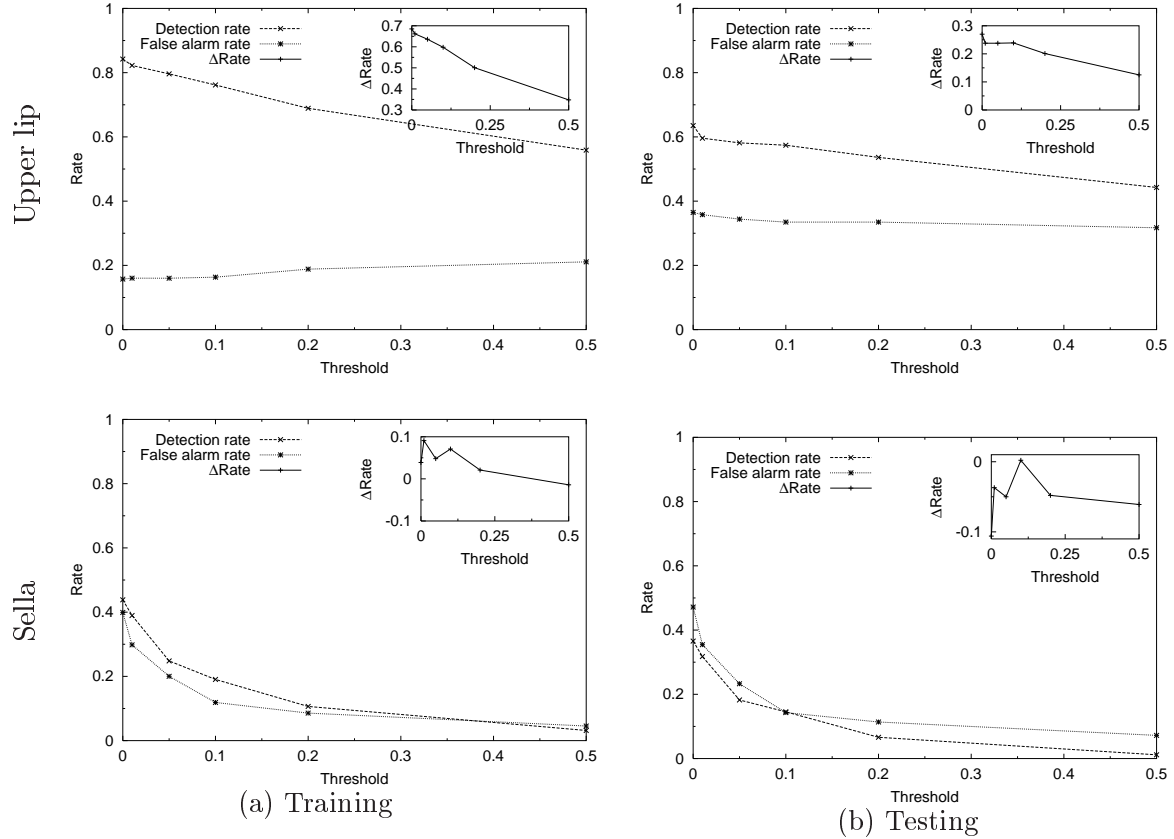


Figure 5.13: The graphs illustrate how the *uncertain region* influences false alarm rate and detection rate. The top and bottom graphs are results for the upper lip and sella landmarks respectively. The graph inset is the difference between detection and false alarm rates, $\Delta Rate$, at each threshold. The results are based on an average which is calculated by averaging the best individual at the end of 80 evolutionary runs.

5.4.5 Highest output: Minimum Distance Error

5.4.5.1 Motivation

The approaches to evaluating fitness for our detection problem, that have been described so far, use a combination of detection and false alarm rates. The detection rate is a measure of how well the programs predict the landmarks within 2 mm of the desired location, however, the accuracy of a detection program beyond the 2 mm tolerance was not rewarded. An

alternative to evaluating fitness is a fitness metric that rewards a program that can accurately locate landmarks by minimising the *cumulative distance error*. The ultimate objective for our problem is to locate landmarks within 2 mm of the desired location. So how well will a fitness metric that minimises the distance error predict the location of landmarks and how will it compare with the fitness measure described as the domain dependent approach in Section 5.4.1?

The approach described in this section uses an alternative measure to evaluate fitness that rewards programs on the basis of minimising cumulative distance error, ϵ , when applied to images in the training set. The distance error is the Euclidean difference between the landmark's true location and the predicted position. The fitness is calculated as follows:

1. The program is applied as a moving window across a training image and the program output, *Output*, is evaluated at each pixel location. The output of the genetic program, *Output*, is a floating point number which is interpreted as the likelihood that the evaluated position from the image is a landmark centre or background. During training the highest value of *Output* from each image is used as the predicted position of the landmark. The predicted position, (x_i, y_i) , given by the genetic program is then compared with the known true location, (X_i, Y_i) , and the error is calculated.
2. The performance of a program is measured by iteratively applying the first step to each image in the training set and calculating the distance error.
3. The fitness is computed as per equation 5.8.

$$fitness = \epsilon \tag{5.7}$$

$$\text{where } \epsilon = \sum_{i=0}^n \sqrt{(X_i - x_i)^2 + (Y_i - y_i)^2}$$

and n is the number of images in the training set

5.4.5.2 Results

Two two-sample t tests are used to determine if there is a difference in mean detection rate and detection accuracy between two samples of programs using: (a) cumulative distance error as a measure of fitness and (b) detection rate as a fitness measure. Let the null hypothesis be that the two samples are from the same population.

The p -values for comparing the detection rate of programs applied to training images are less than an alpha of 0.05 for the nose, incisal upper incisor and sella landmarks. Therefore, the null hypothesis, that the detection rate of programs using the cumulative distance error is the same, can be rejected. However, the average detection rates for the nose and incisal upper incisor were only 0.60% and 2.62% less than the domain dependent approach described in Section 5.4.1. The performance of the sella landmark was worst with the average detection rate decreasing by 36.45%.

The p -values for the detection accuracy of programs applied to training images are less than 0.05 for the bottom corner of ruler, nose and incisal upper incisor landmarks. Therefore, the null hypothesis, that the detection accuracy of programs using the cumulative distance error is the same, can be rejected. A comparison of the average detection accuracy between the two approaches in Table 5.4 indicates a significant improvement in average detection accuracy for the bottom corner of the ruler, nose and incisal upper incisor landmarks. However, there was not enough evidence to suggest that the detection accuracy for the sella landmark had changed.

These results suggest that using detection rate as a fitness measure is advantageous because the function is better suited to a wider range of landmark detection difficulties. However, the cumulative distance error as a fitness measure performs extremely well at precisely locating easier types of landmarks. It is recommended as future work that these two fitness functions should be combined as a multiple objective problem. It is expected that detection programs will have high detection performance and also improved accuracy.

		Training		Testing	
		ϵ	$(1 - Dr)$	ϵ	$(1 - Dr)$
Bottom corner of ruler	Average				
	detection (%)	99.97	99.98	100	99.21
	false alarm(%)	0.03	0.02	0	0.79
	accuracy (pixels)	0.86	2.41	0.83	2.45
	<i>p</i> -value				
	detection (%)		0.563		0.005
	accuracy (pixels)		0.000		0.000
	Best program				
	detection (%)	100	100	100	100
false alarm(%)	0	0	0	0	
accuracy (pixels)	0.81	0.97	0.75	1.14	
Nose	Average				
	detection (%)	98.64	99.24	97.69	97.55
	false alarm(%)	1.36	0.76	2.31	2.45
	accuracy (pixels)	1.85	2.39	2.21	2.67
	<i>p</i> -value				
	detection (%)		0.003		0.736
	accuracy (pixels)		0.000		0.001
	Best program				
	detection (%)	100	100	100	100
false alarm(%)	0	0	0	0	
accuracy (pixels)	1.51	1.69	1.46	1.63	
Incisal	Average				
	detection (%)	91.99	94.61	88.24	88.98
	false alarm(%)	8.01	5.39	11.76	11.02
	accuracy (pixels)	2.64	3.13	3.26	3.34
	<i>p</i> -value				
	detection (%)		0.000		0.282
	accuracy (pixels)		0.000		0.511
	Best program				
	detection (%)	97.59	98.80	92.59	92.59
false alarm(%)	2.41	1.20	7.41	7.41	
accuracy (pixels)	1.93	2.42	2.42	2.43	
Sella	Average				
	detection (%)	18.28	54.73	16.48	44.40
	false alarm(%)	81.72	45.27	83.52	55.60
	accuracy (pixels)	14.13	14.93	17.25	17.58
	<i>p</i> -value				
	detection (%)		0.000		0.000
	accuracy (pixels)		0.310		0.745
	Best program				
	detection (%)	68.29	69.51	59.26	62.96
false alarm(%)	31.71	30.49	40.74	37.04	
accuracy (pixels)	6.08	9.62	9.07	13.14	

Table 5.4: A comparison of cumulative distance error and detection rate fitness functions. The averages are calculated from the best individual from each run for 80 evolutionary runs. The *p*-value is calculated from a two-sample *t* test to compare the mean detection rate and accuracy (pixels) from two independent samples.

5.5 Function Set

5.5.1 Motivation

The aim of this section is to investigate a selection of operators that are commonly used in genetic programming for solving vision and image related applications. As discussed in Section 2.5.2.5, the most common operators used in genetic programming for solving vision and image related problems are the $+$, $-$, \times and $/$ operators. A definition for each of these operators is given in Section 4.3.2. Other operators less commonly used include the *min* and *max* operators. The *min* and *max* operators return the minimum and maximum value from an arity of two respectively. A definition for each operator is given in Table 5.5.

Functions		
Function	Arity	Definition
$+$	2	$a + b$
$-$	2	$a - b$
\times	2	$a \times b$
$/$	2	$\begin{cases} \frac{a}{b} & \text{if } b \neq 0 \\ \text{INT_MAX} & \text{else} \end{cases}$
<i>max</i>	2	$\max(a, b)$
<i>min</i>	2	$\min(a, b)$

Table 5.5: Definition of operators.

The arithmetic operators in Table 5.5 allows the formation of linear and non-linear detection programs and incorporating *min* and *max* into the function set allows non-continuous detection programs to be evolved. Even though many operators can be included in the function set, a discussion of previous research by [74, 138] in Section 2.5.2.5 indicated that the inclusion of unnecessary operators or a large function set can lead to fewer successful runs or slower convergence of an evolutionary run. This is caused by additional operators increasing the size of the search space. Therefore, the aim of this section is to find a combination of operators based on the function sets defined in Table 5.6 that will, on average, lead to better performing solutions.

Case	Function Set
1	+, -
2	+, -, ×, /
3	+, -, ×, /, <i>min</i> , <i>max</i>

Table 5.6: Definition of three function sets that are made available during the genetic search.

Each function set in Table 5.6 will be investigated using the domain dependent approach that uses highest output for predicting the position of the landmark. This approach is described in Section 5.4.1.

5.5.1.1 Results

In this section we investigate the detection performance of programs that have been evolved using three different function sets. For comparing the three different function sets, a one-way ANOVA [132] will be used to measure the differences in mean detection rate. Let the null hypothesis be that the mean detection rates of programs that have been evolved from the three functions are the same. The hypothesis is tested on three different landmark types.

		Training			Testing		
		Case 1 +,-	Case 2 +,-,×,/	Case 3 +,-,×,/, min,max	Case 1 +,-	Case 2 +,-,×,/	Case 3 +,-,×,/, min,max
Nose	Average detection rate(%)	99.65	99.24	99.32	96.81	97.55	96.99
	<i>p</i> -value		0.011			0.050	
	program size	56.83	32.30	47.65	-	-	-
	Best program detection rate(%)	100	100	100	100	100	100
Incisor	Average detection rate(%)	95.09	94.61	94.78	89.07	88.98	89.86
	<i>p</i> -value		0.515			0.346	
	program size	72.20	48.00	50.38	-	-	-
	Best program detection rate(%)	97.59	98.80	97.59	92.59	92.59	96.30
Sella	Average detection rate(%)	62.81	54.73	52.81	53.89	44.40	43.15
	<i>p</i> -value		0.000			0.000	
	program size	71.67	52.00	47.23	-	-	-
	Best program detection rate(%)	68.29	69.51	74.39	62.96	62.96	62.96

Table 5.7: A comparison of average detection rate for three function sets. The average detection rate is calculated from the best individual’s detection rate from each run for 80 evolutionary runs. The *p*-value is based on a comparison of mean detection rate for the three function sets using a one-way ANOVA.

The *p*-values for comparing the detection rate of programs when applied to training images are less than an alpha value of 0.05 for the nose and sella landmarks. Therefore, the null hypothesis, of the mean detection rate of programs that were evolved from the three function sets is the same, is rejected. This indicates that at least two of the means are significantly different. Since the alternative hypothesis is supported, a Tukey’s pairwise comparison [132] is conducted to determine discrepancies between the different function sets. The pairwise comparison from Table 5.8 revealed that during training, the function set of case 1 was different from cases 2 and 3 for the nose landmark. However, the difference in mean detection rates of the three function sets shown in Table 5.7 is minimal. The Tukey’s pairwise comparison of the mean detection rate for the sella landmark showed that the function sets of cases 1 and 2 were different to case 3. The mean detection rate of the sella landmark in Table 5.7 indicates that case 1 on average will produce a program that will outperform cases 2 and 3 by 8.1% and 10.0% respectively. There was not enough evidence to suggest that any of the

three function sets had influenced the mean detection rate for the incisal upper incisor. This analysis is consistent with the average detection results of programs applied to test images.

On average, a program that was evolved using the (+, -) function set was better for the nose and sella landmarks, however, the program size using the (+, -, ×, /) operators were on average considerably smaller. It is not conclusive that the discrepancy in program sizes between the two function sets are caused by introns. No parsimony factor was used to control bloat during training.

	Training			Testing		
Nose	Case 1	2		Case 1	2	
	2	Disimilar		2	Disimilar	
	3	Similar	Similar	3	Similar	Similar
Incisal	Case 1	2		Case 1	2	
	2	Similar		2	Similar	
	3	Similar	Similar	3	Similar	Similar
Sella	Case 1	2		Case 1	2	
	2	Disimilar		2	Disimilar	
	3	Disimilar	Similar	3	Disimilar	Similar

Table 5.8: A Tukey’s pairwise comparison of the detection performance of programs evolved from three function sets.

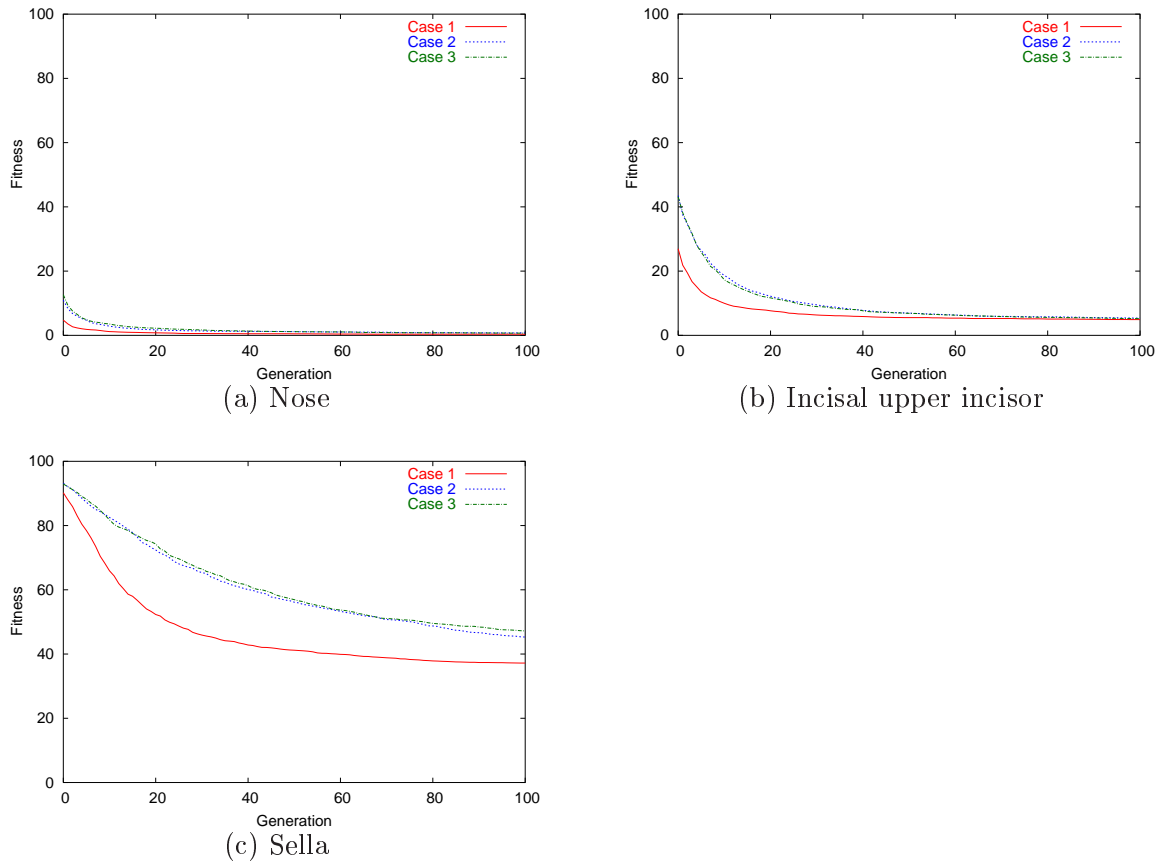


Figure 5.14: The fitness graphs are a comparison of the average fitness score for the three function sets defined in Table 5.6. The average fitness score is calculated from the fitness score of the best individual at each generation for 80 evolutionary runs.

The fitness graphs in Figure 5.14 compare the average fitness scores between the three different function sets that are defined in Table 5.6. The evolutionary process was terminated at either 100 generations or when a program achieved a 100% detection rate. All three graphs indicate that case 1, i.e. the $(+, -)$ function set, achieves an improvement in fitness score that is significantly quicker compared to the function sets of cases 2 and 3. This means that on average, a fitter solution is available sooner compared to programs that were evolved using the richer function set. It was reported in the literature by [152] that the performance may be degraded if too many operators are included in the function set as this increases the size of the search space. The $(+, -, \times, /)$ and $(+, -, \times, /, \min, \max)$ function sets produce a similar rate of convergence for optimising fitness score during training of the nose, incisal upper incisor and sella landmarks.

Training a program to detect the incisal upper incisor using the (+, -) function set demonstrated that negligible improvements were gained after generation 45, while to achieve the same fitness using the (+, -, ×, /) function set meant training for an additional 22 generations. The fitness graph for the sella landmark showed that minimal improvements were gained beyond generation 88. Although a similar fitness was achieved for the incisal upper incisor at generation 100 for the three function sets, we have speculated that if training had continued using the (+, -, ×, /) or (+, -, ×, /, min, max) function set beyond generation 100 then a similar average fitness may have occurred.

Previously we compared the mean detection rate of programs that were evolved from three different function sets. The analysis was based on 80 evolutionary runs. The outcome of the investigation was the (+, -) function set produced programs that were on average comparable to or better performing than the other two function sets. However, upon a closer inspection of the *best evolutionary runs*, i.e. the top 10% of evolutionary runs, as shown in Table 5.9, we noticed that the detection rate for programs that were evolved from the (+, -) function set was not as good as the programs evolved from the (+, -, ×, /) or (+, -, ×, /, min, max) function sets when applied to test images containing the nose landmark. There appears to be no evidence to suggest that there is a significant difference of detection performance for programs evolved from the three function sets when applied to training data. This was supported by a one-way ANOVA and a comparison of average detection rates as shown in Table 5.9.

		Training			Testing		
		Case 1 +,-	Case 2 +,-,×,/	Case 3 +,-,×,/, min,max	Case 1 +,-	Case 2 +,-,×,/	Case 3 +,-,×,/, min,max
Nose	detection rate(%)	100	100	100	96.76	100	100
	<i>p</i> -value		*			0.000	
Incisor	detection rate(%)	97.44	98.04	98.64	88.89	92.59	91.20
	<i>p</i> -value		0.000			0.067	
Sella	detection rate(%)	68.29	68.90	68.90	56.95	56.02	57.87
	<i>p</i> -value		0.731			0.832	

Table 5.9: A comparison of average detection rate from the top 10% of evolutionary runs for each function set. The average detection rate is calculated from the best individual's detection rate from each run based on the top 10% of evolutionary runs. The *p*-value is based on a comparison of mean detection rate for the three function sets using a one-way ANOVA.

	Training			Testing		
Nose		1	2		1	2
	2	Similar		2	Disimilar	
	3	Similar	Similar	3	Disimilar	Similar
Incisal		1	2		1	2
	2	Similar		2	Similar	
	3	Disimilar	Similar	3	Similar	Similar
Sella		1	2		1	2
	2	Similar		2	Similar	
	3	Similar	Similar	3	Similar	Similar

Table 5.10: An ANOVA matrix comparing the average detection performance from the top 10% of evolutionary runs for each function set.

5.5.2 Analysis of a linear function set: $\{+, -\}$

A linear program is defined as a combination of operators and terminals having an equivalent form as the linear model shown in Equation 5.8. The features of M_i and S_i correspond to the mean and standard deviation calculated from the i th shape from grey level intensities. The features and their corresponding shapes are shown in Figure 5.2 on page 85.

$$\begin{aligned}
 \text{Output} &= \alpha_1 M_1 + \beta_1 S_1 + \alpha_2 M_2 + \beta_2 S_2 + \cdots + \alpha_n M_n + \beta_n S_n + C & (5.8) \\
 &= \sum_{i=1}^n (\alpha_i M_i + \beta_i S_i) + C, \text{ where } \alpha_i \text{ and } \beta_i \text{ are integers}
 \end{aligned}$$

Our landmark detection problem has been formulated so that the highest output is used to locate the position of the landmark. This means that the constant, C , has no effect in predicting the position of the landmark and therefore the constant can be eliminated. The equation can be reduced to the linear model shown in Equation 5.9.

$$\text{Output} = \sum_{i=1}^n (\alpha_i M_i + \beta_i S_i) \quad (5.9)$$

```

(- (- (+ (- M1 S3) S5) (+ S1 (+ (+ (+ M2 (- S1 (- M1 M7))) (+ M2 S1))
(- (+ M3 (- S4 (- M1 108.475))) (- M1 108.475)))))) (+ S3 (+ (+ (+ (+ M7
(+ M2 S1)) (+ (- S4 S3) S1)) (- S1 (- M1 M7))) (+ (- S1 (- M1 M7)) S1))
(- (+ M2 (- S1 (- M1 108.475))) (- M1 108.475))))))

```

Figure 5.15: An evolved linear detection program for locating the sella landmark.

An example of a linear program evolved to locate the sella landmark is shown in Figure 5.15. This LISP S-expression can be simplified to the equivalent program shown in Equation 5.10.

$$Output = 8M_1 - 9S_1 - 4M_2 - M_3 - S_3 - 2S_4 + S_5 - 4M_7 \quad (5.10)$$

Figure 5.16 shows an analysis of the coefficients generated in 80 evolutionary runs. The magnitude of each coefficient, α and β , is calculated using the best program from each evolutionary run. The frequency that a coefficient's magnitude is either positive, negative or zero, is collated for each terminal and presented in the form of a bar chart shown in the figure. The three bar charts in Figure 5.16 show that some of the terminals have coefficients that are predominantly all positive or negative. What this indicates is that the evolutionary process has directed the search toward the systematic learning of an underlying algorithm that may be consistent between each evolutionary run. A further investigation into understanding the evolved programs is presented in Chapter 7.

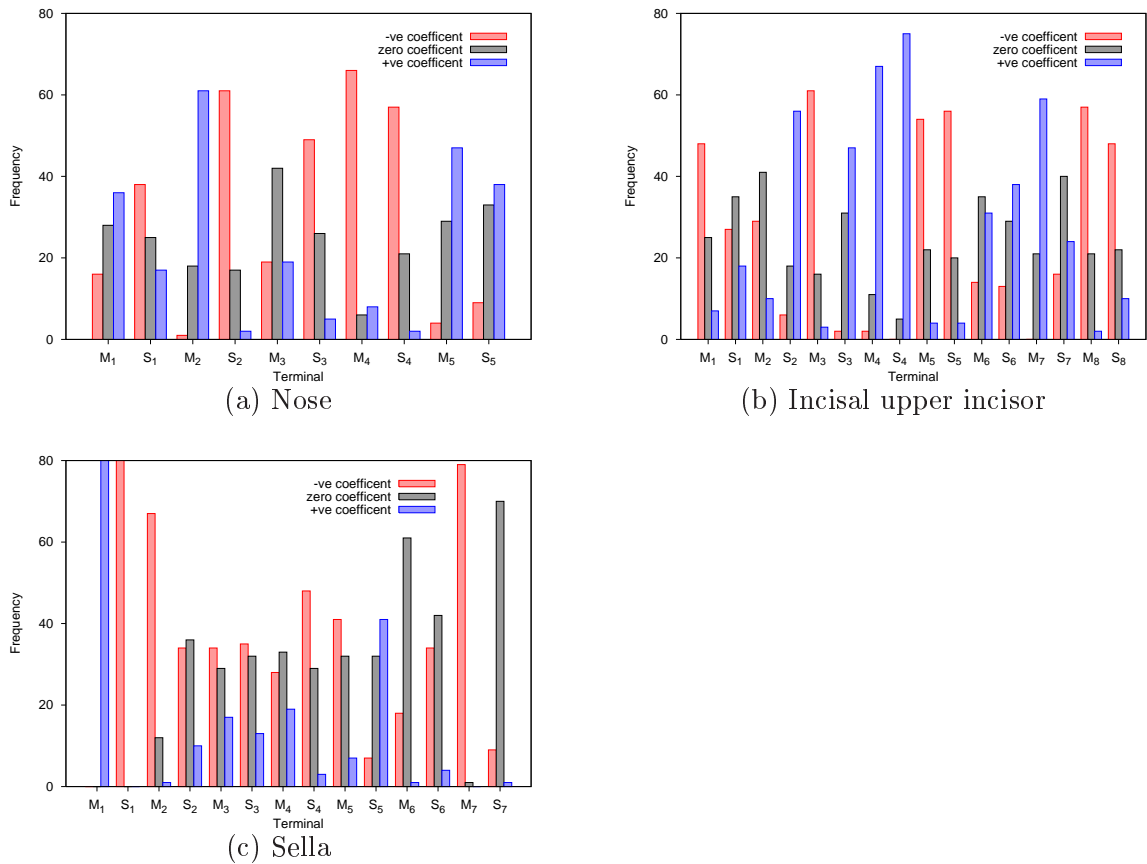


Figure 5.16: Frequency that a terminal’s coefficient, α_i or β_i in Equation 5.9, is negative, positive or zero (i.e. not used in a program). The analysis uses the best program at the end of 80 evolutionary runs.

5.6 Highest Output: Pixels as Features

5.6.1 Motivation

Previous work in this chapter has investigated several approaches for evaluating the fitness of a program using features that were calculated from handcrafted shapes. The objective was to determine a suitable approach for solving our landmark detection problem. We demonstrated a *domain dependent approach* that uses the highest output for predicting the position of the landmark. The most suitable fitness metric used detection rate as a measure of performance. The overall performance of the evolved programs, i.e. detection and false alarm rates, was preferable to the *domain independent approach* described in Chapter 4. Both methods were compared on four types of landmarks ranging from easy to hard.

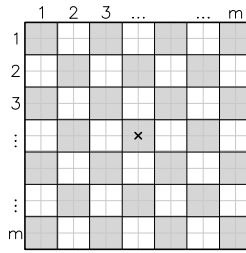


Figure 5.17: The diagram illustrates an input window of size 14×14 pixels divided into m^2 sub-regions. The average grey level intensity of the pixels within each sub-region is calculated. The averages at each sub-region represent the terminals that are available for the evolutionary process.

However, we would now like to apply our domain dependent approach that was described above and compare the performance of programs that were evolved from features using handcrafted shapes with pixels as features described in Section 4.3.1. The operators used in the genetic search are limited to the $\{+, -, \times, /\}$ function set.

5.6.2 Results

A two-sample t test is used to determine if there is a difference in mean detection rate between two samples of programs using: (a) pixels as features and (b) programs evolved using features calculated from handcrafted shapes. Let the null hypothesis be that the two samples are from the same population.

		Training		Testing	
		Pixel based	Hand-crafted	Pixel based	Hand-crafted
Bottom corner of the ruler	Average detection(%)	99.94	99.98	99.44	99.21
	false alarm(%)	0.06	0.02	0.56	0.79
	program size	33.05	24.88	-	-
	p -value		0.176		0.472
	Best program detection	100	100	100	100
	false alarm	0	0	0	0
Nose	Average detection(%)	97.62	99.24	95.79	97.55
	false alarm(%)	2.38	0.76	4.21	2.45
	program size	53.75	32.30	-	-
	p -value		0.000		0.001
	Best program detection	100	100	100	100
	false alarm	0	0	0	0
Incisal	Average detection(%)	94.28	94.61	88.06	88.98
	false alarm(%)	5.72	5.39	11.94	11.02
	program size	45.50	48.00	-	-
	p -value		0.489		0.221
	Best program detection(%)	98.80	98.80	96.30	92.59
	false alarm(%)	1.20	1.20	3.70	7.41
Sella	Average detection(%)	18.35	54.73	10.60	44.40
	false alarm(%)	81.65	45.27	89.40	55.60
	program size	58.12	52.00	-	-
	p -value		0.000		0.000
	Best program detection(%)	30.49	69.51	25.93	62.96
	false alarm(%)	69.51	30.49	74.07	37.04

Table 5.11: A comparison of detection performance for programs that were evolved from pixel based features and features using handcrafted shapes. The averages are calculated from the best individual from each run for 80 evolutionary runs. The p -value is calculated from a two-sample t test to compare the mean detection rate from two independent samples.

The p -value is less than an alpha value of 0.05 for the nose and sella landmarks. Therefore, the null hypothesis, that the mean detection rate of programs evolved from pixel based features and features calculated using handcrafted shapes is the same, can be rejected. The average detection rates for the nose and sella landmarks decreased by 1.62% and 36.38% respectively when programs were evolved using pixel based features. There was no evidence to suggest that the average detection rate for bottom corner of the ruler and incisal upper incisor

landmark had changed. Therefore, while pixel based features work as well as handcrafted features for evolving programs on two of the four landmarks, the features fail to produce a comparable detection rate on the more difficult sella landmark. This indicates that pixel based features do not perform as well as features calculated using handcrafted shapes.

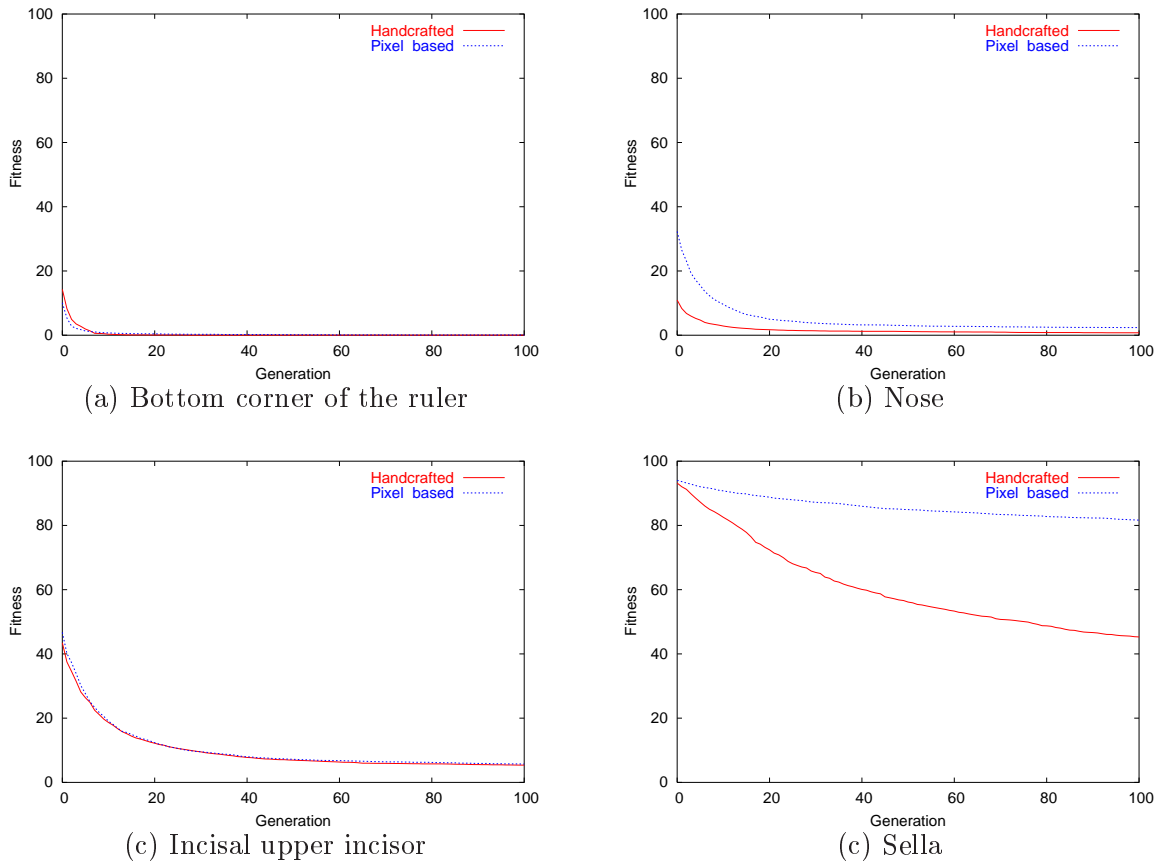


Figure 5.18: The fitness graphs are a comparison of the average fitness score for programs evolved using features calculated from handcrafted shapes and programs evolved using pixels as features. The average fitness is calculated from the fitness score of the best individual at each generation for 80 evolutionary runs.

The fitness graphs in Figure 5.18 indicate that on average, the improvement in fitness scores of programs for the nose and sella landmarks is significantly quicker when using features calculated from handcrafted shapes. The fitness of programs using pixel based features for the sella landmark at generation 100 had not improved much from the initial population. However, the fitness of programs when using handcrafted features had improved significantly from the initial population. This suggests that pixel based features do not work well on

difficult detection problems. The rate of convergence of both feature sets used to evolve detection programs for predicting the bottom corner of the ruler and incisal upper incisor landmark is similar.

5.7 Summary

The aim of this chapter was to determine if genetic programming is able to evolve a detection program that is accurate enough for the purpose of predicting the position of cephalometric landmarks. This was achieved experimentally by focusing on how to formulate genetic programming for the landmark detection problems presented in this study. The investigation focused on: (a) how to improve fitness evaluation for reducing the number of false alarms, (b) a comparison between handcrafted and domain independent features and (c) an investigation of different function sets. To improve the robustness of the approach, the experimental work was conducted on a selection of landmarks ranging from easy to difficult. The findings from this chapter will form the foundation for future work and become the investigative basis to be used in the subsequent chapters. The investigation indicated that:

- When it is known there is only one object present in an image, using the highest output for predicting the position of the landmark produces significantly less false alarms compared with the domain independent approach described in Chapter 4. We also demonstrated that the accuracy of the detection program could be improved by minimising the error – the distance between the known and predicted position – in the fitness function. However, this fitness metric was not as effective as detection rate for landmarks that are difficult to locate. A fitness metric that uses detection rate was shown to be a good measure of performance.
- The results indicate that the $\{+, -\}$ function set on average produces a detection program that is comparable to or better than the $\{+, -, \times, /\}$ and $\{+, -, \times, /, \min, \max\}$ function sets. However, this comparison is driven by average and not optimal. An analysis of the programs from the best performing evolutionary runs (i.e. the top 10% of evolutionary runs) indicates that the $\{+, -, \times, /\}$ function set produces programs

that are comparable to, or better than the $\{+, -\}$ function set when applied to test data.

- The results suggest that handcrafted shapes specific to a particular landmark are able to produce better performing programs when applied to difficult landmarks when compared to pixels as features.

Chapter 6

Learning with Features from Pulse Coupled Neural Networks

6.1 Introduction

Previous work in Chapter 5 has improved the domain independent approach of genetic programming by using handcrafted shapes and the highest output to predict the position of the landmark. A fitness measure using detection rate was used as a measure of performance. This approach was tested and verified on a range of cephalometric landmarks with varying levels of success. The handcrafted shapes are manually constructed and contained within a moving input window. The set of shapes is used for calculating feature values based on image statistics of the grey level intensities. The features correspond to terminals that are used as inputs for the genetic programming method.

We have established in Section 5.6 that detection programs evolved using pixel based features were not as successful as a specific set of handcrafted shapes devised for each landmark. However, the process for determining what are useful shapes is tedious, time consuming and open to interpretation. Therefore, the initial investigation of this chapter is to determine if it is possible to automatically capture useful regions of interest by generating useful shapes using a segmentation algorithm. If this question is answered in the affirmative, then will these shapes improve the detection performance compared to the handcrafted shapes from

Section 5.3.1?

6.2 Can regions of interest be extracted using a segmentation algorithm?

The aim of this section is to determine if we can achieve an accurate representation of a landmarks shape using a segmentation algorithm for highlighting regions of interest. The segmented shapes will subsequently be used for calculating features and used as part of the genetic search. Whilst there are many segmentation techniques used in image processing, a promising technique that was discussed Section 2.6 for segmenting regions of interest in complex images is the Pulse Coupled Neural Network (PCNN). The PCNN is a relatively new edge detection and segmentation method that has produced promising results in segmenting regions of interest in medical images [67, 72, 86, 159].

The difficulty of segmenting regions of interest in our problem is that a number of areas located within the bony tissue of the head are subject to both noise and low contrast. While fairly accurate segmentation can be achieved for a few images using segmentation algorithms, many of the algorithms tend to fail when applied to a larger suite of images unless parameters are constantly adjusted. Because the ultimate goal of our problem is to develop a methodology that is automated, we are trying to avoid the scenario of interfering with the methodology by continually altering parameters in an ad hoc manner to achieve the best possible outcome.

6.2.1 Pulse Coupled Neural Network segmentation

The aim of this section is to determine whether the outputs from the Pulse Coupled Neural Network (PCNN) are able to produce a segmented output that highlights regions of interest that will be useful for landmark detection. It is anticipated that segmenting the output will provide useful shapes for landmark detection programs by assisting with discriminating the landmark from background. The PCNN algorithm used in these experiments is based on the code from Lindblad and Kinser [85]. The PCNN will focus on highlighting regions of interest on both soft and bony tissue. The segmented regions will later be used to assist with locating the position of the landmark. To determine whether pre-processing by a PCNN could be

useful in landmark detection four landmarks are selected. Two easy landmarks (the menton and upper lip landmarks), one of medium difficulty (incisal upper incisor landmark) and an extremely hard one (sella landmark) are selected. The sella landmark is located in an area of bony tissue that is shown on the X-ray as subtle changes in greyscale. The other landmarks are located on the edge of bone/soft-tissue and soft-tissue. The different landmark types used in Chapter 5 that regularly achieved 100% detection performance have been omitted from this investigation. By using the PCNN it is anticipated that a set of parameters for each landmark will produce a binary image that have highlighted shapes relevant for landmark identification.

6.2.1.1 Segmentation results

The results shown in Figure 6.1 are the binarised output from the PCNN applied to four types of landmarks. A set of parameters was empirically determined for each type of landmark prior to segmentation. The parameter values remained constant throughout segmentation of the training data. The parameters shown in Figure 6.1 are used for segmenting regions of interest for each type of landmark.

To determine the likelihood that the PCNN output could be used to assist with locating landmarks, the segmented outputs were manually classified into three categories, i.e. *Definitive*, *Partially defined* and *Failure*. The categories qualify the output and establish the validity of the PCNN parameters. Table 6.1 is a summary of segmentation results from the PCNN applied to the four types of landmarks. The results indicate the PCNN method was able to accurately segment regions located on the edge of bone/soft-tissue or soft-tissue, however, the technique was less successful for highlighting the semi-circular region that encompasses the sella landmark. The reduced segmentation success rate for the sella landmark is because of the low contrast between the region of interest (semi-circular region) and background. The images shown in Figure 6.1 are a sample of results from a dataset of 83 images.

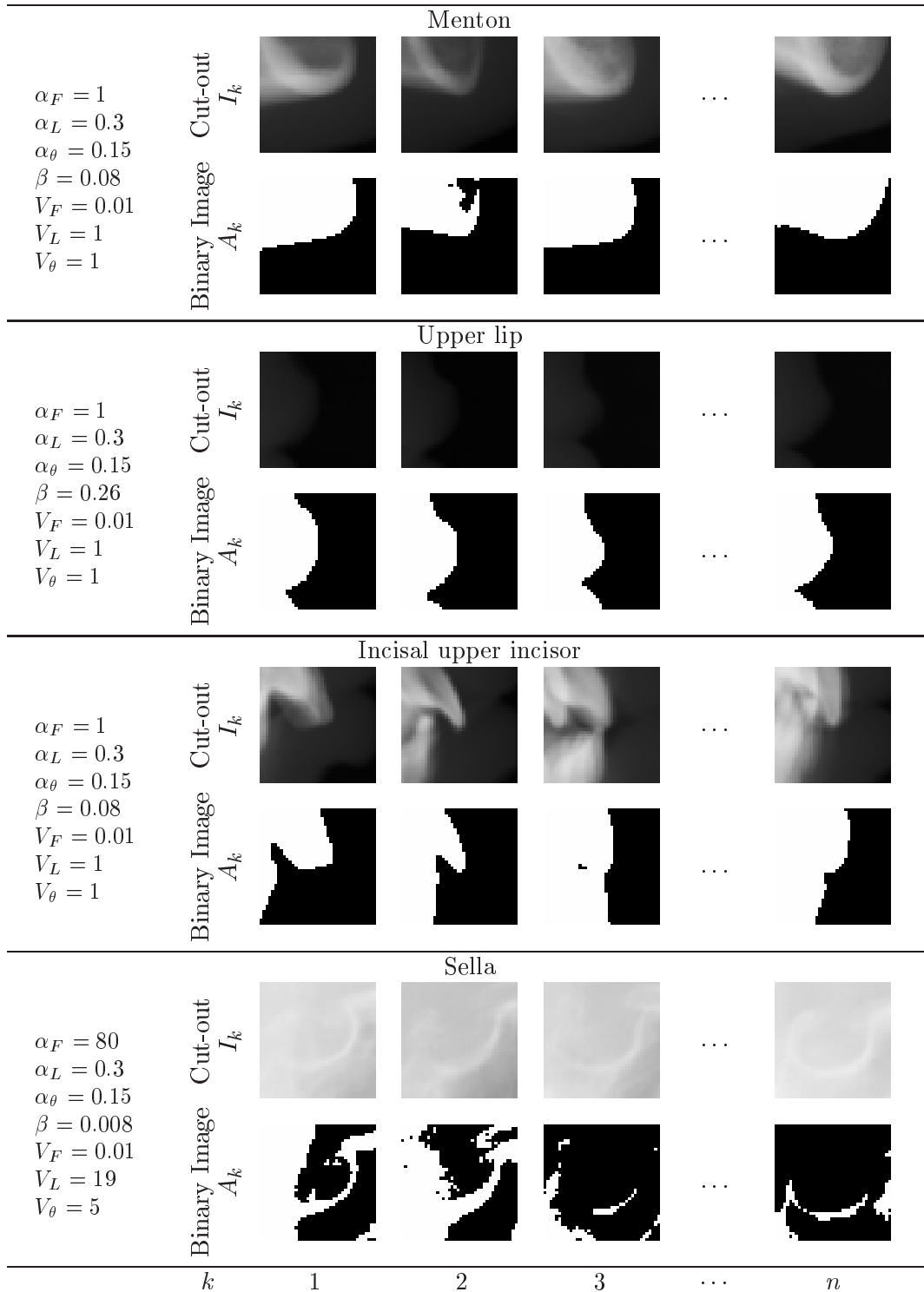


Figure 6.1: Segmentation results using the PCNN for the menton, upper lip, incisal upper incisor and sella cut-outs of size 40×40 pixels. Below each image cut-out is the corresponding binary image output from the PCNN. The values in the left column correspond to the parameters of the PCNN used to achieve the binary image. The images have been scaled by 130% to enhance clarity.

	Menton	Upper lip	Incisal	Sella
Definitive (%)	88.0	97.6	85.5	32.5
Partially defined (%)	12.0	2.4	12.0	33.7
Failure (%)	0	0	2.4	33.7

Table 6.1: Summary of segmentation results for the four types of landmarks as shown in Figure 6.1. Results are based on a set of 83 images

6.2.1.2 PCNN derived shapes

The PCNN is used to highlight a region of interest that may assist with predicting the position of the landmark. Because one binary image is created from each cut-out, the method is limited to extracting two shapes (i.e. black and white regions) from each cut-out as shown in Figure 6.1. The PCNN derived shapes are created as follows:

1. An image cut-out is centred on the known position of the landmark with the dimensions of the cut-out predetermined by the input window's square size in Figure 5.2 on page 85. The PCNN is then applied to the image cut-out, I_k , to produce a binary image, $A_k(i, j)$.
2. Step one is iteratively applied to each image in the training set.
3. The binary image output from the PCNN, $A_k(i, j)$, is used to create a template matrix, $Template$, by computing the average at each pixel position. The template is calculated using Equation 6.1.

$$Template(i, j) = \frac{1}{n} \sum_{k=1}^n A_k(i, j), \quad (6.1)$$

where $A_k(i, j)$ is the segmented image and n is the number of images in the training set. An example template for each landmark is shown in the top row of Figure 6.2.

4. Two shapes are extracted by thresholding the $Template$ (refer to Algorithm 2). Shapes A and B correspond to white and black pixels in the bottom row of Figure 6.2. The term given to describe these shapes is *PCNN derived shapes*.

The two PCNN derived shapes are used to calculate features M_1 , S_1 , M_2 and S_2 .

```

if Template(i,j) ≥ 70 then
  Shape A
else
  Shape B

```

Algorithm 2: The procedure for segmenting *Template* into two distinct shapes.

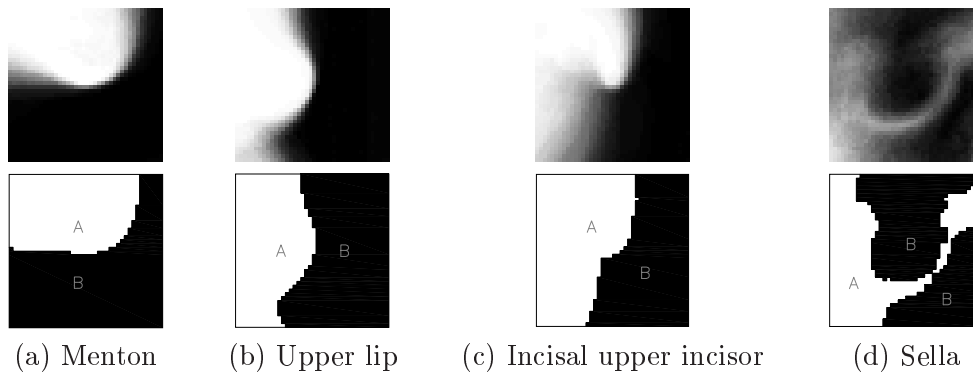


Figure 6.2: Templates computed using the output from a PCNN.

Grey pixels in the template, as shown in the top row of Figure 6.2, indicate the PCNN has either not produced an ideal segmentation or there is biological variability that has been captured whilst averaging the binary outputs. Grey pixels are defined as pixels having grey level intensities in the interval $[1, 254]$. Because the PCNN method performs well at extracting soft and bony tissue, the grey pixels for the menton, upper lip, and incisal upper incisor landmarks are caused by biological variability. However, segmentation results for the sella cut-out were less successful and achieved a definitive segmentation of only 32.5%. This produces a lower contrast template compared to the other three landmarks.

A value of 70 was chosen to threshold the template into two distinct regions, i.e. shape *A* and shape *B*. This value was empirically determined for the sella template and was subsequently applied for thresholding the templates for the other landmarks shown in Figure 6.2. If we could improve the process for segmenting regions of interest in areas that are subject to subtle changes of greyscale, then this may improve the PCNN derived shapes.

This section demonstrated that shapes can be automatically generated using a PCNN. The PCNN derived shapes have found regions of interest that we believe will be useful for locating landmarks.

6.3 Genetic Programming: Learning from PCNN derived shapes

6.3.1 Motivation

Previous work in Section 5.6 has compared the performance of programs that were evolved from features using handcrafted shapes with pixel based features. The outcome was that handcrafting shapes specific to a landmark are able to produce better performing programs when applied to a difficult landmark. However, choosing a set of handcrafted features is difficult and time consuming. Therefore, we would like to determine if the detection performance of programs using PCNN derived shapes is comparable to programs that used handcrafted shapes from Section 5.3.1.

6.3.2 Methodology

We will investigate three feature sets that use PCNN derived shapes. Each feature set is a progression from manually created handcrafted shapes to a method that can generate shapes automatically. The feature sets are defined in Table 6.2.

The first feature set uses handcrafted shapes from Section 5.2 and is used as a benchmark for measuring detection performance for the following three feature sets. The second feature set substitutes the PCNN derived shapes for some of the handcrafted shapes from the previous feature set. This allows us to determine whether PCNN derived shapes can improve detection performance using the same number of terminals. The aim of the third feature set is to measure the detection performance for programs that use only PCNN derived shapes. The fourth feature set determines if additional square shapes combined with PCNN derived shapes can improve detection performance. The fourth feature set is automatically generated. The feature values are calculated from the means and standard deviations of grey level pixel values for each shape. To determine the effectiveness of each feature set for evolving programs, four landmarks of varying detection difficulty have been selected. They are the menton, upper lip, incisal upper incisor and sella landmarks.

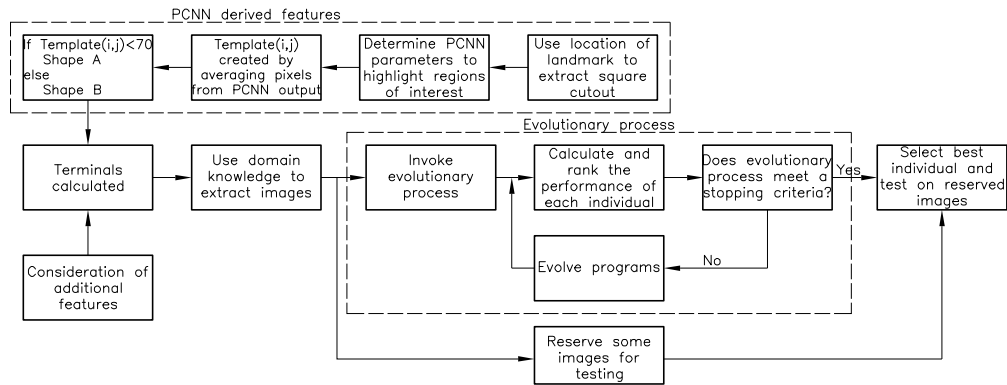


Figure 6.3: Diagram depicting an approach for extracting PCNN and additional features from the extracted images along with the methodology for evolving and evaluating detection programs for the task of locating a landmark.

Case	Feature Set
1	Handcrafted shapes
2	PCNN derived + handcrafted shapes
3	PCNN derived shapes only
4	PCNN derived shapes + quadrants

Table 6.2: Definition of four feature sets that are made available during the genetic search.

The use of genetic programming for landmark detection is similar to the methodology described in Section 5.2 on page 82 with the exception of steps 3 and 4. A schematic for describing the methodology is shown in Figure 6.3. The amended items are as follows:

3. The size of the input window that is applied to each landmark type is determined by the *square size* defined in Figure 5.2. This gives an unbiased comparison for the detection performance of programs that are evolved using the different feature sets.
4. Three configurations using the PCNN derived shapes defined in Figure 6.2 will be conducted. The shapes are specific to capturing a landmark's characteristics and also discriminating against background. The results from each of these experiments will be compared with handcrafted shapes defined in Section 5.3.1.

6.3.3 The Function Set

An investigation of different operators in Section 5.5 established that a function set consisting of $(+, -, \times, /)$ is a good selection of operators for evolving detection programs.

6.3.4 Fitness Evaluation

An investigation of alternative approaches for evaluating fitness in Chapter 5 demonstrated that using the highest output for predicting the position of a landmark produces significantly less false alarms than the domain independent approach. The fitness of a program will be measured as per Equation 5.1 in Section 5.4.1, i.e. $Fitness = (1 - DR)$. The calculation for detection rate, DR , is defined in Section 2.2. The tolerance for correctly locating a landmark is 2 mm (5 pixels) of the position located by the orthodontist.

6.3.5 Case2: PCNN derived and Handcrafted shapes

6.3.5.1 Motivation

The aim of this section is to determine the effectiveness of the PCNN derived shapes when compared to the handcrafted shapes. This is achieved by using a selection of the handcrafted shapes from Section 5.3.1 and the PCNN derived shapes and then determining if the detection rate is significantly different.

The two handcrafted shapes in Section 5.3.1 are substituted with the corresponding PCNN derived shapes as shown in Figure 6.4. The reason for selecting these handcrafted shapes is because of the resemblance to the PCNN derived shapes. This will determine if an accurate representation of a landmark's region of interest using PCNN derived shapes will improve detection rate compared to the handcrafted shapes. There were no similarities between the PCNN derived shapes and the handcrafted shapes for the incisal upper incisor, so we selected two features that were commonly chosen by GP as a result of the evolutionary process. Substituting the PCNN derived shapes into the handcrafted set of shapes has created a hybrid set that we have described as *PCNN derived + handcrafted shapes*. The reason for substituting rather than appending the two shapes to the feature set is to compare the program's performance and not increase the number of features that are available for selection

during the genetic search.

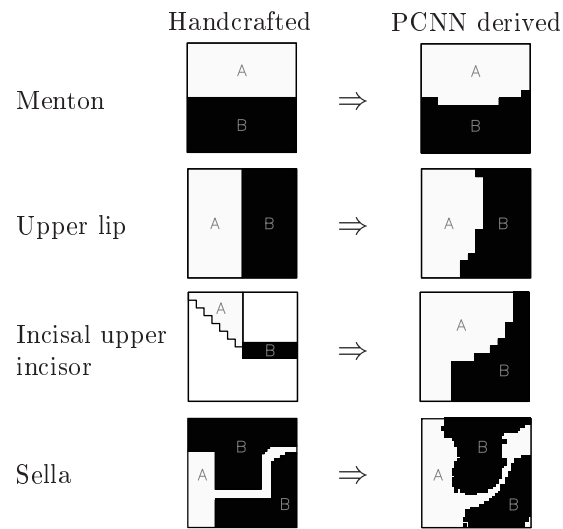


Figure 6.4: Substituting two handcrafted shapes, i.e. A and B , for PCNN derived shapes.

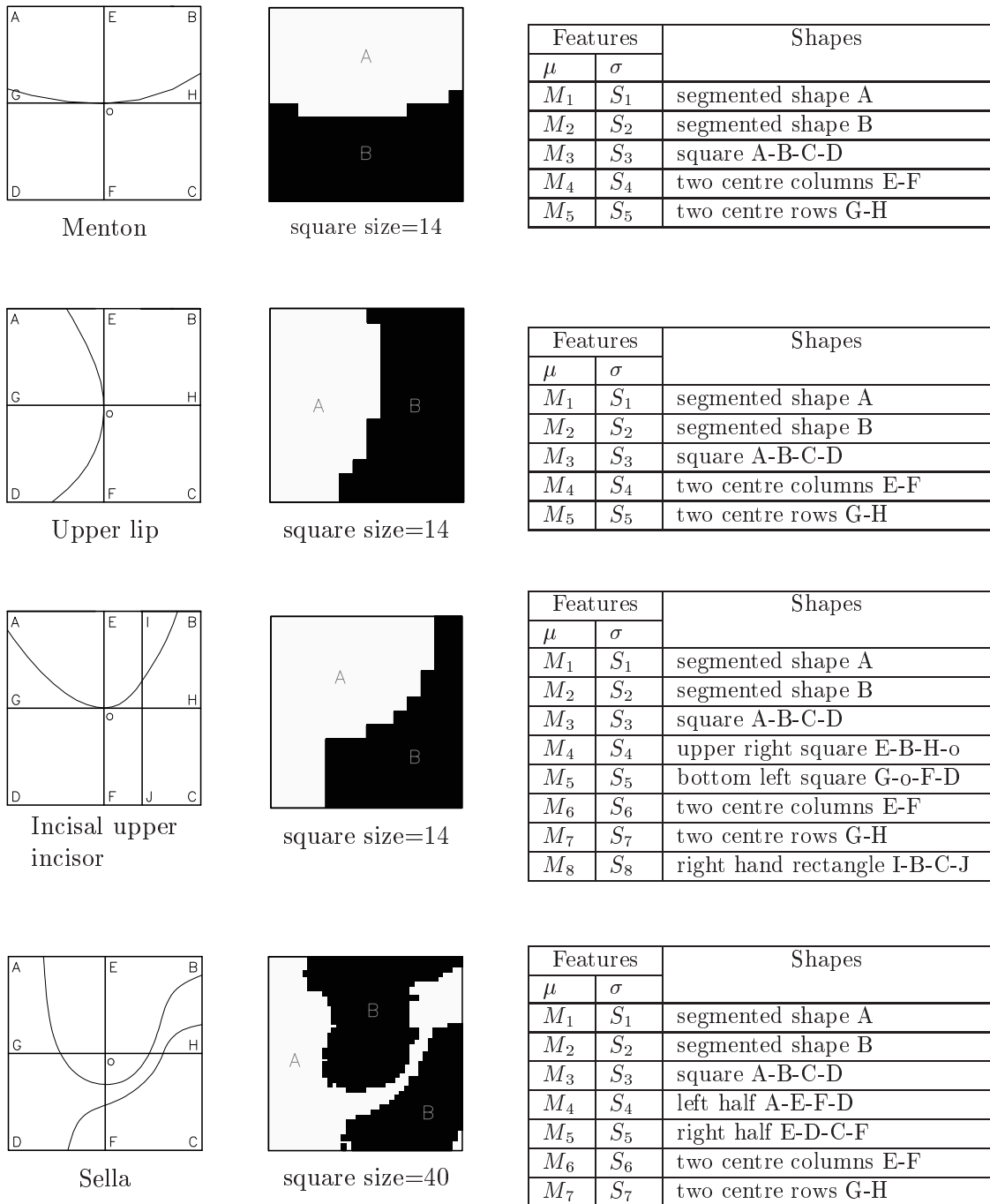


Figure 6.5: The diagrams in the left column are shapes that have been manually determined. The second column are shapes automatically extracted using the PCNN template. The shapes manually and automatically extracted are combined to produce the feature set for the menton, upper lip, incisal upper incisor and sella landmarks. A feature set consists of the mean and standard deviation of pixel intensities for each shape.

6.3.5.2 Results

The aim is to determine the effectiveness of the PCNN derived shapes by measuring the detection performance of programs that are evolved from *PCNN derived + handcrafted shapes*. The performance of the programs will then be compared with programs evolved using handcrafted shapes. A two-sample t test is used to determine if there is a difference in mean detection rate between two samples of programs using: (a) *PCNN derived + handcrafted shapes* and (b) *handcrafted shapes*. Let the null hypothesis be that the two samples are from the same population.

The p -value is less than an alpha value of 0.05, the critical ‘cut-off’ boundary, for the menton, upper lip and sella landmarks. Therefore the null hypothesis, that the mean detection rate of programs evolved using *PCNN derived + handcrafted shapes* and *handcrafted shapes* is the same, can be rejected. The average detection rates for menton, upper lip and sella landmarks has increased by 5.77%, 15.69% and 18.67% respectively by the inclusion of the PCNN derived shapes. There is no evidence to suggest that the average detection rate for the incisal upper incisor had changed. The same hypothesis for comparing the mean detection rate of programs are supported for each landmark when applied to the test images. The average detection performance has increased for the menton, upper lip and sella landmarks by 10.56%, 19.31% and 29.67% respectively. The mean detection rate of the best program has increased more for test data than training data. This suggests that programs using the PCNN derived shapes have generalised better compared to programs that used handcrafted shapes. This is shown as a scatter plot in Figure 6.6. This analysis demonstrates that improving the representation of a landmark’s region of interest improves the detection performance of programs.

		Training		Testing	
		Case 1	Case 2	Case 1	Case 2
Menton	Average				
	detection (%)	93.07	98.84	83.47	94.03
	false alarm(%)	6.93	1.16	16.53	5.97
	<i>p</i> -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	100	100	96.30	100	
false alarm(%)	0	0	3.70	0	
Upper lip	Average				
	detection (%)	83.74	99.43	63.70	83.01
	false alarm(%)	16.26	0.57	36.30	16.99
	<i>p</i> -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	90.36	100	77.78	92.59	
false alarm(%)	8.64	0	22.22	7.41	
Incisal upper incisor	Average				
	detection (%)	94.61	94.44	88.98	88.80
	false alarm(%)	5.39	5.56	11.02	11.20
	<i>p</i> -value				
	detection (%)	0.675		0.787	
	Best program				
detection (%)	98.80	97.59	92.59	96.30	
false alarm(%)	1.20	2.41	7.41	3.70	
Sella	Average				
	detection (%)	54.73	73.40	44.40	74.07
	false alarm(%)	45.27	26.60	55.60	25.93
	<i>p</i> -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	69.51	82.93	62.96	85.19	
false alarm(%)	30.49	17.07	37.04	14.81	

Table 6.3: A comparison of detection performance for programs that were evolved using *handcrafted shapes* (Case 1) defined in Section 5.2 with *PCNN derived + handcrafted shapes* (Case 2). The averages are calculated from the best individual from each run for 80 evolutionary runs. The *p*-value is calculated from a two-sample *t* test to compare the mean detection rate from two independent samples.

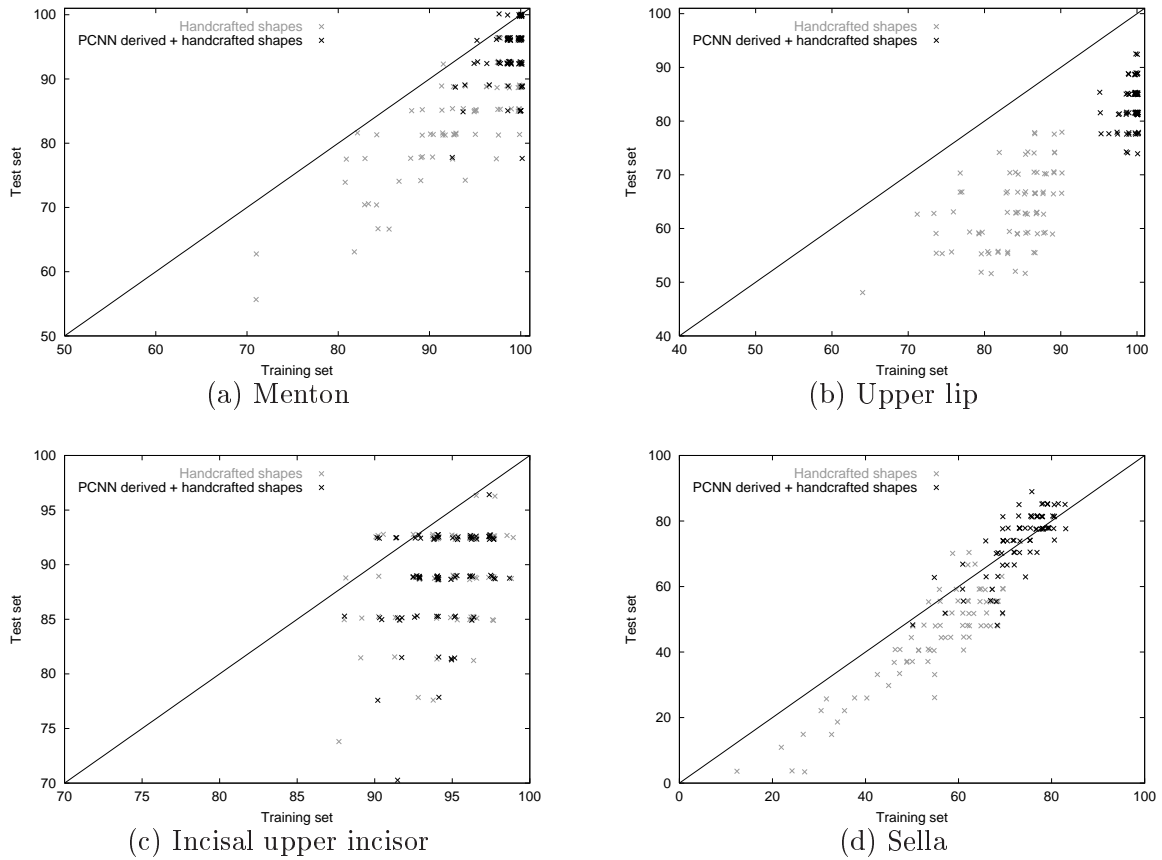


Figure 6.6: A comparison of detection performance for programs that were evolved from *PCNN derived and handcrafted shapes* with *handcrafted shapes*. The points are the detection rate of the best program for each run from 80 evolutionary runs.

Figure 6.7 is a selection of randomly chosen examples of four different landmark types. The positions of each landmark type are predicted by using the best performing program of 80 evolutionary runs. The best program for each landmark is shown in Figure 6.8. The coordinates below each image show the detection error defined as the difference between the predicted and known position. If the error is in excess of a Euclidean distance of 5 pixels (2 mm) then this is regarded as a false alarm. The landmarks in all the images shown in Figure 6.7 were predicted within the acceptable tolerance with the exception of a sella landmark in the bottom row of Figure 6.7. The landmark for this image was recorded as a false alarm because the predicted error was (41, 35) pixels or 21.9 mm from the correct location.

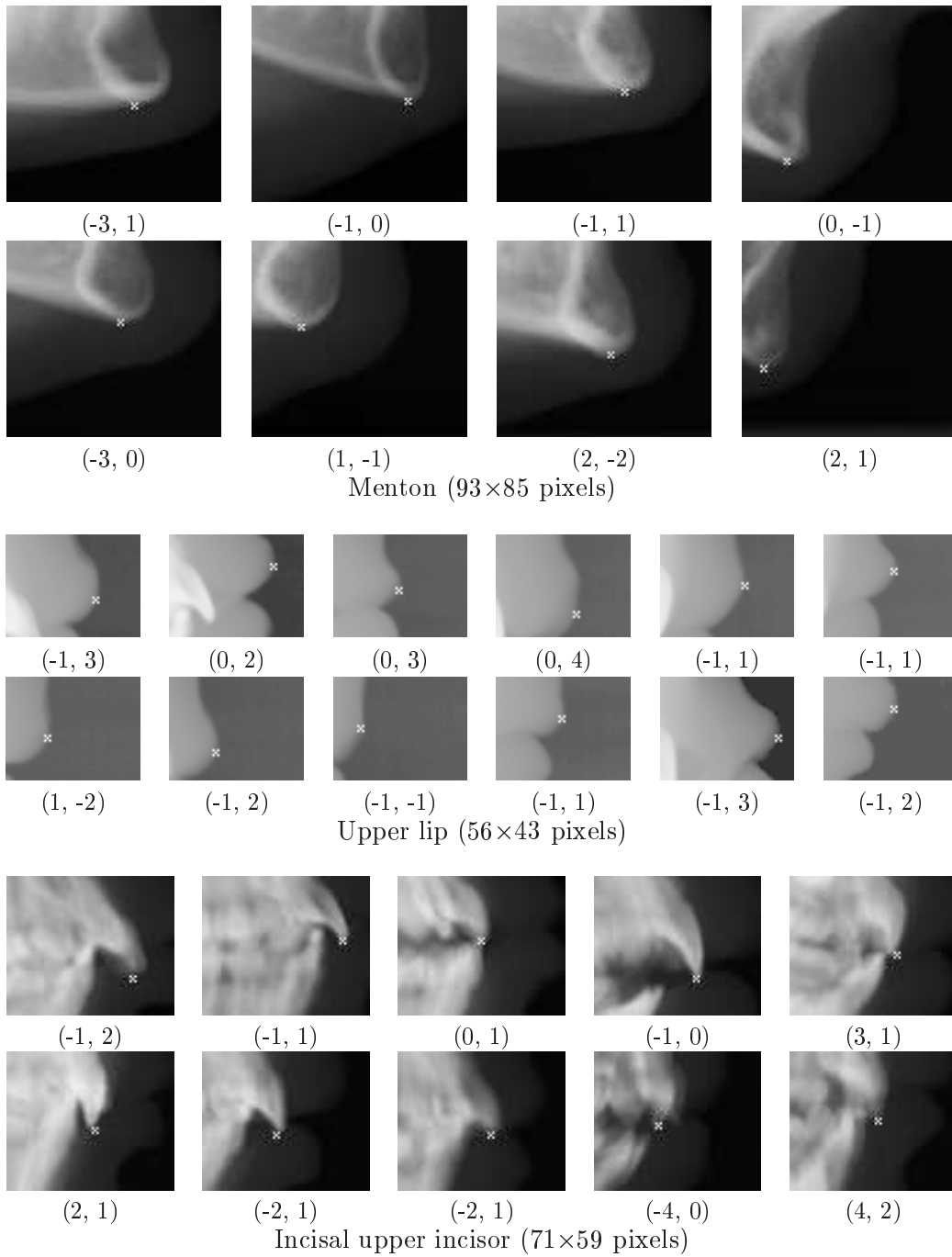


Figure 6.7: The rows from top to bottom are indicative of the variation in biological shapes for the menton, upper lip, incisal upper incisor and sella landmarks respectively. The position found by the detection program is marked with the cross. The positional error (pixels) is displayed under each image.

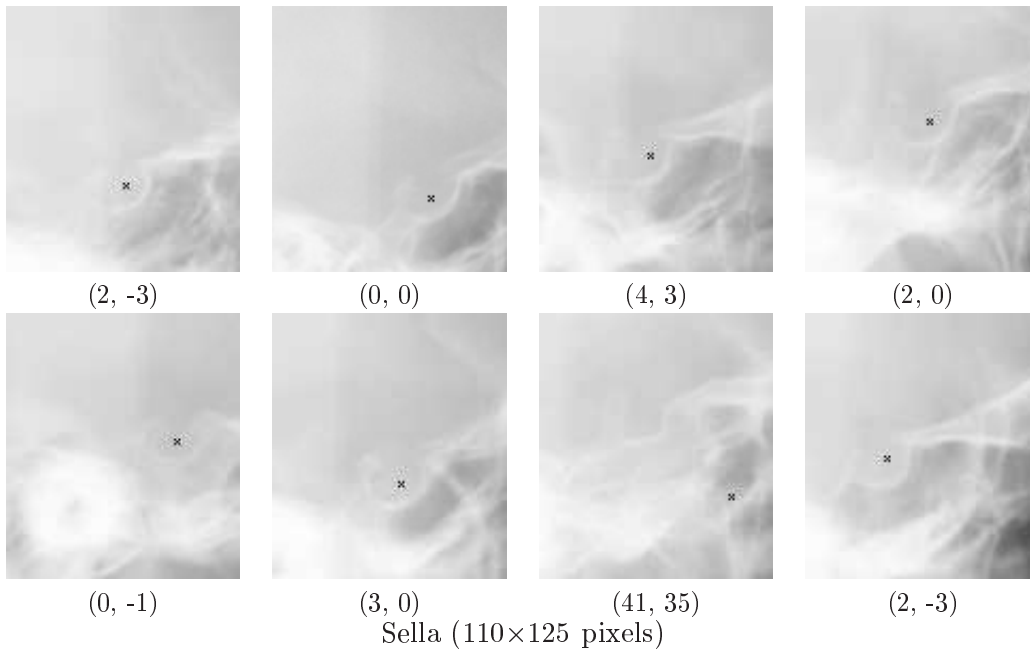


Figure 6.7 (continued)

```
(* (- (* M5 162.198) (+ S1 (* M5 162.198))) (+ (* (- S1 S3) (- S5 S4)) (-
(* (- S1 (- S3 S2)) M1) S5)))
```

Menton program (Fitness=0)

```
(/ (/ M4 (+ S1 (* 230.061 (+ M2 S2)))) (+ (+ 230.061 (- (- (/ M1 (+ S2 (-
M1 73.1477))) S4) (+ (/ S1 S1) M3))) (* (+ 234.024 M3) (+ M2 S2))))
```

Upper lip program (Fitness=0)

```
(* (* (/ M6 M7) (+ M3 (- M8 219.569))) (- M2 (+ M3 M8)))
```

Incisal upper incisor detection program (Fitness=2.41)

```
(/ (/ (/ (/ (+ (* M5 M6) (/ S7 S3)) (/ S2 M2)) (/ S1 S5)) (- (/ (+ (/ S7 M2)
(- (/ S7 S3) (- S5 M2))) (+ (+ (* M6 87.2767) (- S3 S2)) M5)) (+ (/ M5 S6)
M5))) (/ (+ (/ (+ M2 (- S2 S5)) (/ (* S7 M5) M2)) (+ (* S5 S5) M2)) (/ (/ (*
(- M3 M1) (- (- S2 S5) S5)) (+ (* M1 S5) (/ M5 S7))) (+ (- (- 58.6679 M5) (/
S6 M7)) S7))))
```

Sella detection program (Fitness=17.07)

Figure 6.8: The programs expressed as LISP S-expressions are the best individuals from 80 evolutionary runs for the menton, upper lip, incisal upper incisor and sella landmarks.

As evident from the experiments, the size of the program increased with the difficulty of the landmark. The best performing programs from 80 evolutionary runs are shown in Figure

6.8. It is not conclusive that the program sizes increased because of fitness or an accumulation of introns during an evolutionary run. No parsimony factor was used to control bloat during training.

6.3.6 Case 3: PCNN derived shapes only

6.3.6.1 Motivation

In the previous section we analysed the effectiveness of the PCNN derived shapes by comparing them with the simplified handcrafted shapes shown in Figure 6.4. This work demonstrated that incorporating shapes that are a better representation of a landmark's region of interest improves the detection performance of evolved programs. However, the previous feature set was partially constructed from handcrafted shapes and we would like a method that can automatically generate the feature set. As an intermediate step in automatically generating a feature set, we would like to determine how well the PCNN derived shapes can be used for detecting the previous set of landmarks without using handcrafted shapes.

Features		Shapes
μ	σ	
M_1	S_1	segmented shape A
M_2	S_2	segmented shape B

Figure 6.9: The feature set consists of features derived from the PCNN derived shapes. Features are calculated using the mean and standard deviation of pixel intensities for each shape.

6.3.6.2 Results

The aim is to measure the detection performance of programs evolved from a feature set consisting of only PCNN derived shapes. The detection performance of programs using handcrafted shapes will be used as a benchmark. A two-sample t test is used to determine if there is a difference in mean detection rate between two samples of programs using: (a) *PCNN derived shapes* and (b) *handcrafted shapes*. Let the null hypothesis be that the two samples are from the same population.

The p -value is less than an alpha value of 0.05 for all four landmarks, i.e. menton, upper lip, incisal upper incisor and sella landmarks. Therefore the null hypothesis, that the mean

detection rate of programs evolved using *PCNN derived shapes* and *handcrafted shapes* is the same, can be rejected. The average detection rate for the menton, upper lip and sella landmarks has increased by 3.45%, 15.36% and 20.15% respectively, however, the average detection rate for the incisal upper incisor has decreased by 38.28%. The same hypothesis for comparing the mean detection rate of programs is supported for each landmark type when applied to the test images.

The decrease in detection performance for the incisal upper incisor landmark suggests that the PCNN derived shapes are not as effective at discriminating a landmark from background as the menton, upper lip and sella landmarks. This statement is supported by the decrease in detection rate for the incisal upper incisor whereas there was a significant increase in detection rate for the other landmark types. This suggests that the PCNN derived shapes were either not a good representation of the incisal upper incisor or that additional shapes are required in order to evolve comparable detection programs.

		Training		Testing	
		Case 1	Case 3	Case 1	Case 3
Menton	Average				
	detection (%)	93.07	96.52	83.47	92.64
	false alarm(%)	6.93	3.48	16.53	7.36
	<i>p</i> -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	100	100	96.30	96.30	
false alarm(%)	0	0	3.70	3.70	
Upper lip	Average				
	detection (%)	83.74	99.10	63.70	78.52
	false alarm(%)	16.26	0.90	36.30	21.48
	<i>p</i> -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	90.36	100	77.78	85.19	
false alarm(%)	8.64	0	22.22	14.81	
Incisal upper incisor	Average				
	detection (%)	94.61	56.33	88.98	51.57
	false alarm(%)	5.39	43.67	11.02	48.43
	<i>p</i> -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	98.80	71.08	92.59	55.56	
false alarm(%)	1.20	28.92	7.41	44.44	
Sella	Average				
	detection (%)	54.73	74.88	44.40	71.76
	false alarm(%)	45.27	25.12	55.60	28.24
	<i>p</i> -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	69.51	79.27	62.96	77.78	
false alarm(%)	30.49	20.73	37.04	22.22	

Table 6.4: A comparison of detection performance for programs that were evolved using *handcrafted shapes* (Case 1) with *PCNN derived shapes* (Case 3). The averages are calculated from the best individual from each run for 80 evolutionary runs. The *p*-value is calculated from a two-sample *t* test to compare the mean detection rate from two independent samples.

6.3.7 Case 4: PCNN derived shapes and quadrants

6.3.7.1 Motivation

In the previous section the aim was to determine how well the PCNN derived shapes can be used for detecting a selection of landmarks by comparing with the detection performance of programs that were evolved from handcrafted shapes. We found that PCNN derived shapes were not as effective for detecting the incisal upper incisor compared to the other three types

of landmarks. This suggests that additional shapes may have been useful for improving the detection performance of programs. This section will investigate if additional shapes created without any prior knowledge of the landmark, or independent of domain, can improve the detection performance for the incisal upper incisor.

We discussed in Section 2.5.2.4 that feature sets were generally a composition of various shapes within an input window. One of those feature sets used by Zhang et al. in [165] was created by dividing the input window into four quadrants. The aim of this section is to determine if features calculated using PCNN derived shapes in conjunction with the quadrants can improve the detection rate in comparison with programs that were evolved using handcrafted shapes.

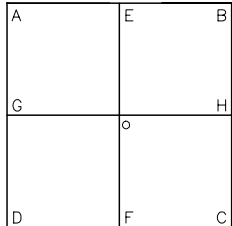
		Features		Shapes
		μ	σ	
		M_1	S_1	segmented shape A
		M_2	S_2	segmented shape B
		M_3	S_3	square A-B-C-D
		M_4	S_4	upper left quadrant A-E-o-G
		M_5	S_5	upper right quadrant E-B-H-o
		M_6	S_6	bottom left quadrant G-o-F-D
		M_7	S_7	bottom right quadrant o-H-C-F

Figure 6.10: The feature set consists of features derived from the PCNN derived shapes and quadrants. Features are calculated using the mean and standard deviation of pixel intensities for each shape.

6.3.7.2 Results

The aim is to measure the detection performance of programs that are evolved from a feature set that has been automatically generated. The feature set consists of PCNN derived shapes and quadrants. The detection performance of programs using handcrafted shapes will again be used as the benchmark. A two-sample t test is used to determine if there is a difference in mean detection rate between two samples of programs using: (a) *PCNN derived shapes and quadrants* and (b) *handcrafted shapes*. Let the null hypothesis be that the two samples are from the same population.

The p -value is less than an alpha value of 0.05 for the menton, upper lip and sella landmarks. Therefore the null hypothesis, that the mean detection rate of programs evolved using

PCNN derived shapes and quadrants and *handcrafted shapes* is the same, can be rejected. The average detection rates for the menton, upper lip and sella landmarks has increased by 6.12%, 15.64% and 18.55% respectively. There is no evidence to suggest that the average detection rate for the incisal upper incisor has changed. The same hypothesis for comparing the mean detection rate of programs is supported for each landmark when applied to the test images with the exception of the incisal upper incisor. However, the average detection rate for the incisal upper incisor has decreased by only 1.53%. The average detection performance has increased for the menton, upper lip and sella landmarks by 11.67%, 16.86% and 27.31% respectively.

These results suggest that the evolved programs using features calculated from the PCNN derived shapes and quadrants are comparable to or better than programs that were evolved from handcrafted shapes. An advantage of using PCNN derived shapes and quadrants is that they have been automatically generated without any manual intervention.

		Training		Testing	
		Case 1	Case 4	Case 1	Case 4
Menton	Average				
	detection (%)	93.07	99.19	83.47	95.14
	false alarm(%)	6.93	0.81	16.53	4.86
	p -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	100	100	96.30	100	
false alarm(%)	0	0	3.70	0	
Upper lip	Average				
	detection (%)	83.74	99.38	63.70	80.56
	false alarm(%)	16.26	0.62	36.30	19.44
	p -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	90.36	100	77.78	88.89	
false alarm(%)	8.64	0	22.22	11.11	
Incisal upper incisor	Average				
	detection (%)	94.61	94.19	88.98	87.45
	false alarm(%)	5.39	5.81	11.02	12.55
	p -value				
	detection (%)	0.262		0.028	
	Best program				
detection (%)	98.80	97.59	92.59	92.59	
false alarm(%)	1.20	2.41	7.41	7.41	
Sella	Average				
	detection (%)	54.73	73.28	44.40	71.71
	false alarm(%)	45.27	26.72	55.60	28.29
	p -value				
	detection (%)	0.000		0.000	
	Best program				
detection (%)	69.51	84.15	62.96	81.48	
false alarm(%)	30.49	15.85	37.04	18.52	

Table 6.5: A comparison of detection performance for programs that were evolved using *handcrafted shapes* (Case 1) with *PCNN derived shapes and quadrants* (Case 4). The averages are calculated from the best individual from each run for 80 evolutionary runs. The p -value is calculated from a two-sample t test to compare the mean detection rate from two independent samples.

6.4 Comparison of Feature Sets

This section compares the detection performance of programs that were evolved using the different feature sets presented in this chapter. The different feature sets are presented in Table 6.6. For comparing the four feature sets, a one-way ANOVA procedure [132] is used to measure the differences in mean detection rate. Let the null hypothesis be that the mean

detection rate of programs that have been evolved from the four features is the same, i.e.

$$H_0 : \mu_{\text{case 1}} = \mu_{\text{case 2}} = \mu_{\text{case 3}} = \mu_{\text{case 4}}.$$

The p -values for comparing the detection rate of programs when applied to training images are less than an alpha value of 0.05 for the upper lip, incisal upper incisor and sella landmarks. Therefore, the null hypothesis, that the mean detection rate of programs that were evolved from the four feature set sets is the same, is rejected. This indicates that at least two of the means are significantly different. Since the alternative hypothesis is supported, a Tukey's pairwise comparison [132] was conducted to determine significant differences between the different feature sets. The results of the pairwise comparison are shown in Table 6.8.

An analysis of the pairwise comparison from Table 6.8 and the graph from Figure 6.11 indicates that on average, the feature sets using the PCNN derived shapes produced better performing programs compared to the feature sets using the handcrafted shapes. The incisal upper incisor is an exception. The results for the *PCNN derived shapes only* when applied to the incisal upper incisor indicate that other shapes were required to discriminate the object from background when comparing the results from the other feature sets.

A further analysis of the pairwise comparison reveals that there was no significant difference between programs that were evolved from features using *PCNN derived and handcrafted shapes* and *PCNN derived shapes and quadrants*. The results of programs using the *PCNN derived shapes and quadrants* produced detection programs that were either equivalent to or exceeded the detection performance of the handcrafted shapes from Chapter 5. This is a desirable result since the method to create the *PCNN derived shapes and quadrants* is a process where only a small amount of prior knowledge is required about a landmark.

Case 1	Handcrafted shapes
Case 2	PCNN derived + handcrafted shapes
Case 3	PCNN derived shapes only
Case 4	PCNN derived shapes + quadrants

Table 6.6: Definition of four feature sets that are made available during the genetic search.

		Training				Testing			
		Case 1	Case 2	Case 3	Case 4	Case 1	Case 2	Case 3	Case 4
Menton	Average								
	detection (%)	93.07	98.84	96.52	99.19	83.47	94.03	92.64	95.14
	false alarm(%)	6.93	1.16	3.48	0.81	16.53	5.97	7.36	4.86
	p -value								
	detection (%)	0.000				0.000			
	Best program								
detection (%)	100	100	100	100	96.30	100	96.30	100	
false alarm(%)	0	0	0	0	3.70	0	3.70	0	
Upper lip	Average								
	detection (%)	83.74	99.43	99.10	99.38	63.70	83.01	78.52	80.56
	false alarm(%)	16.26	0.57	0.90	0.62	36.30	16.99	21.48	19.44
	p -value								
	detection (%)	0.000				0.000			
	Best program								
detection (%)	90.36	100	100	100	77.78	92.59	85.19	88.89	
false alarm(%)	8.64	0	0	0	22.22	7.41	14.81	11.11	
Incisal	Average								
	detection (%)	94.61	94.44	56.33	94.19	88.98	88.80	51.57	87.45
	false alarm(%)	5.39	5.56	43.67	5.81	11.02	11.20	48.43	12.55
	p -value								
	detection (%)	0.000				0.000			
	Best program								
detection (%)	98.80	97.59	71.08	97.59	92.59	96.30	55.56	92.59	
false alarm(%)	1.20	2.41	28.98	2.41	7.41	3.70	44.44	7.41	
Sella	Average								
	detection (%)	54.73	73.40	74.88	73.28	44.40	74.07	71.76	71.71
	false alarm(%)	45.27	26.60	25.12	26.72	55.60	25.93	28.24	28.29
	p -value								
	detection (%)	0.000				0.000			
	Best program								
detection (%)	69.51	82.93	79.27	84.15	62.96	85.19	77.78	81.48	
false alarm(%)	30.49	17.07	20.73	15.85	37.04	14.81	22.22	18.52	

Table 6.7: A comparison of programs' detection performance that are evolved using variations of handcrafted, PCNN derived shapes and quadrants. The averages are calculated from the best individual from each run for 80 evolutionary runs. The p -value is calculated using a one-way ANOVA to compare the mean detection rate between the four independent samples.

A comparison of detection performance for training results of programs using *handcrafted shapes* and *PCNN derived shapes and quadrants* reveals that the latter improved detection performance by 15.6% and 18.55% for the upper lip and sella landmarks respectively. The detection performance was also higher during testing with an improvement of 18.55% and 27.31% for the upper lip and sella landmark respectively. There was no significant difference in detection performance for the incisal upper incisor landmark.

The fitness graphs in Figure 6.12 compare the average fitness scores between the four

different feature sets. The evolutionary process was terminated at either 100 generations or when a program achieved a 100% detection rate. All four graphs indicate that the *PCNN derived and handcrafted shapes* and *PCNN derived shapes and quadrants* have a similar rate of convergence when optimising fitness score and has generally outperformed the *Handcrafted shapes* and *PCNN derived shapes only* throughout the evolutionary process.

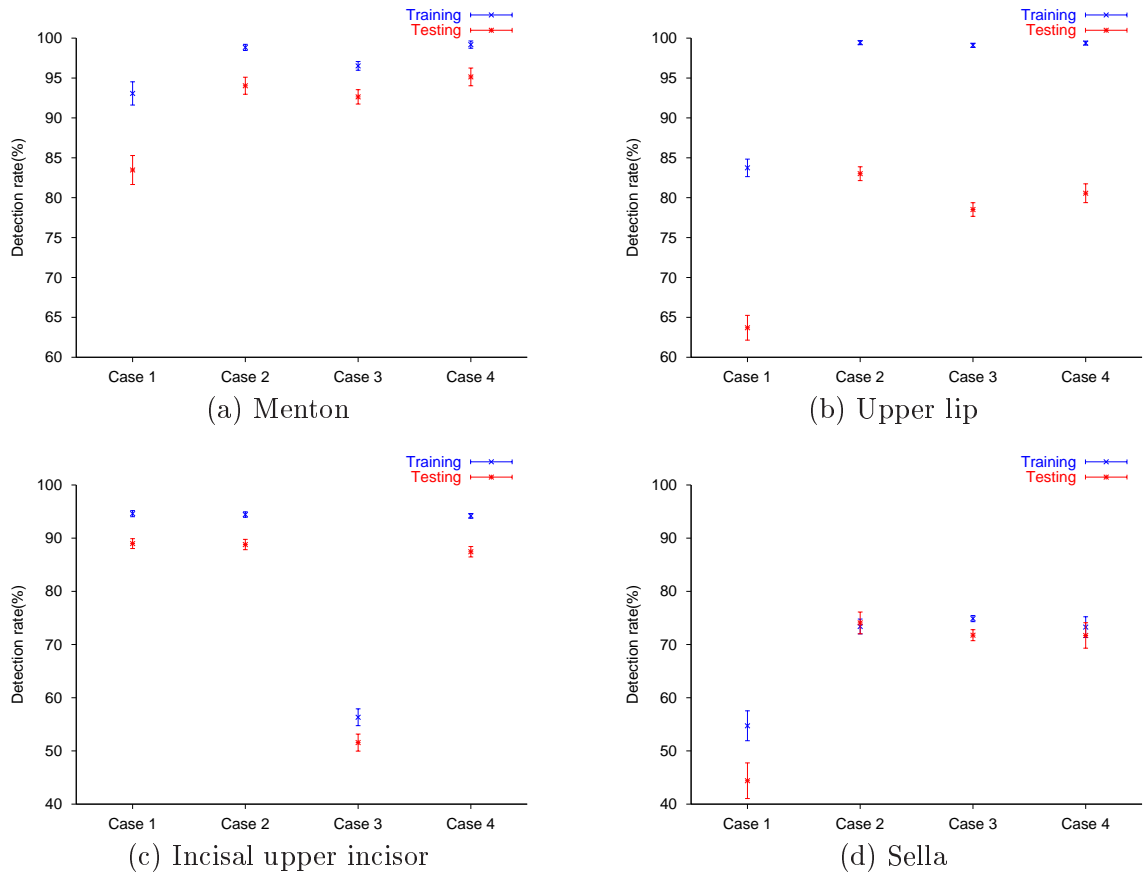


Figure 6.11: A summary of detection performance of programs that are evolved from features using variations of handcrafted, PCNN derived shapes and quadrants. The error bars represent 95% confidence intervals.

	Training				Testing			
Menton	Case	1	2	3	Case	1	2	3
	2	Disimilar			2	Disimilar		
	3	Disimilar	Disimilar		3	Disimilar	Similar	
	4	Disimilar	Similar	Disimilar	4	Disimilar	Similar	Disimilar
Upper lip	Case	1	2	3	Case	1	2	3
	2	Disimilar			2	Disimilar		
	3	Disimilar	Similar		3	Disimilar	Disimilar	
	4	Disimilar	Similar	Similar	4	Disimilar	Disimilar	Similar
Incisal	Case	1	2	3	Case	1	2	3
	2	Similar			2	Similar		
	3	Disimilar	Disimilar		3	Disimilar	Disimilar	
	4	Similar	Similar	Disimilar	4	Similar	Similar	Disimilar
Sella	Case	1	2	3	Case	1	2	3
	2	Disimilar			2	Disimilar		
	3	Disimilar	Similar		3	Disimilar	Similar	
	4	Disimilar	Similar	Similar	4	Disimilar	Similar	Similar

Table 6.8: A Tukey's pairwise comparison of the detection performance of programs that were evolved from four feature sets.

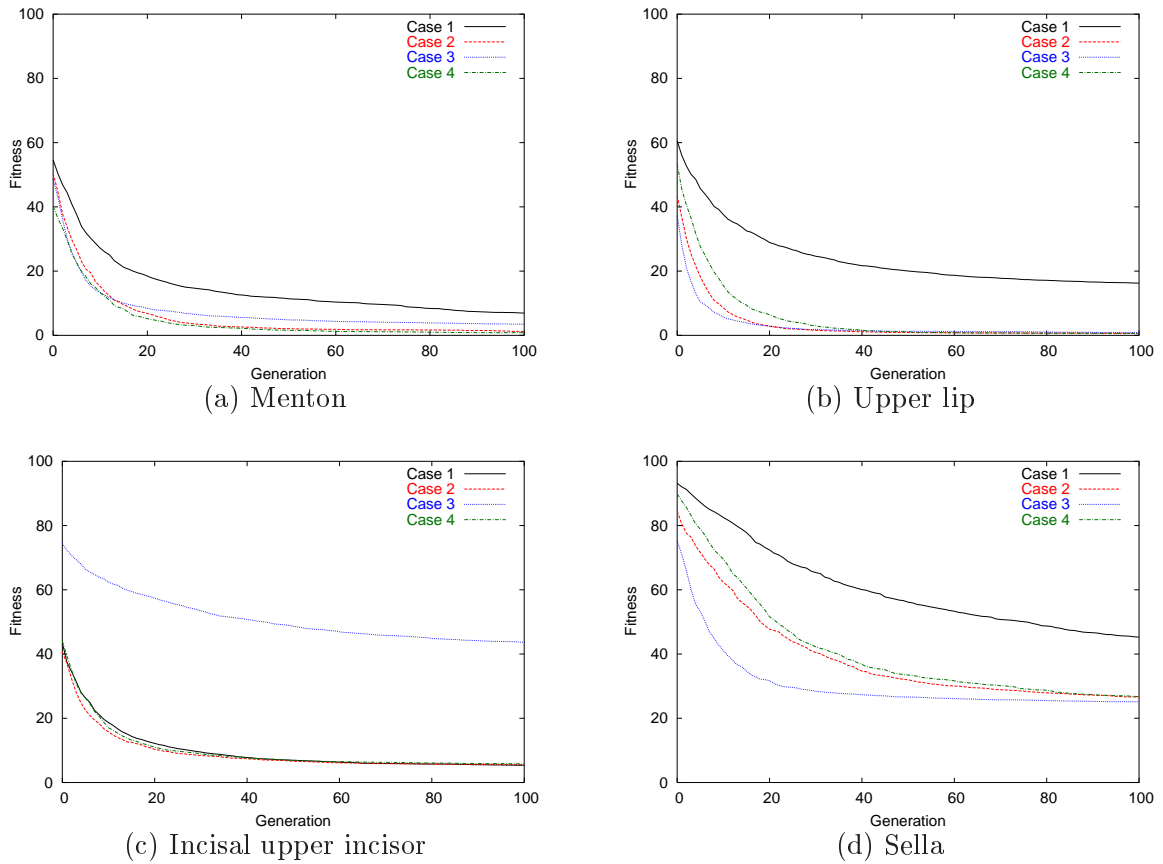


Figure 6.12: The fitness graphs are a comparison of average fitness scores for four different feature sets. The average fitness is calculated by averaging the fitness score of the best individual from each generation for 80 evolutionary runs.

6.5 Determining Input Window Size

6.5.1 Motivation

The aim of this section is to determine how sensitive the selection of the input window size is with respect to detection performance. Previously the size of the input window was manually determined based on the notion of selecting an input window that is large enough to capture sufficient detail in order to differentiate the landmark from background.

The justification for introducing the work here rather than as part of the exploratory work in Chapter 5, is that some of the handcrafted shapes used in the previous chapter were customised for a fixed window size. As a result, any changes to the input window size would

have required manual adjustment of the handcrafted shapes. Therefore, this work is most valuable in this chapter because the PCNN derived shapes and quadrants can be automatically created independent of window size. The analysis for determining the sensitivity of input window size is applied to the menton and upper lip landmarks.

6.5.2 Results

The graphs in Figure 6.13 show the relationship between the size of the input window and the detection performance of a program; the detection performance of a program is based on the average detection rate from the best individual from each run for 80 evolutionary runs. Both graphs indicate that the detection performance of programs is sensitive to the size of the input window. The results for the menton and upper lip landmarks suggest that an input window size should be somewhere between the interval of [14, 40] and [20, 30] respectively. Selecting an input window size with dimensions outside of this interval will on average produce a program that will have a lower detection performance. However, determining what is an ideal input window size using this approach is time consuming, i.e. incrementally testing several size input windows, as the evolutionary process for producing enough runs to convey meaningful statistics is computationally expensive. Therefore, the size of the input window should be selected prior to the evolutionary process and is recommended as future work.

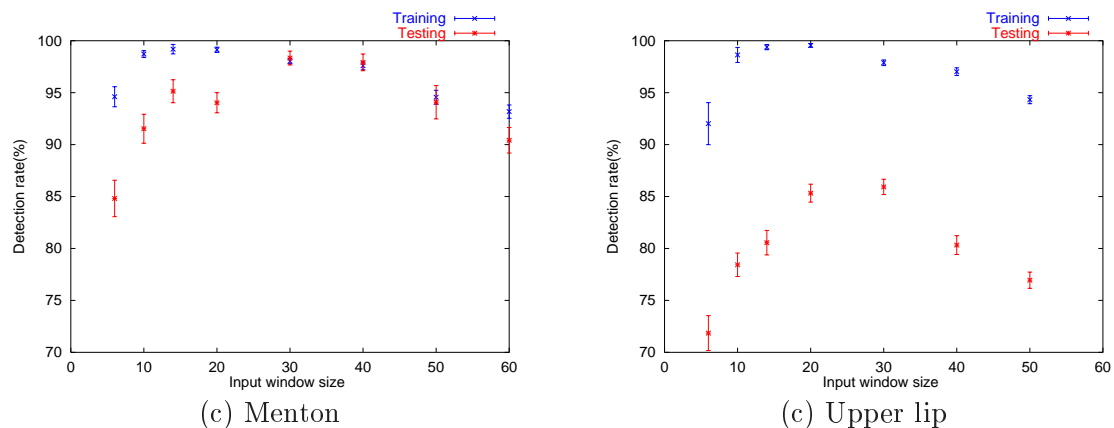


Figure 6.13: The sensitivity of selecting the size of the window with respect to a program's expected performance based on training and test results. The detection rate is based on an average of the best individual from each run for 80 evolutionary runs. The error bars represent a 95% confidence interval.

6.6 Summary

The aim of this chapter was to investigate whether using segmented shapes would improve the detection performance of programs compared to the handcrafted shapes used in Section 5.3.1. The segmented shapes were generated using the output from a PCNN to create a template that was ultimately binarised into two distinct shapes. The segmented shapes were referred to as the *PCNN derived shapes*.

The PCNN derived shapes were used to create three feature sets with the purpose of comparing the effectiveness of the PCNN derived shapes against the handcrafted shapes. The feature sets were calculated using the mean and standard deviation of pixel intensities within each shape. The three feature sets were:

1. PCNN derived and handcrafted shapes;
2. PCNN derived shapes only; and
3. PCNN derived shapes and quadrants.

An investigation of the detection programs that used the PCNN derived shapes demonstrated a significant improvement in detection performance when compared to the handcrafted shapes described in Section 5.3.1 when tested on the menton, upper lip and sella landmarks. However, the results for the incisal upper incisor landmark suggest that the PCNN derived shapes were not as useful in evolving detection programs as there was no improvement in detection performance.

Ideally, we would like an approach for landmark detection which is entirely automated, from the creation of useful features to be used within a program, to the identification of landmarks within an image. Even for a domain expert, constructing shapes for the purposes of extracting useful features is not a trivial task. This is a compelling argument for the utility of an automated approach to the problem. We have gone a long way towards achieving this goal; the method described in this chapter has shown that the PCNN approach using automatically derived square shapes, is comparable to, or better than a handcrafted approach.

Chapter 7

Analysis of Evolved Programs

Previous work in Chapters 5 and 6 has established that our genetic programming approach to landmark detection was successful at evolving detection programs for several cephalometric landmarks of varying difficulty. However, one of the drawbacks of genetic programming solutions is that the evolved programs can become very complex and, as a result, difficult to understand. This makes it hard to “sell” genetic programming solutions because many domain experts are unhappy with black box solutions. To achieve a greater acceptance by domain experts requires that a program’s solution be better understood in order to appreciate its limitations and generality. The aim of this chapter is to determine whether any regularities are being discovered, and if so, what they are. We hope that by identifying regularities in evolved detection programs we will have a better understanding of the landmark detection process. The initial part of the investigation will focus on programs evolved to locate an easy landmark, the tip of the nose, followed by an analysis of programs evolved to locate a more difficult landmark, the sella landmark.

The analysis methodology will be to evolve simplified detection programs by limiting the function and terminal sets and to encourage small programs by including a size penalty in the fitness. The underlying algebraic expressions in the programs will be simplified and compared. The programs will be manually executed at key positions in the training and test images.

7.1 Genetic Programming Configuration

The genetic programming parameters to be used during training are the same as those used in Table 4.3 on page 72. The results presented in this chapter will be based on the best individual from each run for 80 evolutionary runs.

7.2 Fitness function

Previous work in Section 5.4.1 established that Equation 7.1 was successful as a measure of the program's performance when we know that only one landmark is located in an image.

$$\textit{fitness} = (1 - DR) \times 100, \text{ where} \quad (7.1)$$

DR is the detection rate.

To assist with the simplification of programs, we attempt to reduce the size of programs by favouring smaller trees on the proviso that detection rate is not compromised during an evolutionary run, i.e. we would like to remove extraneous segments of code not contributing to a program's performance. In Section 2.5.4 we described extraneous code segments not contributing to a program's performance as introns. A common approach used to reduce bloat is to incorporate parsimony pressure as part of the fitness measure.

The parsimony pressure is achieved by incorporating a second objective into the fitness function as a way of penalising large programs. The revised fitness function using parsimony pressure is shown in Equation 7.2. The second term provides the parsimony pressure, which divides the number of nodes in the program by the maximum number of nodes in a tree. The maximum number of nodes in a tree is dictated by the maximum depth of the tree. The maximum tree depth in our problem has been limited to nine, giving a maximum of $(2^9 - 1) = 511$ nodes in a full binary tree. To ensure the main objective of the fitness function is to evolve programs with good detection performance, the second term is multiplied by $\frac{1}{10}$. This limits the second term between $[0, 0.1]$. The first term is limited between $[0, 100]$. The weightings for each of the two terms ensure that detection performance is the primary

objective and program size is secondary.

$$fitness = (1 - DR) \times 100 + \frac{Program\ Size}{511} \times \frac{1}{10}, \text{ where} \quad (7.2)$$

DR is detection rate and
 $Program\ Size$ is the number of nodes in the program

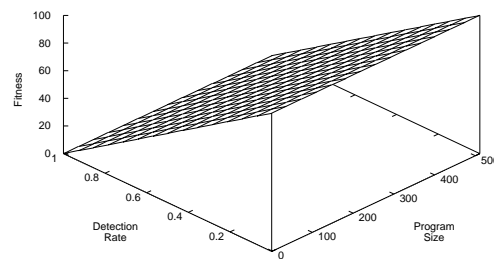


Figure 7.1: A graphical representation of the fitness function, as defined in Equation 7.2, for optimising detection rate and program size.

Figure 7.1 is a graphical representation of Equation 7.2 with the aim of minimising fitness. The graph illustrates that the easiest way for improving fitness is to increase detection rate. This is indicated by the larger gradient along the detection rate axis in comparison to the program size axis.

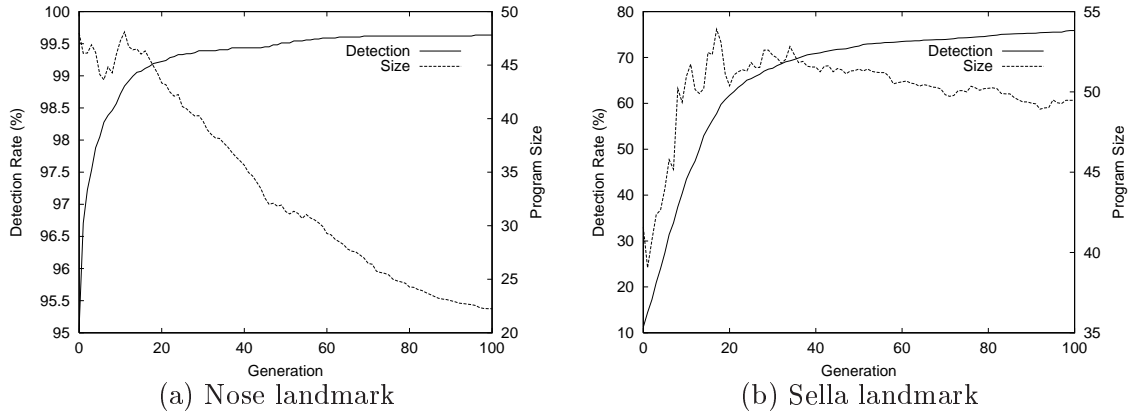


Figure 7.2: The change in detection performance and size of programs during an evolutionary run when using parsimony pressure to reduce program size. The detection rate and program size are an average calculated using the best individual from each generation for 80 evolutionary runs.

Although the termination criteria remain unchanged, early termination cannot be achieved because a program with a fitness score of zero is not possible, i.e. a program with 100% detection rate and a program size of zero nodes. Each evolutionary run is terminated at the end of 100 generations as per the previous work presented in this thesis. The graphs in Figure 7.2 show the improvement of average detection rate and the reduction in program size throughout the evolutionary process for both the nose and sella landmarks.

7.3 Terminal set

The terminals used for the purpose of this analysis are based on two terminal sets previously used in thesis that have produced programs with good detection performance. The terminal set used for the nose landmark consists of features using handcrafted shapes as defined in Section 5.2, whereas the sella landmark will use a terminal set composed of features using PCNN derived shapes and quadrants as defined in Section 6.10. Both terminal sets are shown in Figure 7.3, where each terminal set consists of the calculated mean and standard deviation of pixel intensities within each shape.

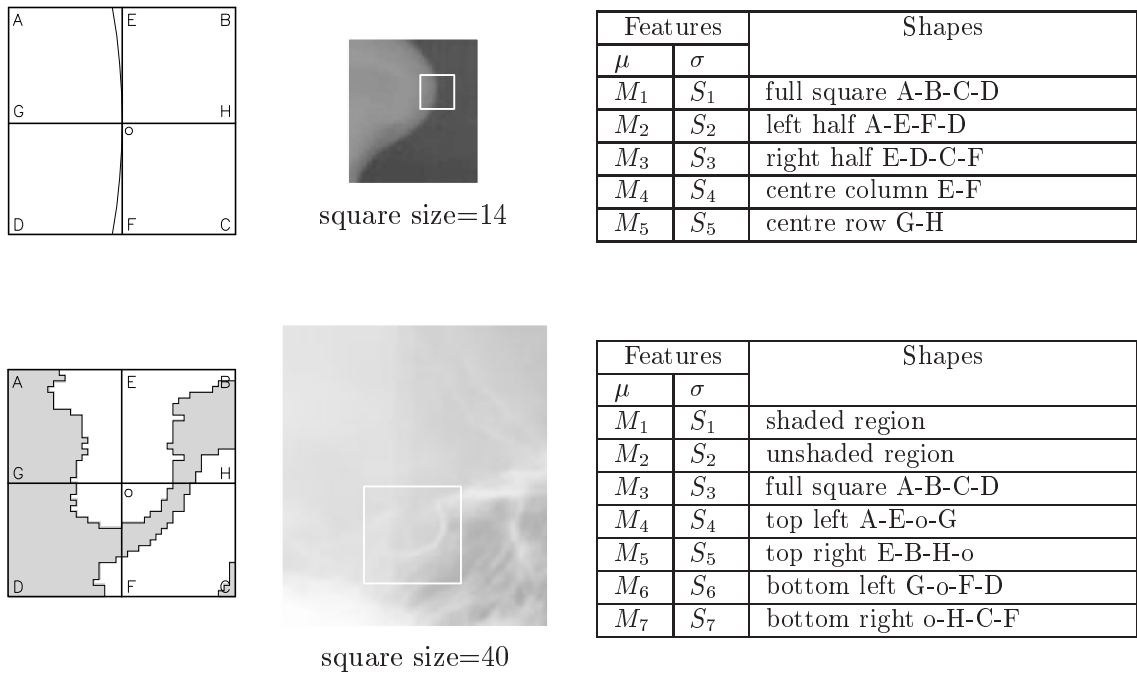


Figure 7.3: The diagrams in the left column depict the shapes used to extract the features for the nose and sella landmarks. The features consist of the mean and standard deviation calculated for each shape from grey level intensities. The corresponding picture in the middle column depicts the size of the input window, shown as the white square, relative to the image. Note: The nose image has had the contrast enhanced to improve the clarity of the soft tissue.

7.4 Function set

The analysis of the programs is restricted to function sets which will only evolve programs exhibiting the behaviour of linear functions. The reduced function sets will be limited to using the $\{+, -\}$ operators that are described in Section 4.3.2. The operators are divided into two function sets as shown in Table 7.1.

Case 1	+
Case 2	+, -

Table 7.1: Operators used to evolve programs that will exhibit the behaviour of linear functions.

7.5 Results

This section analyses detection programs for the nose and sella landmarks where the programs are limited to exhibiting the behaviour of a linear function. The following analyses are based on 80 evolutionary runs using the best individual at the end of 100 generations.

7.5.1 Nose landmark

7.5.1.1 Case 1: ‘+’ operator

The motivation for using the $\{+\}$ operator is to simplify programs so we can understand if there are any regularities captured when using the simplest operator. All the evolutionary runs evolved a best individual using only the S_5 terminal, i.e. the standard deviation of two rows of pixels centred within an input window, as shown in Equation 7.3. This program produced a detection performance of 65.9% (54/82). The results from training suggest that Equation 7.3 was unsuccessful at locating the position of landmarks when the nose was located near the edge of the image. A selection of images where the landmark is incorrectly identified is shown in Figure 7.4.

$$Output = S_5 \tag{7.3}$$

As described in Section 5.4.1, the landmark is located by moving an input window across the image and calculating the detection program’s output at each pixel location. The highest output is used for predicting the position of the landmark. If we compute the output when the input window is positioned on an area of constant brightness, such as the soft tissue or background, the program returns a low positive output, i.e. $Output$ is ≈ 0 . However, if the input window is positioned partially on a vertical or diagonal edge such as the soft-tissue/background edge, the program’s output will increase until the centre of the input window coincides with an edge. The reason is that S_5 is computed using the standard deviation of two rows of pixels and most variation of pixel intensities occurs when the input window is centred on either diagonal or vertical edges. But as mentioned previously, some of

the nose tips are located near the edge of the image that also contain a bright vertical band as shown in Figure 7.4. So not only is the output high on the soft-tissue/background edge but also on the bright vertical band.

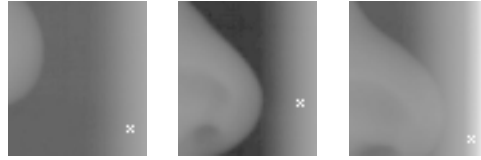


Figure 7.4: Images where the nose landmarks are incorrectly identified. In all three cases, the landmarks are located near a bright vertical band that is an artifact near the edge of an X-ray. All images have been enhanced using a logarithmic LUT to improve the clarity of the soft tissue.

The analysis described above is consistent with the graph shown in Figure 7.5. The surface plot represents the output of the detection program, Equation 7.3, applied to a greyscale image. This shows that when the input window is positioned on an area of constant brightness the output of the program is ≈ 0 . The output increases whenever the input window is positioned on an edge and approaches a maximum when centred on the soft-tissue/background edge. The ridge shown in Figure 7.5 correlates to the soft-tissue/background edge. The highest peak in Figure 7.5 is used to predict the position of the landmark and in this case the program has predicted the landmark with a detection error of (2, 3) pixels.

The methodology is very promising that even with the extremely simplified situation we get a program that is somewhat accurate whose behaviour we can understand.

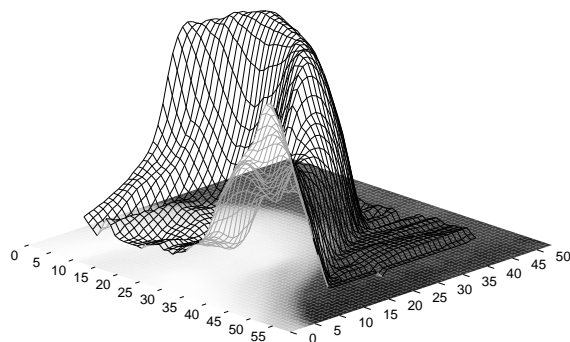


Figure 7.5: The graph illustrates the output from the fittest individual, S_5 , when only the $\{+\}$ operator is available for selection. The output, which is represented as the surface plot, is superimposed on the greyscale image.

7.5.1.2 Case 2: ‘+ , -’ operators

The previous section demonstrated that a function set consisting of the $\{+\}$ operator alone was not able to evolve a detection program for locating all nose landmarks within the image data set. With only the $\{+\}$ operator available for selection, the evolutionary process was only able to evolve a fittest individual that used the S_5 terminal. We know from previous work in Section 5.5 that having $\{+, -\}$ operators available for selection allows the evolutionary process to evolve programs that can locate the nose landmark with a 100% detection rate. The aim of this section is to analyse programs that have the use of $\{+, -\}$ operators to determine: (a) how these detection programs locate the nose landmark and (b) if there are any underlying algorithms learnt as a result of the evolutionary process. As noted in Section 7.4, a function set consisting of $\{+, -\}$ operators will only evolve programs exhibiting the behaviour of linear functions. This is analogous to evolving a coefficient for each terminal, α_i and β_i , in Equation 7.4. Note that only integer coefficients can be evolved for each terminal.

$$\begin{aligned}
 \text{Output} &= \alpha_1 M_1 + \beta_1 S_1 + \alpha_2 M_2 + \beta_2 S_2 + \cdots + \alpha_5 M_5 + \beta_5 S_5 & (7.4) \\
 &= \sum_{i=1}^5 (\alpha_i M_i + \beta_i S_i), \text{ where} \\
 &\alpha_i \text{ and } \beta_i \text{ are integers, and } i \text{ corresponds to the } i\text{th shape}
 \end{aligned}$$

The evolutionary process using the fitness function in Equation 7.2 was able to produce programs with an average detection rate of 99.6%. Comparing this result to a similar experiment in Section 5.5 that does not include parsimony pressure when calculating fitness suggests that using a penalty to reduce extraneous code does not influence detection performance; both experiments are similar in terms of function set, terminal set and the method used to locate the landmark. The comparison was based on a two-sample t test to determine if there was a difference in mean detection rate between two samples of programs using: (a) detection rate as the fitness metric and (b) detection rate and parsimony pressure as the fitness metric; A large p -value of 0.87 indicated that there was no evidence to suggest that the average detection performance had changed. However, a two-sample t test comparing the size

of the programs between the two fitness functions produced a p -value of 0.000 indicating that parsimony pressure had significantly decreased the size of the programs. Using parsimony pressure, the average program size has decreased from 56.8 nodes to 22.3 nodes. The linear functions shown in Equations 7.5-7.8 are derived from four of the fittest programs taken from 80 evolutionary runs when using parsimony pressure in the fitness metric.

$$\text{Output} = M_1 - S_2 - S_3 - 2M_4 + M_5 \quad (7.5)$$

$$\text{Output} = M_2 - 2S_2 - 2M_4 + M_5 \quad (7.6)$$

$$\text{Output} = 2M_1 - S_1 - 2M_4 - S_4 \quad (7.7)$$

$$\text{Output} = M_2 - 2S_2 - 2M_4 + M_5 \quad (7.8)$$

Table 7.2 is a list of linear functions – these functions account for 43/80 of the evolutionary runs – that have regularly occurred at the end of the evolutionary process. The analysis reveals that the evolutionary process has not evolved identical programs but has learnt an equivalent function. A selection of programs that produced an equivalent function are shown in Figure 7.6. To determine if they are equivalent, the relationships of Equations 7.10 and 7.11 are substituted into each function; the reason for the substitution is explained on page 168.

Frequency	Program	Detection Rate
13/80	$3M_2 - 4S_2 + M_3 - 4M_4$	100% (82/82)
13/80	$3M_2 - 2S_2 + M_3 - 4M_4 - 2S_4$	100% (82/82)
7/80	$M_2 - 2S_2 - M_3 - 2S_3 + 2S_5$	98.7% (81/82)
6/80	$M_2 - S_2 + M_3 - S_3 - 2M_4$	100% (82/82)
4/80	$-S_1 + M_2 + M_3 - 2M_4 - S_4$	100% (82/82)

Table 7.2: Frequency that a linear function has occurred as a result of the evolutionary process. Each function is derived using the best individual from each run for 80 evolutionary runs. This table has been restricted to functions that have commonly occurred from the 80 evolutionary runs.

-
1. $(- (- M_5 S_2) (+ M_4 (+ S_2 (- M_4 M_2))))$
 2. $(- (+ (- M_5 (+ M_4 (+ M_4 (- S_2 M_2)))) S_5) (+ M_1 S_2))$
 3. $(+ (+ (+ (- (- S_5 M_4) S_2) (- (- (- S_2 (+ S_2 M_3)) M_4) (+ S_1 S_5))) (+ M_4 M_3)) (- (- (+ (+ M_4 M_5) (+ (+ M_5 S_1) (+ M_3 M_1))) (+ M_3 M_1)) (+ (+ S_5 (+ (+ M_4 M_3) M_4)) (- M_2 (+ (- M_2 S_2) (+ M_1 M_2))))))$
-

Figure 7.6: The above three evolved programs are significantly different in terms of tree structures and genetic material used, however, each of the programs is equivalent to $3M_2 - 4S_2 + M_3 - 4M_4$. A detection rate of 100% (82/82) was achieved by each of these programs.

Analysis of an individual program

The following analysis describes how Equation 7.5 is used to predict the position of the landmark based on the value of features at six positions across two images. Each position defined in Table 7.3 is indicative of regular image patterns that occur within the image data set. The positions include soft tissue (1), background (2), a soft-tissue/background edge (3) and two examples where the input window is centred on the nose landmark (4, 5). The last position is related to the tip of nose located near the edge of an image, i.e. the tip of the nose is slightly obscured by the bright vertical band (6).

If we evaluate the program when the input window is located on an area of constant brightness, such as the soft-tissue or background, the output of the program is ≈ 0 . Intuitively we know that when the input window is positioned on an area of constant brightness, the values of: (a) features calculated using the mean of grey level intensities within each shape, M_i , will be approximately the same and (b) features calculated using the standard deviation of grey level intensities within each shape, S_i , will be ≈ 0 . A sample calculation is shown in Equation 7.9 when the input window is located on an area of constant brightness. We have chosen an arbitrary value, x , to describe the average grey level intensity for each feature. This is because the value of each feature, calculated using the average from grey level intensities within each shape, is approximately the same. The calculation demonstrates that when the input window is located on an area of constant brightness the output is \approx zero.

$$\begin{aligned}
Output &= M_1 - S_2 - S_3 - 2M_4 + M_5 & (7.9) \\
&\approx x - 0 - 0 - 2x + x \\
&\approx 0
\end{aligned}$$

where x is an arbitrary value of feature M_i

If we evaluate the program when the input window is centred on a diagonal soft-tissue/background edge, as shown as position 3 in Table 7.3, the program's output is negative. When the input window is located on the tip of the nose, the program produces a high output in comparison to the previous positions. If we compare both outputs when the input window is located on the soft-tissue/background edge (refer to positions 3 and 4 in Table 7.3) and treat each term in the equation as a separate component, we observe that the most significant component for varying the output is contributed by M_4 . If the input window moves either side of the soft-tissue/boundary edge, the value of either S_2 , the standard deviation of the left half, or S_3 , the standard deviation of the right half, will decrease the output because of the negative coefficients.

When the highest output is used to predict the position of the landmark, the detection errors are (1, -1) and (1, 0) pixels with respect to the known positions for the images shown in Table 7.3.

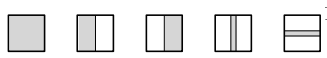
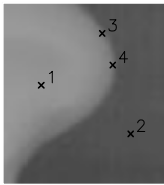




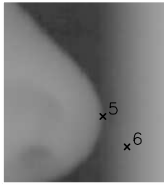


	Position		$M_1 - S_2 - S_3 - 2M_4 + M_5 = Output$
	1		$23.3 - 0.9 - 1.3 - 45.8 + 22.3 = -2.4$
	2		$4.5 - 0.5 - 0.5 - 8.6 + 4.8 = -0.3$
	3		$11.6 - 5.1 - 3.3 - 22.4 + 12.0 = -7.2$
	4		$10.2 - 3.9 - 0.3 - 14.2 + 10.8 = 2.6$
	Highest $\epsilon = (1, -1)$		$9.2 - 4.7 - 0.2 - 10.5 + 9.8 = 3.6$
	5		$10.1 - 3.1 - 1.4 - 15.5 + 10.6 = 0.7$
	6		$6.8 - 1.3 - 2.7 - 11.0 + 6.7 = -1.5$
	Highest $\epsilon = (1, 0)$		$9.5 - 3.9 - 1.8 - 10.2 + 10.0 = -3.6$

Table 7.3: Sample evaluations of a linear function, $M_1 - S_2 - S_3 - 2M_4 + M_5$, when applied to six different positions across two images.

The analysis described above is consistent with the graph shown in Figure 7.7. The graph in Figure 7.7 is the output of a detection program, Equation 7.5, that has been applied to a grayscale image. A graphical representation of Equations 7.6-7.8 applied to the same image show landscapes similar to the one shown in Figure 7.7.

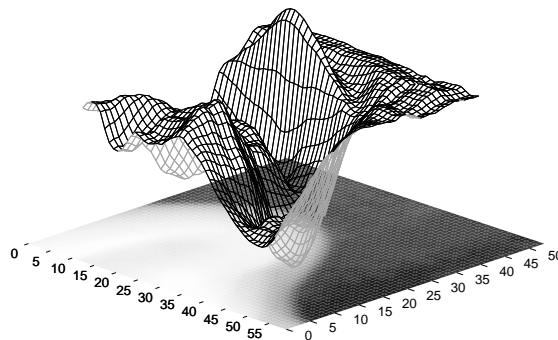


Figure 7.7: The graph illustrates the output from a fit individual, $M_1 - S_2 - S_3 - 2M_4 + M_5$, when only the $\{+, -\}$ operators are available for selection. The output of the program, which is represented as the surface plot, is superimposed on the grayscale image.

¹Each shaded region shows the pixels that are used in the statistical operation for calculating the feature.

Analysis of regularly occurring patterns across programs

We have analysed a program and have shown that the program's output is not ad-hoc and that the underlying evolved algorithm is reasonable for detecting the tip of the nose. In this section we investigate whether there are any regularities learnt through the evolutionary process that are consistent among the evolutionary runs.

To reduce the complexity of the analysis we remove the redundant terminals in the terminal set. For example, we know the sum of the average grey level intensity of the left and right halves of the input window is related to the average grey level intensity of the entire input window (refer to Equation 7.10). We also confirmed experimentally that if M_1 is substituted in place of M_5 then the detection performance of the evolved programs is not compromised (refer to Equation 7.11). This suggests that the nose landmark training data for both M_5 and M_1 terminals are similar. The programs' complexity is reduced by eliminating the terminals of M_1 and M_5 by substituting the relationships of Equations 7.10 and 7.11 into the evolved programs.

$$M_1 = \frac{1}{2}(M_2 + M_3) \quad (7.10)$$

$$M_1 \approx M_5 \quad (7.11)$$

An analysis of the best program from each evolutionary run reveals a perfect correlation, using best subsets regression [132], between the terminal coefficients of α_1 and the terminal coefficients of α_2 , α_3 , α_4 and α_5 . A multiple linear regression indicates that the sum of all these terminal coefficients, i.e. coefficients of terminals based on the average grey level intensities of the various shapes, is zero. The relationship is shown in Equation 7.12. The significance of this relationship is that when the input window is located on an area of constant brightness, the program's output is \approx equal to 0. The reason why this occurs is described on page 165.

We have established that programs produce an output of ≈ 0 when the input window is located on an area of constant brightness, but how is the program's output manipulated so

that non-landmark positions (such as on an edge) are differentiated from a landmark located on an edge? The bar chart shown in Figure 7.8 indicates that the evolutionary process selects a combination of features calculated from both the average and standard deviation of grey level intensities within each shape. Generally speaking, we have found that the evolutionary process generates detection programs that can detect the majority of landmarks with features using the calculated means of the grey level intensities within each shape, and that the performance is further enhanced with features that have been calculated using the standard deviations.

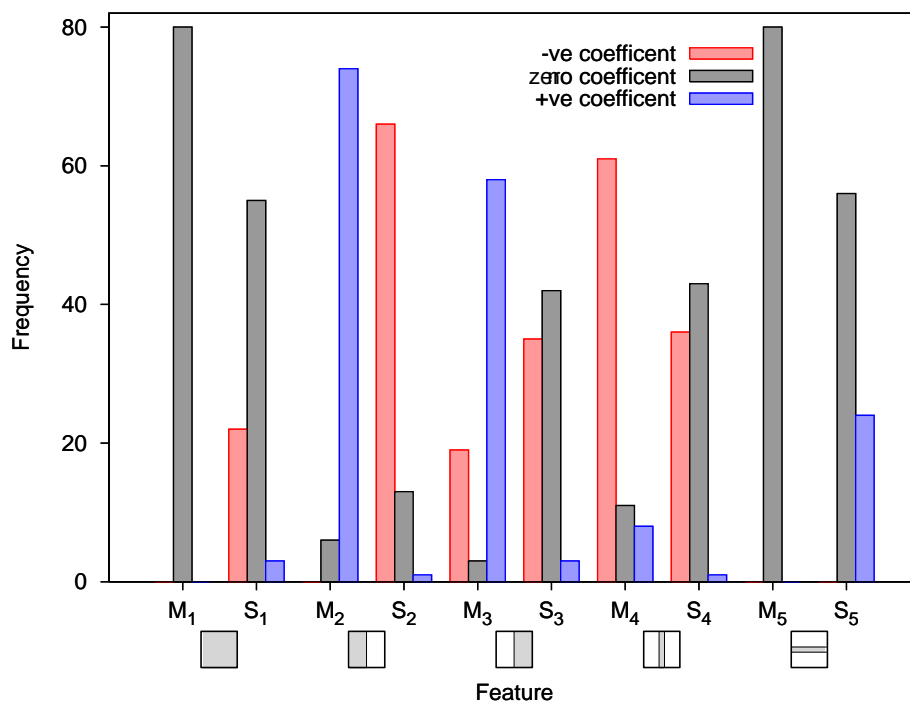


Figure 7.8: Frequency that a feature's coefficient, α_i or β_i in Equation 7.4, is negative, positive or zero (i.e. not used in a program). The analysis uses the best program at the end of 80 evolutionary runs. Each program was substituted with Equations 7.10 and 7.11 resulting in a zero coefficient for features M_1 and M_5 .

We shall start by analysing how features calculated using the mean are used for locating landmarks. If we examine the coefficients that regularly occur in Figure 7.8, we observe that the coefficients of M_2 and M_3 are generally positive and the coefficient of M_4 is generally negative. On the occasions when the coefficient of M_3 is not positive and the coefficient of M_4 is not negative, the detection performance of the program was less than 100%. This is

shown in Figures 7.9(a) and 7.9(b).

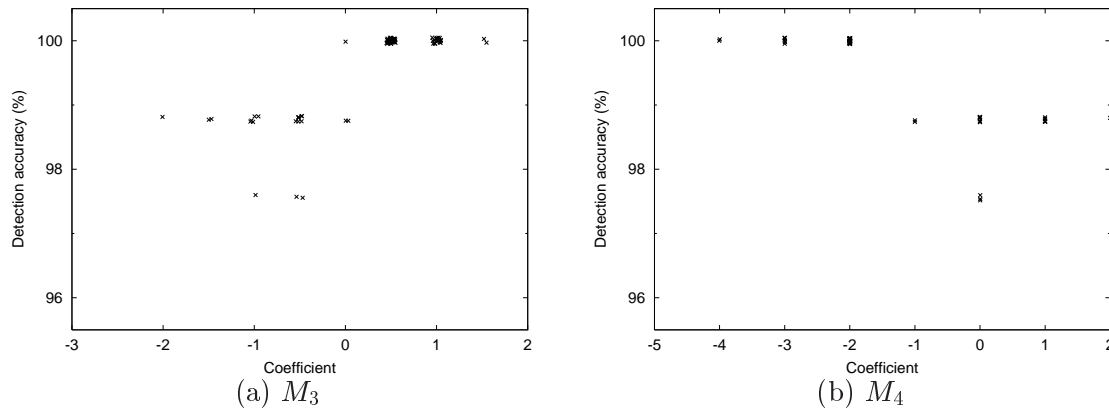


Figure 7.9: Graphs (a) and (b) show the relationship between the coefficients of M_3 and M_4 versus detection performance. The data has been jittered so that the density of points at a given point are visible.

This re-occurring pattern is interesting as M_2 and M_3 are the means of the left and right halves of the input window respectively, and M_4 is the mean of the two centre columns. Although the combination of M_2 and M_3 are important, M_4 is able to penalise the program's output when the input window is located on the soft-tissue/background edge, above or below the tip of the nose. The penalty is associated with the negative coefficient of M_4 that decreases the program's output when values of M_4 are high relative to M_2 and M_3 . When the input window is located on the tip of the nose, the penalty of M_4 is decreased because the mean is associated with pixels that have low greyscale values.

If we analyse in more detail how features using the standard deviation work, we observe that S_2 , S_3 and S_4 – these are features that regularly produce a non-zero coefficient – give a higher output when the input window is positioned on a diagonal edge compared to the tip of the nose. Because we want to penalise the program's output when the input window is located on a diagonal edge, the features are multiplied by a negative coefficient and so the program's output is reduced. However, if the input window is located on the tip of the nose, the standard deviation values of S_2 , S_3 and S_4 are negligible and the program's output is not reduced. It appears that the features computed using standard deviation are used to differentiate the ideal position from other positions that contain clutter or areas that are not

constant brightness. However, these features only differentiate the ideal position from clutter and do not highlight the position of the landmark against areas of constant brightness.

If we analyse the frequency that a feature's coefficient occurs using Figure 7.8, we observe that useful features, i.e. features that regularly produce either a positive or negative coefficient, are those that are calculated using shapes within an input window that coincide with the shape of the nose tip. For example, the left and right halves of the input window are approximately the shape of the soft tissue and background respectively when the input window is centred on the tip of the nose. Another shape regularly used was the two central columns.

The shapes used to calculate S_1 , the full square, and S_5 , two central rows, do not match the profile of the nose and as a result a high standard deviation is produced when the input window is positioned on the tip of the nose. These features of S_1 and S_5 , do not occur in 43/80 programs and have the same magnitude coefficient but opposite signs on an additional 9/80 programs. The significance of this is that the similarity between the two shapes will result in similar feature values and thereby have a negligible influence on the output. Shapes that correlate to the profile of the nose are S_2 , standard deviation of the left half, and S_3 , standard deviation of the right half. The frequency of occurrence of S_2 and S_3 is 67/80 and 42/80 respectively. The reason for the lower occurrence of terminal S_3 in programs is that when the nose is partially obstructed by the edge of the image, the output would be penalised when the input window is located on the tip of the nose, thus thereby reducing the output at the landmark's location. Terminal S_2 is more effective at discriminating false alarms in these images and so appears more frequently.

Another regularity common with the best individual of each evolutionary run is that the sum of the coefficients of terminals calculated using standard deviation is negative. This relationship is shown in Equation 7.13. The significance of this is if the input window is positioned on a cluttered scene, such as the soft-tissue/background boundary, the computed output from a program will be reduced as a result of the accumulated penalty from each standard deviation feature.

$$\sum_{i=1}^5 \alpha_i = 0 \quad (7.12)$$

$$\sum_{i=2}^4 \beta_i < 0 \quad (7.13)$$

where α_i and β_i are the coefficients of M_i and S_i respectively.

The significance of the above analysis is that the most important features were those derived from shapes that match the nose's profile when the input window is centred on the tip of the nose. Features calculated using shapes that do not match the nose's profile were either not selected during the evolutionary process or were made redundant by a similar feature. This is consistent with the results of Section 5.6 in which it was found that handcrafted shapes gave better detection performance than pixel based features that were not specific to the shape of the landmark.

7.5.2 Sella landmark

The previous section analysed programs used to detect a simple landmark, the tip of the nose, and demonstrated that the evolved programs were not ad-hoc and in fact the evolutionary process learnt underlying regularities that were consistent between the evolutionary runs. The analysis in this section is for a more difficult detection problem, the sella landmark. The increased difficulty is in terms of the variation in shape of the sella region as well as the amount of background clutter.

7.5.2.1 Case 2: '+, -' operators

The aim of this section is to analyse detection programs for a more difficult detection problem when only the $\{+, -\}$ operators are available for selection. As mentioned previously, a function set consisting of only $\{+, -\}$ operators will only evolve a linear function which is analogous to evolving a coefficient for each terminal, α_i and β_i , in Equation 7.14.

$$\begin{aligned}
Output &= \alpha_1 M_1 + \beta_1 S_1 + \alpha_2 M_2 + \beta_2 S_2 + \cdots + \alpha_7 M_7 + \beta_7 S_7 & (7.14) \\
&= \sum_{i=1}^7 (\alpha_i M_i + \beta_i S_i), \text{ where} \\
&\alpha_i \text{ and } \beta_i \text{ are integers, and } i \text{ corresponds to the } i\text{th shape}
\end{aligned}$$

An intermediate analysis of programs that were evolved from a feature set calculated using PCNN derived shapes

As an intermediate step to understanding programs that were evolved using the feature set outlined in Figure 7.3, this section will investigate programs when only the first four features are available for selection, i.e. M_1 , S_1 , M_2 and S_2 . The features are based on the calculated mean and standard deviation for each of the two PCNN derived shapes. The process to extract the PCNN derived shapes is described in Chapter 6.

The evolutionary process using the fitness function in Equation 7.2 produced an average detection rate of 70.3%. Although the best individual from each of the 80 evolutionary runs produced vastly different programs, upon simplification it was established that each of the programs are variants of two linear functions. The two functions are shown in Equations 7.15 and 7.16, which produced similar detection rates of 70.7% and 69.5% respectively. Both functions shown in Equations 7.15 and 7.16 are similar with the exception of the coefficient for S_2 .

$$\begin{aligned}
Output &= 5M_1 - 5S_1 - 5M_2 - 2S_2 & (7.15) \\
&= M_1 - S_1 - M_2 - 0.4S_2 \text{ (equivalent fitness function)}
\end{aligned}$$

$$\begin{aligned}
Output &= 3M_1 - 3S_1 - 3M_2 - S_2 & (7.16) \\
&\approx M_1 - S_1 - M_2 - 0.3S_2 \text{ (equivalent fitness function)}
\end{aligned}$$

Analysis of an individual program

The following analysis describes how Equation 7.15 is used to predict the position of a

landmark based on the value of features at six different positions across two images. Each of the positions shown in Table 7.4 are indicative of regular patterns that occur within the image data set. The positions include an area that has similar pixel intensity values (1), edge of bone with some background clutter (3, 4, 5) and two examples where the input window is centred on the sella landmark (6, 7).

If we evaluate the program when the input window is located on an area of constant brightness, i.e. an area where the input window contains pixels having similar pixel intensity values, the output of the program is ≈ 0 . The reason for this was described on page 165, but in brief, the values of: (a) features calculated using mean, M_i , are approximately the same and (b) features calculated using standard deviation, S_i , are ≈ 0 . An analysis of the output at the first position in Table 7.4 indicates that when the input window is located on an area that visually appears to contain pixels with constant brightness, the output is close to zero relative to the output of the other five positions.

If we treat the second, third and fourth positions from Table 7.15 as a similar type of pattern, i.e. the input window containing either bone or background clutter, we observe the output is significantly lower compared to the first position. The components calculated from standard deviation features, i.e. S_1 and S_2 , have increased because the PCNN derived shapes used to calculate values of S_1 and S_2 do not coincide with any of the cluttered regions. The output is penalised because the coefficients of both S_1 and S_2 are negative thereby reducing the program's output at the landmark's location.

If we analyse position five, i.e. when the input window is centred on the known position of the landmark, we achieve an output that is higher than the previous four positions. This is partly due to the relationship between components M_1 and M_2 and the lower values of S_1 and S_2 . If we subtract $5M_1$, the calculated mean of pixel intensity values depicted as brighter pixels when centred on the landmark, from $5M_2$, the calculated mean of slightly darker pixels, we get a value of $1111 - 1050 = 51$. Even though the values for both features calculated using standard deviation are high, they are low relative to the other positions that are located on a cluttered scene. The detection error was (1, -2) pixels with respect to the known position when using the highest output to predict the position of the landmark.

The bottom image in Table 7.4 (Table 7.4b) is an example where the program has in-

correctly located the landmark and has produced a detection error of (29, 14) pixels. The reason for the incorrect prediction is a combination of two factors. The first factor is the low contrast between the semi-circular region (or region depicted as a saddle) and background pixels. This has produced a low value when subtracting the values $5M_2$ from $5M_1$, when the input window is located over the position of the landmark. Secondly, the standard deviations of both shapes are high relative to the analysis at position five in the top image of Table 7.4. The high values of S_1 and S_2 are attributed to the generality of the PCNN derived shapes used to capture the large variations of the semi-circular region that encompass the sella landmark. It is reasonable to expect that if these shapes better matched the regions of interest then this would have produced a lower standard deviation potentially increasing the program's output at the desired location.















			    ¹		
	Position		$5M_1 - 5S_1 - 5M_2 - 2S_2 = Output$		
 <p>(a)</p>	1		$1142 - 10.3 - 1135 - 2.8 = -6.1$		
	2		$1188 - 54.0 - 1172 - 18.5 = -55.5$		
	3		$986 - 68.8 - 1014 - 32.3 = -129.1$		
	4		$1047 - 72.8 - 1041 - 29.9 = -96.7$		
	5		$1111 - 28.8 - 1050 - 20.3 = 11.9$		
	Highest	$\epsilon = (1, -2)$		$1111 - 27.4 - 1051 - 19.2 = 13.4$	
 <p>(b)</p>	6		$1079 - 34.3 - 1037 - 19.3 = -11.6$		
	Highest	$\epsilon = (29, 14)$		$986 - 55.8 - 869 - 39.7 = 21.5$	

Table 7.4: Sample evaluations of a linear function, $5M_1 - 5S_1 - 5M_2 - 2S_2$, when applied to six different positions across two images.

To summarise this analysis:

¹Each shaded region represents the pixels that are used in the statistical operation for calculating the feature value.

- Features M_1 and M_2 assist with differentiating between the known position and areas where the input window contains pixel intensity values of constant brightness. The distinction is made by a positive output of $\Delta M_{1,2}$, where $\Delta M_{1,2} = 5M_2 - 5M_1$.
- Features M_1 and M_2 combined with S_1 and S_2 are used to differentiate between cluttered scenes and the known position.

The above analysis shows how each component within the linear function is able to manipulate the output for the purpose of predicting the position of the sella landmark. As mentioned on page 173, the programs are variants of two linear functions of which are similar. This demonstrates that although each evolutionary run evolves a significantly different program, the evolutionary process has captured an underlying regularity consistent across each evolutionary run. Another regularity captured by the evolutionary process is that the sum of the coefficients of features calculated using the mean, equates to zero. This is shown in Equation 7.17 and is consistent with the finding for programs that were evolved to detect the nose landmark. The coefficients of features calculated using standard deviation are always negative and decrease the program's output when the input window is located on a cluttered background.

$$\sum_{i=1}^2 \alpha_i = 0 \quad (7.17)$$

$$\beta_i < 0, \text{ where } i=1, 2 \quad (7.18)$$

where α_i and β_i are the coefficients of M_i and S_i respectively.

An analysis of programs that were evolved from a feature set calculated using PCNN derived shapes and quadrants

The previous section analysed the functionality and determined if there were any underlying regularities captured by the evolved detection programs when only four features, based on the two PCNN derived shapes, were available for selection. The aim of this section is to incorporate the additional features based on the quadrants, i.e. analyse programs where

the features from Figure 7.3 are available for selection, and then determine if there are any regularities captured as part of the evolutionary process. The features analysed in this section are the features used in Section 6.3.7.

The evolutionary process using the fitness function in Equation 7.2 produced an average detection rate of 75.9%. This result is significantly higher than the previous result when only the four features, i.e. M_1 , S_1 , M_2 and S_2 , were available for selection. This indicates that at least one of the other features were used to improve the detection performance of the evolved programs. The linear functions shown in Equations 7.19-7.22 are derived from four of the fittest programs taken from 80 evolutionary runs. The best evolutionary run achieved a detection rate of 84.1%.

$$\text{Output} = 5M_1 - 2M_2 - S_2 - 2M_3 - 3S_3 - 2S_4 + S_5 - M_6 + S_7 \quad (7.19)$$

$$\begin{aligned} \text{Output} &= 6M_1 - 2S_1 - 5M_2 - 3S_2 - M_3 - 3S_4 + M_5 + S_5 \\ &\quad - 2M_6 + S_6 + M_7 + S_7 \end{aligned} \quad (7.20)$$

$$\text{Output} = 3M_1 - 3S_1 - M_2 - 4S_2 - M_3 + 4S_3 - S_4 - M_6 + S_7 \quad (7.21)$$

$$\text{Output} = 6M_1 - S_1 - 5M_3 - 3S_3 - 2S_4 + S_5 - 2M_6 + M_7 + S_7 \quad (7.22)$$

Analysis of an Individual Program

Previous work on page 173 analysed how the position of a landmark is predicted based on the value of features at seven positions across two images. The positions were indicative of regular patterns that occur within the image data set. The analysis demonstrated that if the input window is located on an area of constant brightness or background clutter then the program's output will be approximately zero or negative respectively. The rationale of the program's output is the sum of each feature's coefficient calculated using mean, as described by Equation 7.15, is equal to zero. Likewise, the program's output is generally negative when the input window is positioned over cluttered background. This is shown by the numerical analysis for position 1 and positions 2-4 respectively in Table 7.5.

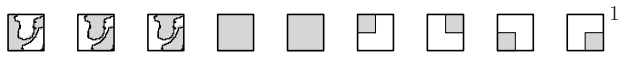









	Position		 ¹	$5M_1 - 2M_2 - S_2 - 2M_3 - 3S_3 - 2S_4 + S_5 - M_6 + S_7 = Output$
 (a)	1			1142- 454 - 1.4- 455 - 5.6- 1.3+ 1.0- 230 + 1.4= -2.9
	2			1188- 469 - 9.2- 472 -30.3- 5.2+ 1.8- 247 + 7.3=-35.6
	3			986- 406 -16.2- 401 -45.7-31.3+16.3- 188 + 9.7=-76.2
	4			1047- 417 -14.9- 418 -44.3-12.2+ 7.1- 202 +12.9=-41.4
	5			1111- 420 -10.2- 431 -31.2- 6.7+ 6.6- 222 +11.0= 7.5
	Highest $\epsilon = (1, -2)$			1111- 420 - 9.6- 431 -30.0- 7.1+ 6.5- 221 +10.6= 9.4
 (b)	6			1079- 415 - 9.6- 422 -28.5- 7.0+ 5.0- 213 +11.8= 0.7
	7			986- 348 -19.9- 369 -60.6-27.3+11.1- 193 +23.1= 2.4
	Highest $\epsilon = (1, -3)$			1084- 416 - 8.9- 424 -26.3- 6.8+ 5.2- 215 +11.6= 3.8

Table 7.5: Sample evaluations of a linear function, $5M_1 - 2M_2 - S_2 - 2M_3 - 3S_3 - 2S_4 + S_5 - M_6 + S_7$, when applied to seven different positions across two images.

Rather than repeat the analysis on an image where the landmark’s location was previously correctly identified, the analysis in this section will be applied to an image where the landmark’s location was previously incorrectly identified (refer to the bottom image in Table 7.4). The bottom image in Table 7.5 is an example where the landmark is now located within the allowable tolerance. Position six in Table 7.5 is the location of the known position of the sella landmark and position seven is located in a cluttered background and centred on the position that was previously recorded as a false alarm – the false alarm was a result of applying Equation 7.15 to the image.

The sum of all components within the linear function that consist of features calculated using mean for positions six and seven in Table 7.5 are 29 (1079-415-422-213) and 76 (986-348-369-193) respectively. The value at position seven is high relative to position six because the highest output should be located at the known position of the landmark. However, the features calculated using standard deviation reduce the program’s output at position seven

¹Each shaded region represents the pixels that are used in the statistical operation for calculating the feature value.

relative to position six – the sum of all components within the linear function that consists of features calculated using standard deviation for positions six and seven in Table 7.5 are -28.3 ($-9.6-28.5-7+5+11.8$) and -73.6 ($-19.9-60.6-27.3+11.1+23.1$) respectively. Although the highest output does not coincide with the known position, the detection error is $(1, -3)$ which is within the allowable tolerance for our detection problem.

A comparison of detection performance shows that Equation 7.19, a linear function evolved from fourteen features, has increased the detection performance from 70.7% ($58/82$) to 84.1% ($69/82$) with respect to Equation 7.15, a linear function evolved from four features. An analysis of these results show that Equation 7.19 has improved detection performance by correctly identifying the sella landmark in an additional thirteen images. It is worth noting that although the way in which features used by both linear functions are different, the functions have correctly located the landmark in identical positions in 47.6% ($39/82$) of the images. For example, refer to the top image in Tables 7.4 and 7.5 where the landmark is correctly identified by two different programs with a detection error of $(1, -2)$ pixels. This result is not entirely unexpected because the graphs in the top row in Figure 7.10 produce similar surface plots which are the result of two different programs applied to the same image. The surface plot is the output from a program that has been applied to each position within the image. A similar surface plot occurs when both programs are applied to the bottom image in Figure 7.10, except the height of the peak that was previously highest – refer to View A in Image B(a) – has been reduced by the use of additional features – refer to View A in Image B(b). The highest peak in View A of each image is circled.

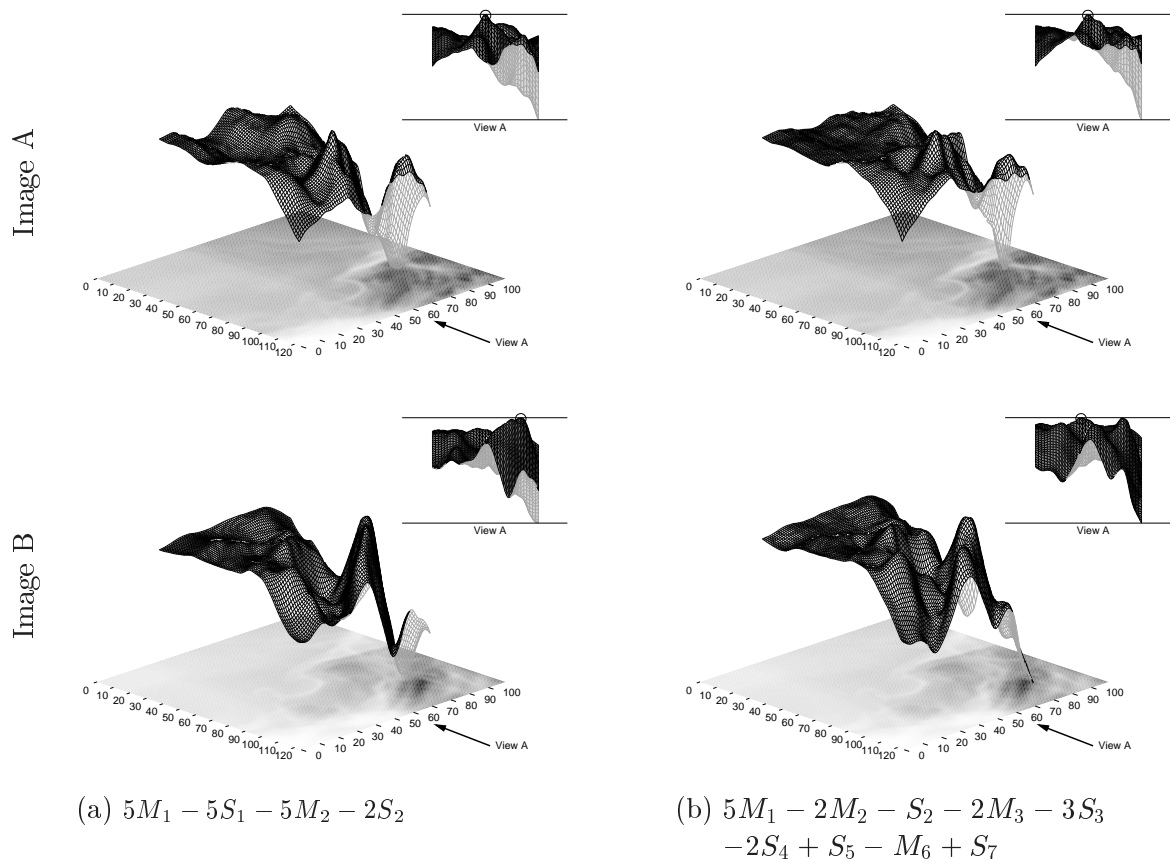


Figure 7.10: Graphs (a) and (b) illustrate the output from a fit individual when four and fourteen features are available for selection respectively. In both cases, only the ‘+,’ ‘-’ operators are available. The output of the program, which is represented as the surface plot, is superimposed on the greyscale image.

It is worth stating that although the evolutionary process evolves a diverse range of programs, some of the evolutionary runs learn an equivalent function. A sample of programs that produce an equivalent function are shown in Figure 7.11. To determine if they are equivalent, the relationship of Equation 7.23 is substituted into each function; the reason for the substitution is explained on page 181. This indicates that although the genetic programming method evolves a program that appears somewhat ad-hoc, it is not uncommon for the genetic programming method to learn an underlying regularity, and in this case an identical function, from different evolutionary runs.

1	(+ (- (+ S ₇ (- (- M ₁ S ₁) S ₃)) (- (- M ₂ S ₁) (- (- (- M ₁ S ₁) S ₃) M ₂))) (- (- M ₃ S ₃) M ₂)))
2	(+ (- (+ (- S ₂ (- (+ (+ M ₂ S ₆) (+ M ₄ S ₁)) (+ (- (- M ₁ S ₃) M ₂) (+ (- S ₆ M ₂) M ₄)))) (+ M ₃ S ₇)) (+ (+ S ₂ (- S ₃ (- M ₁ S ₃))) M ₃)) M ₃)
3	(+ (+ (+ M ₃ S ₆) (+ M ₃ (+ (- S ₇ (- (- M ₂ M ₅) S ₁)) (- (- (- M ₁ S ₁) S ₃) (+ (+ S ₆ M ₃) M ₂)))))) (- (- (- (- M ₁ S ₁) S ₃) S ₃) (+ M ₅ M ₂)))

Figure 7.11: The above three evolved programs are significantly different in terms of tree structures and genetic material used, however, each of the programs is equivalent to $5M_1 - 2S_1 - 5M_2 - 2S_3 - 2S_7$. A detection rate of 76.8% (63/82) was achieved for each of these programs.

Analysis of regularly occurring patterns across programs

We have analysed two programs used to locate the sella landmark where each program was evolved using two different feature sets. The analysis showed that the program's output is not ad-hoc and that the underlying algorithm is reasonable for detecting the sella landmark. In this section, we shall perform a similar type of analysis that was conducted for nose landmark, to determine if there are any regularities captured through the evolutionary process that are consistent amongst the evolutionary runs.

To reduce the complexity of this analysis we remove the number of redundant terminals that are available in the terminal set. For example, we know the sum of the average grey level intensity of M_1 and M_2 is related to the average grey level intensity of the entire input window – refer to Equation 7.23. The constants n_1 and n_2 are the number of pixels used to calculate M_1 and M_2 respectively. The program's complexity is reduced by eliminating the terminal M_3 and substituting the relationship of Equation 7.23 into the evolved programs.

$$\begin{aligned}
 M_3 &= \left(\frac{1}{n_3} M_1 + \frac{1}{n_1} M_2 \right) \frac{n_1 n_3}{n_1 + n_3}, \text{ where } n_1=719 \text{ and } n_2=881 \\
 &\approx 0.45 M_1 + 0.55 M_2
 \end{aligned} \tag{7.23}$$

An analysis of the best program from each evolutionary run reveals a correlation, using best subsets regression, between the terminal coefficients α_1 and the terminal coefficients $\alpha_2, \alpha_3, \dots, \alpha_7$. A multiple linear regression indicates that the sum of all these terminal

coefficients, i.e. coefficients of terminals based on the average grey level intensities of the various shapes, is zero. The relationship is shown in Equation 7.24. The same underlying regularity has been captured in programs used to detect the nose landmark and the sella landmark (evolved using the restricted feature set) – refer to Equations 7.12 (p. 172) and 7.17 (p. 176) respectively. The significance of this relationship is that when the input window is located on an area of constant brightness, the program’s output is \approx equal to 0. For an explanation into why this occurs, refer to page 165.

There are no other correlations between the various feature coefficients suggesting that there are inter-relationships between the different coefficients. An analysis of the bar chart shown in Figure 7.12 indicates that the evolutionary process selects a combination of features calculated from both the average and standard deviation of grey level intensities within each shape. Several generalisations of influential features are derived from the bar chart and shown in the form of Equations 7.25-7.28. We shall begin by analysing how the most influential features, i.e. the frequency that a feature’s coefficient is either positive or negative in at least 70% of the best programs, are used to locate a landmark. The following investigation will analyse influential features calculated using: (a) mean and then (b) standard deviation.

Influential features calculated using mean, as shown in Figure 7.12, are M_1 and M_2 which are features derived using the PCNN derived shapes. The signs of the coefficient for M_1 and M_2 are positive and negative respectively, which is also consistent with programs that were evolved from the reduced feature set. This is reasonable because if the input window is centred on the sella landmark and we subtract $\alpha_2 M_2$ from $\alpha_1 M_1$, we expect a high output compared to other positions within the image.

Influential features calculated using standard deviation, as shown in Figure 7.12, are S_1 , S_2 and S_7 . Features S_1 and S_2 are derived using the PCNN derived shapes and feature S_7 is derived using the bottom right quadrant. The signs of both coefficients for S_1 and S_2 are negative, which is also consistent with programs that were evolved from the reduced feature set. The coefficient of S_7 is generally positive and when the feature is combined with S_1 and S_2 the program’s output will have a greater decrease when the input window is located on background clutter relative to when located on the known position. The magnitude of S_1 and S_2 coefficients are higher than the other coefficients of features calculated using standard

deviation suggesting the PCNN derived shapes are more influential features.

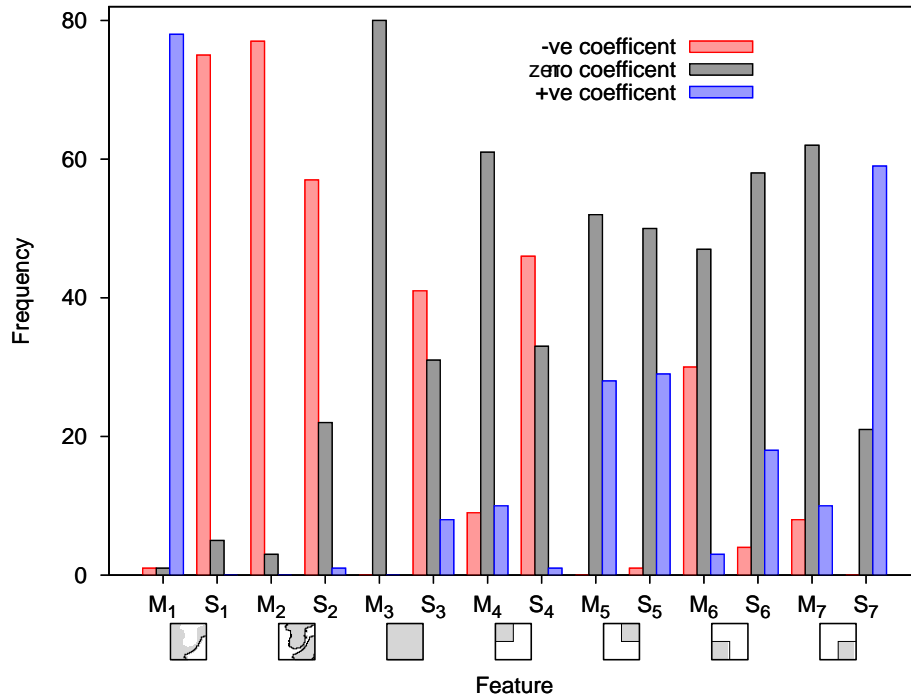


Figure 7.12: Frequency that a feature's coefficient, α_i or β_i in Equation 7.14, is negative, positive or zero (i.e. is not used in a program). The analysis uses the best program at the end of 80 evolutionary runs. Each program was substituted with Equation 7.23 resulting in a zero coefficient for feature M_3 .

$$\sum_{i=1}^7 \alpha_i = 0 \tag{7.24}$$

$$\alpha_i < 0, \text{ where } i=2 \tag{7.25}$$

$$\alpha_i > 0, \text{ where } i=1 \text{ and } 5 \tag{7.26}$$

$$\beta_i < 0, \text{ where } i=1, 2 \text{ and } 4 \tag{7.27}$$

$$\beta_i > 0, \text{ where } i=5 \text{ and } 7 \tag{7.28}$$

where α_i and β_i are the coefficients of M_i and S_i respectively.

7.6 Summary

The aim of this chapter was to determine if there are any regularities captured during the evolutionary process, and if so, whether we could develop explanations of how the evolved programs work for an object detection problem. The investigation was restricted to function sets which only evolved programs exhibiting the behaviour of linear functions for the detection of two landmark types. The investigation demonstrated that the evolved programs are not ad-hoc, and in fact, underlying regularities are being captured during the evolutionary process. The underlying regularities regularly learnt are:

- There is a perfect correlation between the feature coefficients calculated using the means of grey level intensity values within the pre-defined shapes. The significance of this relationship is that when the input window is located on an area of constant brightness, the program's output is \approx equal to 0.
- Features that are calculated using shapes matching the landmark's profile were utilised more frequently than shapes that do not match the landmark's profile. This presents a strong argument for the use of shapes that match the landmark's profile.
- The coefficients of features that are calculated using the standard deviation of grey level intensity values within shapes that match a landmark's profile, are generally negative. The negative coefficients reduce the output when the input window is located on edges or background clutter which are not related to the position of the landmark. The increased values are a result of the shapes not matching the clutter contained within the input window.
- Although each of the evolved programs from the different evolutionary runs were different, it was not uncommon for the evolutionary process to learn an equivalent function. This is strong evidence that the evolved programs are not ad-hoc but are capturing important domain characteristics. This suggests that for situations where there are too many terminals and functions to permit understanding, the evolved programs are still capturing regularities of the domain.

The investigation has provided an understanding of how the simplified programs work and that they are capturing important regularities in the problem domain. We conjecture that programs using the full function and terminal sets are also capturing domain regularities, even though they are difficult to identify.

Chapter 8

Conclusions

In this thesis, a methodology for using genetic programming for accurately locating cephalometric landmarks has been presented. This methodology has been developed using several landmark types that vary in terms of detection difficulty. As part of this investigation we explored a wide range of features, operators and fitness measures.

8.1 Research Questions

The research questions put forward by this thesis were:

1. **Can an existing domain independent approach using pixels as features and genetic programming be used for landmark detection?**

This question was addressed within Chapter 4 to determine if the method claiming to be domain independent – genetic programming and pixel based features – can be used to detect cephalometric landmarks. We demonstrated that this methodology was successful on simple landmarks, however, we found that the false alarm rate increased with the detection difficulty of the landmark. The methodology used to detect the nose (easy) and sella (hard) landmarks produced false alarm rates of 14.8% and 214.8% respectively. False alarms are unacceptable for a cephalometric analysis and therefore further work was required to investigate if the methodology can be improved to reduce false alarm rate. Our work demonstrated that the domain independent approach used in Chapter 4 is not suitable for these types of detection problems.

2. How can the domain independent approach be modified and extended to give better detection performance?

This research question was addressed in Chapter 5. The investigation focussed on alternative fitness measures, function sets and the handcrafting of shapes for generating features.

An investigation into the main issues for improving the detection performance of programs within the genetic programming paradigm revealed:

- That using both the highest output to predict the position of the landmark and detection rate as a fitness metric produces significantly less false alarms than the domain independent approach. We also demonstrated that the accuracy of the detection program could be improved by minimising the error – the distance between the known and predicted position – in the fitness function. However, this fitness metric was not as effective as detection rate for landmarks that are difficult to locate.
- That handcrafted shapes, idiosyncratic to the type of landmark, are able to produce better performing programs compared to the domain independent approach that uses pixel based features.
- That the $\{+, -, \times, /\}$ function set evolved programs that were marginally better performing than programs evolved using $\{+, -\}$ or $\{+, -, \times, /, \min, \max\}$ function sets. The function sets were derived from operators that are commonly cited in genetic programming literature within the computer vision domain. Due to the unexpectedly good performance of programs evolved using the $\{+, -\}$ function set, it would be worthwhile as future work to ascertain the performance of other paradigms that are more suited to learning a linear function.

3. Can handcrafting of shapes be replaced by learning the shapes from examples and will this increase detection accuracy?

This research question was addressed in Chapter 6 where we have investigated the use of a pulse coupled neural network to segment a landmark's regions of interest. The

segments were subsequently used to generate a set of shapes. An investigation of the detection programs that used the PCNN derived shapes and quadrants demonstrated a significant improvement in detection performance when compared to the handcrafted shapes. When tested on three out of the four landmarks (on average, the detection rate increased by 11.7%, 16.9% and 27.3% for the menton, upper lip and sella landmarks respectively) and there was no significant difference in detection performance for the other landmark (incisal upper incisor). This work has gone a long way to achieving the goal of a fully automated approach to generating a set of shapes, however further work is required to be able to automatically identify a suitable set of parameter values for the PCNN.

4. **Are there any underlying algorithms that are learnt during the evolutionary process?**

We have shown that a methodology of simplifying the function set, that restricts programs to linear functions and simplifying programs using parsimony pressure, yields insight into the underlying regularities in the evolved programs. We found that the same underlying regularities were consistently being discovered in many of the evolutionary runs. The regularities include an identical output for each of the programs when located on an area of constant brightness, the higher utilisation of shapes that match a landmark's region of interest and the evolutionary process learning an identical function even though the programs were significantly different.

Even though we have simplified the complexity of the programs by restricting the type of operators in the function set, the simplified approach still gives us confidence that the more complex non-linear functions are still capturing regularities within the problem domain.

8.2 Comparison with other work

The aim of this section is to compare the detection performance of our method with other notable results from the literature survey¹. The methodology we used for this comparison

¹El-Feghi et al. has not been included in Table 8.1 for the reasons explained in Section 2.7.1.2 on page 51.

is based on the PCNN derived shapes and quadrants, as we perceive this to be toward one of our goals of achieving an automated approach to generating a set of features. Our test results are based on a three-fold cross validation applied to 110 images.

	Cardillo 1994	Chakrabartty 2003	Giordano 2005	Rueda 2006	Yue 2006	Our work
Test set size	40	40	26	96	86	110
Hard tissue landmarks						
Sella	53	87	-	39	76	80
Nasion	83	85	81	56	86	84
A Point	77	-	73	68	-	73
Incisal Upper Incisor	76	-	92	-	90	95
Menton	78	-	92	70	98	100
Soft tissue landmarks						
Nose Mid	94	-	-	-	-	100
Upper Lip Mid	-	-	-	-	-	89

Table 8.1: A comparison of our detection rates with results from the literature. Our results are based on detection programs that were evolved using PCNN derived shapes and quadrants. The GP and PCNN parameter settings are detailed in Appendix A.

The results provided in Table 8.1 compare favourably against other published detection results from the literature. However, it should be noted that the detection results of the different approaches are not directly comparable for the following reasons: the sizes of the datasets are not the same, the resolution of the images are different; no statistical test comparing test results is performed; and the comparison is performed on a different set of images.

8.3 Further Work

1. The work presented in this thesis used a selection of landmarks that exhibit a range of detection difficulties. We would like to apply the methodology outlined in Chapter 6 that used the PCNN derived shapes and quadrants to the remaining landmarks.
2. The size of the input window has been determined in a somewhat ad-hoc fashion, based on the criteria of what size performs ‘well’. Work presented in Chapter 6 (page 154) illustrated the sensitivity of the size of the input window with respect to a program’s expected detection performance. The reason for the variation in a program’s expected

detection performance is due to a trade-off between the input window's size being too small and the input window not containing enough information, and too large and the input window containing information that is subject to biological variability. Therefore, we would like an automated way to pre-determine the size of the input window prior to the evolutionary process.

3. The PCNN method demonstrated that if there was a significant difference between the region of interest and background then the method was able to consistently segment the relevant regions of interest. However, this method was not consistent at segmenting the regions of interest when there were subtle changes in greyscale, e.g. sella region, or noisy images. As part of this research we explored many classical image processing techniques in order to segment the regions of interest and as future work we would like to extend this exploration to determine if a suitable segmentation algorithm can be learnt.
4. The features presented in this thesis use the calculated mean and standard deviation of grey-level intensity values within pre-defined shapes. The advantage of using these features is that the processing time to calculate these features is relatively small compared to the more complex features that are derived from Wavelet and Fourier transforms. Although the performance of detection programs using these features had produced some very promising results, we would like to incorporate a spatial relationship between the pixels within the input window.
5. The search for the landmark was confined to a much smaller area within the X-ray. The search area was defined using a statistical heuristic that was driven by anatomical knowledge relative to a known datum point. However, the size of the search area varies depending on the distance of the datum point to the landmark's expected position. The search area is a function of variance which is related to biological variability. The key issue is minimising the size of search area that will also contain the landmark of interest.

Previous work by [44] used a MLP to learn the coordinates of landmarks from an initial

set of four key points. This method differs from the other approaches in that it learns the spatial and scale relationships to determine the coordinates of the landmarks rather than a sliding window approach. Although the results were found to be inaccurate for automatically locating landmarks, we believe that a method similar to this would significantly reduce the size of the search area compared to the heuristic that we currently use. We expect that a smaller search area will significantly reduce training times and potentially increase detection rate.

6. Possible scope for future work is to determine if the detection rate (Section 5.4.1) and the distance error (Section 5.4.5) fitness functions can be incorporated as multi-objective optimisation problem. This would hopefully evolve programs that have a high detection rate and high detection accuracy. The fitness function used throughout the majority of this thesis only rewards programs on the basis that the position is predicted within 2 mm of the known position; there is currently no incentive for the landmark to be predicted less than 2 mm.
7. Part of our work used the PCNN to highlight regions of interest with the purpose to improve discrimination between the landmark and background pixels. However, using the PCNN requires several parameters to be manually tuned which can be a difficult task depending on where the region of interest is located. A search method, such as a genetic algorithm, may be able to learn a suitable set of PCNN parameters that would automate this task. A similar concept has been used previously for optimising the parameters regulating a video-based tracking system [105].
8. Work presented in Section 5.5 demonstrated that programs exhibiting the behaviour of linear functions are nearly as successful at detecting landmarks as programs that exhibit the behaviour of non-linear functions. Therefore, it is worth exploring a paradigm that is better suited to learning linear functions, such as genetic algorithms or particle swarm optimisation. The advantages of using a paradigm such as genetic algorithms compared to genetic programming are: they are more suited to solving linear problems, the time spent processing a bit-string is significantly less than processing a program tree and the search space is significantly smaller.

9. Research by [58] used genetic programming to learn a multi-stage approach for detecting vehicles in infrared line scan (IRLS) images. The aim of each stage was to reduce the number of pixel positions (false alarms) from each subsequent stage and the vehicles were detected from the remaining pixel positions in the final stage. This approach was shown to improve the detection performance when compared to a single detection program. As future work it would be worthwhile to learn an initial classifier to reduce the number of pixel positions and then evolve a detection program using our methodology to predict the position of the landmark from the remaining pixel positions.

Bibliography

- [1] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, November 2004. IEEE Computer Society.
- [2] Davide Agnelli, Alessandro Bollini, and Luca Lombardi. Image classification: An evolutionary approach. *Pattern Recognition Letters*, 23(1-3):303–309, January 2002. Elsevier Science Inc.
- [3] Karl Astrom, 28 September 2004. Cognitive vision [online]. Accessed 25 July 2007, <<http://www.maths.lth.se/matematiklth/vision/research2004/node4.html>>.
- [4] Nicholas Ayache. Medical image analysis and simulation. In R.K. Shyamasundar and K. Ueda, editors, *Proceedings of the Third Asian Computing Science Conference on Advances in Computing Science*, Kathmandu, Nepal, 9–11 December 1997, volume 1345 of *Lecture Notes in Computer Science*, pages 4–17. Springer-Verlag.
- [5] W. Banzhaf and W.B. Langdon. Some considerations on the reason for bloat. *Genetic Programming and Evolvable Machines*, 3(1):81–91, March 2002. Springer Netherlands.
- [6] Wolfgang Banzhaf. Genetic programming for pedestrians. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, San Francisco, California, USA, 17–21 July 1993, page 628. Morgan Kaufmann Publishers, Inc.
- [7] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming – An Introduction: On the Automatic Evolution of Computer Programs*

- and Its Applications*. Morgan Kaufmann Publishers, Inc., San Francisco, California, USA and dpunkt.verlag, Heidelberg, Germany, January 1998.
- [8] Karl Benson. Evolving finite state machines with embedded genetic programming for automatic target detection. In *Proceedings of the 2000 Congress on Evolutionary Computation*, San Diego, California, USA, 16–19 July 2000, volume 2, pages 1543–1549. IEEE Press.
- [9] Karl Benson, David Booth, James Cubillo, and Colin Reeves. Automatic detection of ships in spaceborne SAR imagery. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Las Vegas, Nevada, USA, 8–12 July 2000, page 767. Morgan Kaufmann Publishers, Inc.
- [10] Bir Bhanu and Yingqiang Lin. Object detection in multi-modal images using genetic programming. *Applied Soft Computing*, 4(2):175–201, May 2004. Elsevier Science Inc.
- [11] Bir Bhanu, Jing Peng, and Bruce Draper, editors. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVWPR'04)*, Washington, DC, USA, 28 June 2004. IEEE Computer Society.
- [12] Bir Bhanu, Jing Peng, Bruce Draper, Christian Shelton, Katsushi Ikeuchi, Erik Learned-Miller, Sally Goldman, and James Wang, editors. *Proceedings of the Twentieth National Conference on Artificial Intelligence AAAI, Workshop on Learning in Computer Vision*, Pittsburgh, Pennsylvania, USA, 9–10 July 2005. AAAI Press.
- [13] Hendrik Blockeel and Jan Struyf. Deriving biased classifiers for better ROC performance. In Marko Grobelnik and Dunja Mladenic, editors, *Proceedings of the 4th International Multi-conference on Information Society (IS 2001)*, Jožef Stefan Institute, Ljubljana, Slovenia, 22–26 October 2001, pages 124–127.
- [14] Walter Böhm and Andreas Geyer-Schulz. Exact uniform initialization for genetic programming. In Richard K. Belew and Michael D. Vose, editors, *Foundations of Genetic*

- Algorithms (FOGA4)*, University of San Diego, California, USA, 2–5 August 1996, pages 379–407. Morgan Kaufmann Publishers, Inc.
- [15] Celia C. Bojarczuk, Heitor S. Lopes, and Alex A. Freitas. Discovering comprehensible classification rules by using genetic programming: A case study in a medical domain. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, Orlando, Florida, USA, 13–17 July 1999, volume 2, pages 953–958. Morgan Kaufmann Publishers, Inc.
- [16] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. Elsevier Science Inc.
- [17] Markus Brameier and Wolfgang Banzhaf. Effective linear genetic programming. Technical report, Department of Computer Science, University of Dortmund, Dortmund, Germany, 2001.
- [18] Lorenzo Bruzzone and Roberto Cossu. Analysis of multitemporal remote-sensing images for change detection: Bayesian thresholding approaches. in *Geospatial Pattern Recognition*, pages 319–336, 2002. Research Signpost/Transworld Research.
- [19] J. Cardillo and M.A. Sid-Ahmed. An image processing system for the automatic extraction of craniofacial landmarks. In *Conference Record of the 1991 IEEE Nuclear Science Symposium and Medical Imaging Conference*, Santa Fe, New Mexico, USA, 2–9 November 1991, volume 3, pages 2124–2128. IEEE Press.
- [20] J. Cardillo and M.A. Sid-Ahmed. An image processing system for locating craniofacial landmarks. *IEEE Transactions on Medical Imaging*, 13(2):275–289, June 1994. IEEE Engineering in Medicine and Biology Society, IEEE Signal Processing Society, IEEE Nuclear and Plasma Sciences Society, and IEEE Ultrasonics, Ferroelectrics and Frequency Control Society.

- [21] Mario I. Chacón, Alejandro Zimmerman, and David Sanchez. PCNNP: A pulse-coupled neural network processor. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN '02)*, Honolulu, Hawaii, 12–17 May 2002, volume 2, pages 1581–1584. IEEE Press.
- [22] Shantanu Chakrabartty, Masakazu Yagi, Tadashi Shibata, and Gert Cauwenberghs. Robust cephalometric landmark identification using support vector machines. In *Proceedings of the 2003 International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, Hong Kong, 6–10 April 2003, volume 2, pages 825–828. IEEE Signal Processing Society.
- [23] Nitesh Chawla, 6 February 2002. Performance measures [online]. Accessed 25 July 2007, <www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/chawla02a-html/node2.html>.
- [24] Yen-ting Chen, Kuo-sheng Cheng, and Jia-kuang Liu. Feature subimage extraction for cephalogram landmarking. In H.K. Chang and Y.T. Zhang, editors, *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Hong Kong SAR, China, 29 October–1 November 1998, volume 3, pages 1414–1417. IEEE Engineering in Medicine and Biology Society.
- [25] Yen-ting Chen, Kuo-sheng Cheng, and Jia-kuang Liu. Improving cephalogram analysis through feature subimage extraction. *IEEE Engineering in Medicine and Biology Magazine*, 18(1):25–31, January–February 1999. IEEE Engineering in Medicine and Biology Society.
- [26] Kuo-sheng Cheng, Yen-ting Chen, and Jia-kuang Liu. Automatic analysis of landmarks in cephalograms. In Joachim H. Nagel and William M. Smith, editors, *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Orlando, Florida, USA, 31 October–3 November 1991, volume 13, pages 338–339. IEEE Engineering in Medicine and Biology Society.
- [27] Ioanna Christoyianni, Evalgelos Dermatas, and George Kokkinakis. Fast detection of masses in computer-aided mammography. *IEEE Signal Processing Magazine*, 17:54–64, January 2000. IEEE Signal Processing Society.

- [28] Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999. Springer London.
- [29] James H. Cooley and Thomas Cooley. Segmentation and discrimination of structural and spectral information using multi-layered pulse couple neural networks. In Tammy I. Stein, editor, *Proceedings of the IEEE 1999 International Geoscience and Remote Sensing Symposium (IGARSS'99)*, Hamburg, Germany, 28 June–2 July 1999, volume 1, pages 80–82. IEEE Press.
- [30] Michael Lynn Cramer. A representation for the adaptive generation of simple sequential programs. In John J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA, 24–26 July 1985, pages 183–187. Lawrence Erlbaum Associates, Inc.
- [31] Jason M. Daida, Tommaso F. Bersano-Begey, Steven J. Ross, and John F. Vesecky. Computer-assisted design of image classification algorithms: Dynamic and static fitness evaluations in a scaffolded genetic programming environment. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, California, USA, 28–31 July 1996, pages 279–284. The MIT Press.
- [32] Jason M. Daida, Robert G. Onstott, Tommaso F. Bersano-Begey, Steven J. Ross, and John E Vesecky. Ice roughness classification and ERS SAR imagery of arctic sea ice: Evaluation of feature-extraction algorithms by genetic programming. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS'96), Remote Sensing for a Sustainable Future*, Lincoln, Nebraska, USA, 27–31 May 1996, volume 3, pages 1520–1522. IEEE Press.
- [33] I. De Falco, A. Della Cioppa, and E. Tarantino. Unsupervised spectral pattern recognition for multispectral images by means of a genetic programming approach. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Mar-

- row, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, Honolulu, Hawaii, 12–17 May 2002, volume 1, pages 231–236. IEEE Press.
- [34] I. De Falco, A. Della Cioppa, and E. Tarantino. Discovering interesting classification rules with genetic programming. *Applied Soft Computing*, 1(4):257–269, May 2001. Elsevier Science Inc.
- [35] Edwin D. de Jong, Richard A. Watson, and Jordan B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, 7–11 July 2001, pages 11–18. Morgan Kaufmann Publishers, Inc.
- [36] Kalyanmoy Deb and Deb Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., 2001.
- [37] Marc Ebner. On the evolution of interest operators using genetic programming. In Riccardo Poli, W.B. Langdon, Marc Schoenauer, Terry Fogarty, and Wolfgang Banzhaf, editors, *Late Breaking Papers at EuroGP'98: The First European Workshop on Genetic Programming*, Paris, France, 14–15 April 1998, pages 6–10. School of Computer Science, The University of Birmingham, UK.
- [38] Marc Ebner. Evolving color constancy. *Pattern Recognition Letters*, 27(11):1220–1229, August 2006. Elsevier Science Inc.
- [39] Marc Ebner and Andreas Zell. Evolving a task specific image operator. In Riccardo Poli, Hans-Michael Voigt, Stefano Cagnoni, Dave Corne, George D. Smith, and Terence C. Fogarty, editors, *Evolutionary Image Analysis, Signal Processing and Telecommunications: First European Workshops, EvoIASP'99 and EuroEcTel'99*, Göteborg, Sweden, 26–27 May 1999, volume 1596 of *Lecture Notes in Computer Science*, pages 74–89. Springer-Verlag.

- [40] R. Eckhorn, H.J. Reitboeck, M. Arndt, and P. Dicke. Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex. *Neural Computation*, 2(3):293–307, 1990. The MIT Press.
- [41] Masakazu Ejiri. Machine vision technology: Past, present and future. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems '90 (IROS '90). 'Towards a New Frontier of Applications'*, Tsuchiuro, Japan, 3–6 July 1990, volume 1, pages XXIX–XXXX. IEEE Press.
- [42] I. El-Feghi, Y. Alginahi, M.A. Sid-Ahmed, and M. Ahmadi. Craniofacial landmarks extraction by partial least squares regression. In *Proceedings of the 2004 International Symposium on Circuits and Systems ISCAS '04*, Vancouver, British Columbia, Canada, 23–26 May 2004, volume 4, pages IV–45–IV–48. IEEE Circuits and Systems Society.
- [43] I. El-Feghi, M.A. Sid-Ahmed, and M. Ahmadi. Location of craniofacial landmarks on X-ray images by employing fuzzy neural network. In *Proceedings of the 2002 45th Midwest Symposium on Circuits and Systems, 2002 (MWSCAS-2002)*, Tulsa, Oklahoma, USA, 4–7 August 2002, volume 3, pages III–348–III–351. IEEE Press.
- [44] I. El-Feghi, M.A. Sid-Ahmed, and M. Ahmadi. Automatic identification and localization of craniofacial landmarks using multi layer neural network. In Randy E. Ellis and Terry M. Peters, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2003*, Montréal, Canada, 15–18 November 2003, volume 2879 of *Lecture Notes in Computer Science*, pages I–643–I–654. Springer-Verlag.
- [45] I. El-Feghi, M.A. Sid-Ahmed, and M. Ahmadi. Automatic localization of craniofacial landmarks for assisted cephalometry. *Pattern Recognition*, 37(3):609–621, March 2004. Elsevier Science Inc.
- [46] I. El-Henawy, T. El-Areef, and A.A. Karawia. On wavelets applications in medical image denoising. *Machine Graphics & Vision International Journal*, 12(3):393–404, March 2003. Institute of Computer Science of the Polish Academy of Sciences.

- [47] Christos Emmanouilidis. Evolutionary multi-objective feature selection and ROC analysis with application to industrial machinery fault diagnosis. In K.C. Giannakoglou, D.T. Tsahalis, J. Périaux, K.D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems: Proceedings of the EUROGEN2001 Conference*, Athens, Greece, 19–21 September 2001, pages 319–324. International Center for Numerical Methods in Engineering (CIMNE).
- [48] R.J. Ferrari, A.C.P.L.F. de Carvalho, P.M. Azevedo Marques, and A.F. Frère. Computerized classification of breast lesions: Shape and texture analysis using an artificial neural network. In *Proceedings of the Seventh International Conference on Image Processing and its Applications, 1999 (IPA'99)*, Manchester, UK, 13–15 July 1999, volume 2, pages 517–521. IEEE Press.
- [49] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.
- [50] D.B. Forsyth and D.N. Davis. Assessment of an automated cephalometric analysis system. *The European Journal of Orthodontics*, 18(1):471–478, October 1996. Oxford University Press.
- [51] GE Healthcare, 2007. Image quality parameters for digital detector [online]. Accessed 25 July 2007, <<http://www.gehealthcare.com/user/xr/edu/products/digimgtutorial.html>>.
- [52] Daniela Giordano, Rosalia Leonardi, Francesco Maiorana, Gabriele Cristaldi, and Maria Luisa Distefano. Automatic landmarking of cephalograms by cellular neural networks. In Silvia Miksch, Jim Hunter, and Elpida Keravnou, editors, *Proceedings of the 10th Conference on Artificial Intelligence in Medicine*, Aberdeen, UK, 23–27 July 2005, volume 3581 of *Lecture Notes in Computer Science*, pages 333–342. Springer-Verlag.
- [53] H.F. Gray, R.J. Maxwell, I. Martinez-Perez, C. Arus, and S. Cerdan. Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, edi-

- tors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, California, USA, 28–31 July 1996, page 424. The MIT Press.
- [54] Michael Gribskov and Nina L. Robinson. The use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, March 1996. Elsevier Science Inc.
- [55] Steven Gustafson, Anikó Ekárt, Edmund Burke, and Graham Kendall. Problem difficulty and code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 5(3):271–290, September 2004. Springer Netherlands.
- [56] Christopher Harris and Bernard Buxton. Evolving edge detectors with genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, California, USA, 28–31 July 1996, pages 309–315. The MIT Press.
- [57] Jin-Hyuk Hong and Sung-Bae Cho. Effective rule discovery using genetic programming for DNA microarray analysis. In Sung-Bae Cho, Nguyen Xuan Hoai, and Yin Shan, editors, *Proceedings of the First Asian-Pacific Workshop on Genetic Programming (ASPGP03)*, Canberra, Australia, 8 December 2003, pages 53–61.
- [58] Daniel Howard and Simon C. Roberts. Evolving object detectors for infrared imagery: A comparison of texture analysis against simple statistics. In Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Périaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, Jyväskylä, Finland, 30 May–3 June 1999, pages 79–86. John Wiley & Sons, Inc.
- [59] Daniel Howard, Simon C. Roberts, and Richard Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30(5):303–311, May 1999. Elsevier Science Inc.
- [60] Daniel Howard, Simon C. Roberts, and Conor Ryan. Pragmatic genetic programming strategy for the problem of vehicle detection in airborne reconnaissance. *Pattern Recognition Letters*, 27(11):1275–1288, August 2006. Elsevier Science Inc.

- [61] I.A. Hunter, J.J. Soraghan, and T. McDonagh. Fully automatic left ventricular boundary extraction in echocardiographic images. In *Proceedings of the 1995 IEEE Computers in Cardiology Conference*, Vienna, Austria, 10–13 September 1995, pages 741–744. IEEE Computer Society.
- [62] Syed Afaq Husain and Eiho Shigeru. Use of neural networks for feature based recognition of liver region on CT images. In *Proceedings of the 2000 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing*, Sydney, Australia, 11–13 December 2000, volume 2, pages 831–840. IEEE Press.
- [63] Tim J. Hutton, Sue Cunningham, and Peter Hammond. An evaluation of active shape models for the automatic identification of cephalometric landmarks. *European Journal of Orthodontics*, 22(5):499–508, October 2000. Oxford University Press.
- [64] H.H.S. Ip and Z. Hussain. Automatic identification of cephalometric features on skull radiographs. *Acta Polytechnica Scandinavica, Applied physics series*, 2(149):194–197, 1985. Finnish Academy of Technology, Finland.
- [65] Jacob T. Jackson, Gregg H. Gunsch, Roger L. Claypoole, Jr., and Gary B. Lamont. Blind steganography detection using a computational immune system approach: A proposal. In *Proceedings of the Digital Forensic Research Workshop*, Syracuse, New York, USA, 2002.
- [66] Helen E. Johnson, Richard J. Gilbert, Michael K. Winson, Royston Goodacre, Aileen R. Smith, Jem J. Rowland, Michael A. Hall, and Douglas B. Kell. Explanatory analysis of the metabolome using genetic programming of simple, interpretable rules. *Genetic Programming and Evolvable Machines*, 1(3):243–258, July 2000. Springer Netherlands.
- [67] J.L. Johnson and M.L. Padgett. PCNN models and applications. *IEEE Transactions on Neural Networks*, 10(3):480–498, May 1999. IEEE Computational Intelligence Society.
- [68] John L. Johnson and Dieter Ritter. Observation of periodic waves in a pulse-coupled neural network. *Optics Letters*, 18(15):1253–1255, August 1993. Optical Society of America.

- [69] Philip Kahn. Building blocks for computer vision systems. *IEEE Expert Magazine*, 8(6):40–50, 1993. IEEE Computer Society.
- [70] Wolfgang Kantschik and Wolfgang Banzhaf. Linear-tree GP and its comparison with other GP structures. In Julian Miller, Marco Tomassini, Pier Luca Lanzi, Conor Ryan, Andrea G.B. Tettamanzi, and William B. Langdon, editors, *Proceedings of the Genetic Programming: 4th European Conference, EuroGP 2001*, Lake Como, Italy, 18–20 April 2001, volume 2038 of *Lecture Notes in Computer Science*, pages 302–312. Springer-Verlag.
- [71] Wolfgang Kantschik and Wolfgang Banzhaf. Linear-graph GP – A new GP structure. In James A. Foster, Evelyne Lutton, Julian Miller, Conor Ryan, and Andrea G.B. Tettamanzi, editors, *Proceedings of the Genetic Programming: 5th European Conference, EuroGP 2002*, Kinsale, Ireland, 3–5 April 2002, volume 2278 of *Lecture Notes in Computer Science*, pages 83–92. Springer-Verlag.
- [72] Paul E. Keller and A.D. McKinnon. Pulse-coupled neural networks for medical image analysis. In Kevin L. Priddy, Paul E. Keller, David B. Fogel, and James C. Bezdek, editors, *Applications and Science of Computational Intelligence II*, Orlando, Florida, USA, 5-8 April 1999, volume 3722 of *SPIE*, pages 444–451. Society of Photo-Optical Instrumentation Engineers.
- [73] John R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. In N.S. Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, Michigan, USA, 20–25 August 1989, volume 1, pages 768–774. Morgan Kaufmann Publishers, Inc.
- [74] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, USA, December 1992.
- [75] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. The MIT Press, Cambridge, Massachusetts, USA, May 1994.

- [76] G. Kuntimad and H.S. Ranganath. Perfect image segmentation using pulse coupled neural networks. *IEEE Transactions on Neural Networks*, 10(3):591–598, May 1999. IEEE Computational Intelligence Society.
- [77] Matthew A. Kupinski and Mark A. Anastasio. Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curves. *IEEE Transactions on Medical Imaging*, 18(8):675–685, August 1999. IEEE Engineering in Medicine and Biology Society, IEEE Signal Processing Society, IEEE Nuclear and Plasma Sciences Society, and IEEE Ultrasonics, Ferroelectrics and Frequency Control Society.
- [78] Timothy Lai. Discovery of understandable math formulas using genetic programming. In John R. Koza, editor, *Genetic Algorithms and Genetic Programming at Stanford 2003*, pages 118–127. Stanford Bookstore, Stanford, California, USA, 4 December 2003.
- [79] Brian Lam and Vic Ciesielski. Discovery of human-competitive image texture feature extraction programs using genetic programming. In Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund Burke, Paul Darwen, Dipankar Dasgupta, Dario Floreano, James Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andy Tyrrell, editors, *Genetic and Evolutionary Computation – GECCO 2004*, Seattle, Washington, USA, 26–30 June 2004, volume 3103 of *Lecture Notes in Computer Science*, pages 1114–1125. Springer-Verlag.
- [80] W.B. Langdon, S.J. Barrett, and B.F. Buxton. Genetic programming for combining neural networks for drug discovery. In Rajkumar Roy, Mario Köppen, Seppo Ovaska, Takeshi Furuhashi, and Frank Hoffmann, editors, *Soft Computing and Industry: Recent Applications*, On-line, 10–24 September 2001, pages 597–608. Springer-Verlag.
- [81] W.B. Langdon and R. Poli. Fitness causes bloat. In P.K. Chawdhry, R. Roy, and R.K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, On-line, 23–27 June 1997, pages 13–22. Springer-Verlag New York.

- [82] J. Lehr, J.B. Sibarita, and J.M. Chassery. Image restoration in X-ray microscopy: PSF determination and biological applications. *IEEE Transactions on Image Processing*, 7(2):258–263, February 1998. IEEE Signal Processing Society.
- [83] A.D. Levy-Mandel, A.N. Venetsanopoulos, and J.K. Tsotsos. Knowledge-based landmarking of cephalograms. *Computers and Biomedical Research*, 19(3):282–309, June 1986. Academic Press.
- [84] Yingqiang Lin and Bir Bhanu. Discovering operators and features for object detection. In R. Kasturi, D. Laurendeau, and C. Suen, editors, *Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002)*, Québec City, Québec, Canada, 11–15 August 2002, volume 3, pages 339–342. IEEE Computer Society.
- [85] Thomas Lindblad and Jason M. Kinser. *Image Processing using Pulse-Coupled Neural Networks*. Springer-Verlag New York, Inc., 1998.
- [86] Thomas Lindblad, Clark S. Lindsey, Mary Lou Padgett, Alexandre Vanyachine, and Yuri Zanevsky. Digital X-ray image processing using biologically inspired methods. In *The 1998 IEEE International Joint Conference on Neural Networks Proceedings: IEEE World Congress on Computational Intelligence*, Anchorage, Alaska, USA, 4–9 May 1998, volume 2, pages 803–808. IEEE Press.
- [87] J. Liu, Y. Chen, and K. Cheng. Accuracy of computerized automatic identification of cephalometric landmarks. *American Journal of Orthodontics and Dentofacial Orthopedics*, 118(5):535–540, November 2000. Elsevier Science Inc.
- [88] Bradley J. Lucier, Sudhakar Mamillapalli, and Jens Palsberg. Program optimization for faster genetic programming. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, Wisconsin, USA, 22–25 July 1998, pages 202–207. Morgan Kaufmann Publishers, Inc.

- [89] Sean Luke. *Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat*. PhD thesis, Department of Computer Science, A.V. Williams Building, University of Maryland, College Park, Maryland, USA, 2000.
- [90] Sean Luke. When short runs beat long runs. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, 7–11 July 2001, pages 74–80. Morgan Kaufmann Publishers, Inc.
- [91] Sean Luke and Liviu Panait. A survey and comparison of tree generation algorithms. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, 7–11 July 2001, pages 81–88. Morgan Kaufmann Publishers, Inc.
- [92] Sean Luke and Lee Spector. A revised comparison of crossover and mutation in genetic programming. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, Wisconsin, USA, 22–25 July 1998, pages 208–213. Morgan Kaufmann Publishers, Inc.
- [93] Martin C. Martin. Visual obstacle avoidance using genetic programming: First results. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, 7–11 July 2001, pages 1107–1113. Morgan Kaufmann Publishers, Inc.
- [94] Martin C. Martin. Genetic programming for real world robot vision. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*,

- Luasanne, Switzerland, 30 September–5 October 2002, volume 1, pages 67–72. IEEE Press.
- [95] Martin C. Martin. Evolving visual sonar: Depth from monocular images. *Pattern Recognition Letters*, 27(11):1174–1180, August 2006. Elsevier Science Inc.
- [96] Julian F. Miller and Peter Thomson. Cartesian genetic programming. In Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian F. Miller, Peter Nordin, and Terence C. Fogarty, editors, *Proceedings of Third European Conference on Genetic Programming (EuroGP 2000)*, Edinburgh, Scotland, UK, 15–16 April 2000, volume 1802 of *Lecture Notes in Computer Science*, pages 121–132. Springer-Verlag.
- [97] Henning Müller, Nicolas Michoux, David Bandon, and Antoine Geissbuhler. A review of content-based image retrieval systems in medical applications - clinical benefits and future directions. *International Journal of Medical Informatics*, 73(1):1–23, February 2004. Elsevier Science Inc.
- [98] Anuj Mohan. Object detection in images by components. Technical Report AIM-1664, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 11 August 1999.
- [99] R. Mutihac, M. Candusso, A. Cerdeira, A. Cicuttin, F. Fratnik, and A.A. Colavita. X-ray digital image improvement by maximum entropy. In *Proceedings of the 13th International Conference on Digital Signal Processing (DSP 97)*, Santorini, Greece, 2–4 July 1997, volume 2, pages 1149–1152. IEEE Press.
- [100] Po Shun Ngan, Man Leung Wong, Wai Lam, Kwong Sak Leung, and Jack C. Y. Cheng. Medical data mining using evolutionary computation. *Artificial Intelligence in Medicine*, 16(1):73–96, May 1999. Elsevier Science Inc.
- [101] Sooyol Ok, Kazuo Miyashita, and Seiichi Nishihara. Improving performance of GP by adaptive terminal selection. In Riichiro Mizoguchi and John Slaney, editors, *PRICAI 2000 Topics in Artificial Intelligence: 6th Pacific Rim International Conference on*

- Artificial Intelligence*, Melbourne, Australia, 28 August–1 September 2000, volume 1886 of *Lecture Notes in Artificial Intelligence*, pages 435–445. Springer-Verlag.
- [102] Michael O’Neill and Conor Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 2003.
- [103] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Proceedings of the Sixth International Conference on Computer Vision*, Bombay, India, 4–7 January 1998, page 555. IEEE Computer Society.
- [104] S. Parthasarathy, S.T. Nugent, P.G. Gregson, and D.F. Fay. Automatic landmarking of cephalograms. *Computers and Biomedical Research*, 22(3):248–269, June 1989. Academic Press.
- [105] Óscar Pérez, Jesús García, Antonio Berlanga, and José M. Molina. Evolving parameters of surveillance video systems for non-overfitted learning. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, David W. Corne, Rolf Drechsler, Yaochu Jin, Penousal Machado, Elena Marchiori, Juan Romero, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, Lausanne, Switzerland, 30 March–1 April 2005, volume 3449 of *Lecture Notes in Computer Science*, pages 386–395. Springer-Verlag.
- [106] Nemanja Petrović and Vladimir Crnojević. Impulse noise detection based on robust statistics and genetic programming. In Jacques Blanc-Talon, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, Antwerp, Belgium, 20–23 September 2005, volume 3708 of *Lecture Notes in Computer Science*, pages 643–649. Springer-Verlag.
- [107] Riccardo Poli. Genetic programming for feature detection and image segmentation. In Terence C. Fogarty, editor, *Evolutionary Computing: AISB Workshop*, University of Sussex, Brighton, UK, 1–2 April 1996, volume 1143 of *Lecture Notes in Computer Science*, pages 110–125. Springer-Verlag.

- [108] Riccardo Poli. Genetic programming for image analysis. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, California, USA, 28–31 July 1996, pages 363–368. The MIT Press.
- [109] Thierry Pun, Guido Gerig, and Osman Ratib. Image analysis and computer vision in medicine. *Computerized Medical Imaging and Graphics*, 18(2):85–96, March–April 1994. Elsevier Science Inc.
- [110] Marcos I. Quintana, Riccardo Poli, and Ela Claridge. Morphological algorithm design for binary images using genetic programming. *Genetic Programming and Evolvable Machines*, 7(1):81–102, March 2006. Springer Netherlands.
- [111] T. Rakosi. *An atlas of cephalometric radiography*. Wolfe Medical Publications, London, 1982.
- [112] H.S. Ranganath and G. Kuntimad. Pulse coupled neural networks for image processing. In *Proceedings of the IEEE Southeastcon '95: Visualize the Future*, Raleigh, North Carolina, USA, 26–29 March 1995, pages 37–43. IEEE Press.
- [113] H.S. Ranganath and G. Kuntimad. Object detection using pulse coupled neural networks. *IEEE Transactions on Neural Networks*, 10(3):615–620, May 1999. IEEE Computational Intelligence Society.
- [114] Patrick J. Rauss, Jason M. Daida, and Shahbaz Chaudhary. Classification of spectral imagery using genetic programming. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Las Vegas, Nevada, USA, 8–12 July 2000, pages 726–733. Morgan Kaufmann Publishers, Inc.
- [115] Ali M. Reza, Justin G.R. Delva, Roy Schley, and Robert D. Turney. Issues on medical image enhancement. In *Proceedings of the International Conference on Signal Processing Applications and Technology (ICSPAT '98)*, Toronto, Ontario, Canada, 13–16 September 1998.

- [116] Rick L. Riolo and Mark P. Line. Automatic discovery of classification and estimation algorithms for earth-observation satellite imagery. In E.V. Siegel and J.R. Koza, editors, *Proceedings of the AAAI 1995 Fall Symposium on Genetic Programming*, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 10–12 November 1995, pages 73–77. The AAAI Press.
- [117] Daniel Rivero, Juan R. Rabuñal, Julián Dorado, and Alejandro Pazos. Using genetic programming for character discrimination in damaged documents. In Günther R. Raidl, Stefano Cagnoni, Jürgen Branke, David W. Corne, Rolf Drechsler, Yaochu Jin, Colin G. Johnson, Penousal Machado, Elena Marchiori, Franz Rothlauf, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, Coimbra, Portugal, 5–7 April 2004, volume 3005 of *Lecture Notes in Computer Science*, pages 349–358. Springer-Verlag.
- [118] M. Roberts and E. Claridge. A multi-stage approach to cooperatively coevolving image feature construction and object detection. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, David W. Corne, Rolf Drechsler, Yaochu Jin, Penousal Machado, Elena Marchiori, Juan Romero, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, Lausanne, Switzerland, 30 March–1 April 2005, volume 3449 of *Lecture Notes in Computer Science*, pages 396–406. Springer-Verlag.
- [119] Mark E. Roberts and Ela Claridge. Cooperative coevolution of image feature construction and object detection. In Xin Yao, Edmund Burke, José A. Lozano, Jim Smith, Juan J. Merelo-Guervós, John A. Bullinaria, Jonathan Rowe, Peter Tiño, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VIII*, Birmingham, UK, 18–22 September 2004, volume 3242 of *Lecture Notes in Computer Science*, pages 902–911. Springer-Verlag.

- [120] Simon C. Roberts and Daniel Howard. Genetic programming for image analysis: Orientation detection. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Las Vegas, Nevada, USA, 8–12 July 2000, pages 651–657. Morgan Kaufmann Publisher, Inc.
- [121] Paul L. Rosin and Javier Hervás. Image thresholding for landslide detection by genetic programming. In Lorenzo Bruzzone and Paul Smits, editors, *Analysis of Multi-Temporal Remote Sensing Images: Proceedings of the First International Workshop on Multitemp 2001*, University of Trento, Italy, 13–14 September 2001, volume 2 of *Remote Sensing*, pages 65–72. World Scientific Publishing Co.
- [122] Brian J. Ross, Frank Fueten, and Dmytro Y. Yashkir. Edge detection of petrographic images using genetic programming. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Las Vegas, Nevada, USA, 8–12 July 2000, pages 658–665. Morgan Kaufmann Publishers, Inc.
- [123] Brian J. Ross, Anthony G. Gualtieri, Frank Fueten, and Paul Budkewitsch. Hyperspectral image analysis using genetic programming. In William B. Langdon, Erick Cantú-Paz, Keith E. Mathias, Rajkumar Roy, David Davis, Riccardo Poli, Karthik Balakrishnan, Vasant Honavar, Günter Rudolph, Joachim Wegener, Larry Bull, Mitchell A. Potter, Alan C. Schultz, Julian F. Miller, Edmund K. Burke, and Natasa Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, New York, USA, 9–13 July 2002, pages 1196–1203. Morgan Kaufmann Publishers, Inc.
- [124] Brian J. Ross, Anthony G. Gualtieri, Frank Fueten, and Paul Budkewitsch. Hyperspectral image analysis using genetic programming. *Applied Soft Computing*, (5):147–156, January 2005. Elsevier Science Inc.
- [125] Rudjer Boskovic Institute, 2001. Evaluation of models (discovered knowledge) [online]. Accessed 25 July 2007, <http://dms.irb.hr/tutorial/tut_mod_eval_4.php>.

- [126] D.J. Rudolph, P.M. Sinclair, and J.M. Coggins. Automatic computerized radiograph identification of cephalometric landmarks. *American Journal of Orthodontics and Dentofacial Orthopedics*, 113(2):173–179, February 1998. Elsevier Science Inc.
- [127] Sylvia Rueda and Mariano Alcañiz Raya. An approach for the automatic cephalometric landmark detection using mathematical morphology and active appearance models. In Rasmus Larsen, Mads Nielsen, and Jon Sporring, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006*, Copenhagen, Denmark, 1–6 October 2006, volume 4190 of *Lecture Notes in Computer Science*, pages 159–166. Springer-Verlag.
- [128] Berkman Sahiner, Heang-Ping Chan, Datong Wei, Nicholas Petrick, Mark A. Helvie, Dorit D. Adler, and Mitchell M. Goodsitt. Image feature selection by a genetic algorithm: Application to classification of mass and normal breast tissue. *Medical Physics*, 23(10):1671–1684, October 1996. American Institute of Physics.
- [129] Andrew Schein, 5 March 2002. ROCtools: ROC curve code [online]. Accessed 25 July 2007, <http://www.cis.upenn.edu/datamining/software_dist/ROCtools/>.
- [130] Linda G. Shapiro and George Stockman. *Computer Vision*. Prentice Hall Professional Technical Reference, 2001.
- [131] Mukul V. Shirvaikar and Mohan M. Trivedi. A neural network filter to detect small targets in high clutter backgrounds. *IEEE Transactions on Neural Networks*, 6(1):252–257, January 1995. IEEE Computational Intelligence Society.
- [132] Peter J. Smith. *Applied Linear Models – Part II*. RMIT Lecture Notes in Statistics and Operations Research, preliminary version edition, 1995.
- [133] Andy Song and Vic Ciesielski. Fast texture segmentation using genetic programming. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, Canberra, 8–12 December 2003, pages 2126–2133. IEEE Press.

- [134] Andy Song and Vic Ciesielski. Texture analysis by genetic programming. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)*, Portland, Oregon, USA, 19–23 June 2004, volume 2, pages 2092–2099. IEEE Press.
- [135] Andy Song, Vic Ciesielski, and Hugh E. Williams. Texture classifiers generated by genetic programming. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, Honolulu, Hawaii, 12–17 May 2002, volume 1, pages 243–248. IEEE Press.
- [136] Andy Song, Thomas Loveard, and Vic Ciesielski. Towards genetic programming for texture classification. In Markus Stumptner, Dan Corbett, and Mike Brooks, editors, *AI 2001: Advances in Artificial Intelligence: Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, Adelaide, Australia, 10–14 December 2001, volume 2256 of *Lecture Notes in Computer Science*, pages 461–472. Springer-Verlag.
- [137] Terence Soule and James A. Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309, Winter 1998. The MIT Press.
- [138] Terence Soule and Robert B. Heckendorn. Function sets in genetic programming. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, 7–11 July 2001, page 190. Morgan Kaufmann Publishers, Inc.
- [139] Robert D. Stewart, Iris Fermin, and Manfred Opper. Region growing with pulse-coupled neural networks: An alternative to seeded region growing. *IEEE Transactions on Neural Networks*, 13(6):1557–1562, November 2002. IEEE Computational Intelligence Society.
- [140] H. Stollberg, J. Boutet de Monvel, A. Holmberg, and H. M. Hertz. Wavelet-based image restoration for compact X-ray microscopy. *Journal of Microscopy*, 211(2):154–160, 2003. Blackwell Publishing.

- [141] Walter Alden Tackett. Genetic programming for feature discovery and image discrimination. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA-93)*, University of Illinois at Urbana-Champaign, Illinois, USA, 17–21 July 1993, pages 303–309. Morgan Kaufmann Publishers, Inc.
- [142] Astro Teller and Manuela Veloso. A controlled experiment: Evolution for learning difficult image classification. In Carlos Pinto-Ferreira and Nuno J. Mamede, editors, *Proceedings of the 7th Portuguese Conference on Artificial Intelligence (EPIA '95)*, Funchal, Madeira Island, Portugal, 3–6 October 1995, volume 990 of *Lecture Notes in Computer Science*, pages 165–176. Springer-Verlag.
- [143] Astro Teller and Manuela Veloso. PADO: A new learning architecture for object recognition. In Katsushi Ikeuchi and Manuela Veloso, editors, *Symbolic Visual Learning*, pages 77–112. Oxford University Press, 1997.
- [144] A. Teredesai, E. Ratzlaff, J. Subrahmonia, and V. Govindaraju. On-line digit recognition using off-line features. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP 2002)*, Ahmedabad, India, 16–18 December 2002.
- [145] W. Tong, S.T. Nugent, P.H. Gregson, G.M. Jensen, and D.F. Fay. Landmarking of cephalograms using a microcomputer system. *Computers and Biomedical Research*, 23(4):358–379, August 1990. Academic Press.
- [146] W. Tong, S.T. Nugent, G.M. Jensen, and D.F. Fay. An algorithm for locating landmarks on dental X-rays. In Yongmin Kim and Francis A. Spelman, editors, *Images of the Twenty-First Century: Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Seattle, Washington, USA, 9–12 November 1989, volume 2, pages 552–554. IEEE Press.
- [147] William Triggs, editor. *Proceedings of the Pattern Recognition and Machine Learning in Computer Vision Workshop*, Grenoble, France, 3–5 May 2004.

- [148] Leonardo Trujillo and Gustavo Olague. Using evolution to learn how to perform interest point detection. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR2006)*, Hong Kong, 20–24 August 2006, volume 1, pages 211–244. IEEE Computer Society.
- [149] Brijesh Verma and John Zakos. A computer-aided diagnosis system for digital mammograms based on fuzzy-neural and feature extraction techniques. *IEEE Transactions on Information Technology in Biomedicine*, 5(1):46–54, March 2001. IEEE Computer Society and IEEE Engineering in Medicine and Biology Society.
- [150] Joakim Waldemark, Mikael Millberg, Thomas Lindblad, and Karina Waldemark. Image analysis for airborne reconnaissance and missile applications. *Pattern Recognition Letters*, 21(3):239–251, March 2000. Elsevier Science Inc.
- [151] Karina Waldemark, Thomas Lindblad, Vlatko Bečanović, Jose L.L. Guillen, and Phillip L. Klingner. Patterns from the sky: Satellite image analysis using pulse coupled neural networks for pre-processing, segmentation and edge detection. *Pattern Recognition Letters*, 21(3):227–237, March 2000. Elsevier Science Inc.
- [152] Gang Wang and Terence Soule. How to choose appropriate function sets for GP. In Maarten Keijzer, Una-May O’Reilly, Simon M. Lucas, Ernesto Costa, and Terence Soule, editors, *Proceedings of the Genetic Programming: 7th European Conference, EuroGP 2004*, Coimbra, Portugal, 5–7 April 2004, volume 3003 of *Lecture Notes in Computer Science*, pages 198–207. Springer-Verlag.
- [153] Jen-Shiang Wang. Influences of function sets in genetic programming. In John R. Koza, editor, *Genetic Algorithms and Genetic Programming at Stanford 2003*, pages 221–229. Stanford Bookstore, Stanford, California, USA, 4 December 2003.
- [154] Shuqiang Wang, N. Barrie Jones, James B. Richardson, and Eric Klaassens. Automatic positioning of reference systems of radiographic images. In *Proceeding of the 1994 International Symposium on Speech, Image Processing and Neural Networks (ISSIPNN’94)*, Hong Kong, 13–16 April 1994, volume 1, pages 69–72. IEEE Hong Kong Chapter of Signal Processing.

- [155] Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems*. Morgan Kaufmann Publisher, Inc, 1991.
- [156] Mark Wilson, Roger Hargrave, Sunanda Mitra, Yao-Yang Shieh, and Glenn H. Roberston. Automated detection of microcalcifications in mammograms through application of image pixel remapping and statistical filter. In *Proceedings of the 11th IEEE Symposium on Computer-Based Medical Systems*, Lubbock, Texas, USA, 12–14 June 1998, pages 270–274. IEEE Computer Society.
- [157] Jay F. Winkeler and B.S. Manjunath. Genetic programming for object detection. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, California, USA, 13–16 July 1997, pages 330–335. Morgan Kaufmann Publishers, Inc.
- [158] Ian H. Witten and Eibe Frank. *Data mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, Inc., 2000.
- [159] J. Wolfer, S.H. Lee, J. Sandelski, R. Summerscales, J. Soble, and J. Robergé. Endocardial border detection in contrast enhanced echocardiographic cineloops using a pulse coupled neural network. In Alan Murray and Steven Swiryn, editors, *Proceedings of Computers in Cardiology, 1999*, Hannover, Germany, 26–29 September 1999, pages 185–188. IEEE Press.
- [160] Takeshi Yamakawa and Hideaki Kawano. Identification of a landmark in a roentgenographic cephalogram by employing the wavelet neurons. In *Proceedings of the 6th International Conference on Neural Information Processing (ICONIP'99)*, Perth, Australia, 16–20 November 1999, volume 3, pages 890–895. IEEE Press.
- [161] Weining Yue, Dali Yin, Chengjun Li, Guoping Wang, and Tianmin Xu. Locating large-scale craniofacial feature points on X-ray images for automated cephalometric analysis. In *Proceedings of the IEEE International Conference on Image Processing (ICIP 2005)*, Genoa, Italy, 11–14 September 2005, volume 2, pages 1246–1249. IEEE.

- [162] Weining Yue, Dali Yin, Chengjun Li, Guoping Wang, and Tianmin Xu. Automated 2-D cephalometric analysis on X-ray images by a model-based approach. *IEEE Transactions on Biomedical Engineering*, 53(8):1615–1623, August 2006. IEEE Engineering in Medicine and Biology Society.
- [163] Á. Zarándy, T. Roska, G. Liszka, J. Hegyesi, L. Kék, and C. Rekeczky. Design of analogic CNN algorithms for mammogram analysis. In *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA-94)*, Rome, Italy, 18–21 December 1994, pages 255–260. IEEE Press.
- [164] Mengjie Zhang. *A domain independent approach to 2D object detection based on the neural and genetic paradigms*. PhD thesis, Department of Computer Science, RMIT University, Melbourne, Victoria, Australia, 2000.
- [165] Mengjie Zhang, Peter Andrae, and Mark Pritchard. Pixel statistics and false alarm area in genetic programming for object detection. In G. Raidl, S. Cagnoni, J.J.R. Cardalda, D.W. Corne, J. Gottlieb, A. Guillot, E. Hart, C.G. Johnson, E. Marchiori, J.-A. Meyer, and M. Middendorf, editors, *Applications of Evolutionary Computing: Proceeding of EvoWorkshops 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB and EvoSTIM*, Essex, UK, 14–16 April 2003, volume 2611 of *Lecture Notes in Computer Science*, pages 455–466. Springer-Verlag.
- [166] Mengjie Zhang and Victor Ciesielski. Genetic programming for multiple class object detection. In Norman Foo, editor, *Advanced Topics in Artificial Intelligence: Proceedings of the 12th Australian Joint Conference on Artificial Intelligence (AI'99)*, Sydney, Australia, 6–10 December 1999, volume 1747 of *Lecture Notes in Computer Science*, pages 180–191. Springer-Verlag.
- [167] Mengjie Zhang and Victor Ciesielski. Using back propagation algorithm and genetic algorithms to train and refine neural networks for object detection. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications DEXA 99*, Florence, Italy, 1–3 September 1999, pages 626–635. IEEE Press.

- [168] Mengjie Zhang, Victor B. Ciesielski, and Peter Andreae. A domain-independent window approach to multiclass object detection using genetic programming. *EURASIP Journal on Applied Signal Processing*, 2003(8):841–859, 2003. Hindawi Publishing Corporation.

Appendix A

Parameter Settings

A.1 Parameter Settings: Genetic Programming

Parameters	
Population size, M	100
Maximum generation, G	100
Maximum depth, D	8
Initial maximum depth, d	6
Probability of:	
Reproduction, P_R	0.10
Crossover, P_C	0.70
Mutation, P_M	0.20
Probability of crossover at:	
Terminal	0.15
Function	0.85
Terminal Set	+, -, *, /
Function Set	PCNN derived shapes and quadrants
Tolerance (pixels)	5 (2 mm)

Table A.1: Run-time parameters used during the genetic search for evolving detection programs.

A.2 Parameter Settings: Pulse Coupled Neural Network

Landmark	SS	PCNN parameters							T
		α_F	α_L	α_T	β	V_F	V_L	V_T	
Menton	8	1	0.3	0.15	0.08	0.01	1	1	50
Sella	40	80	0.3	0.15	0.008	0.01	19	5	60
Incisal upper incisor	10	1	0.3	0.15	0.08	0.01	1	1	31
Nose mid	14	1	0.3	0.15	0.26	0.01	1	1	40
Nasion	26	1	0.3	0.15	0.03	0.01	1	1	50
A point	32	1	0.3	0.15	0.06	0.01	1	1	54
Upper lip	14	1	0.3	0.15	0.26	0.01	1	1	70

Table A.2: Parameter settings used to generate the PCNN derived shapes in Table A.3.

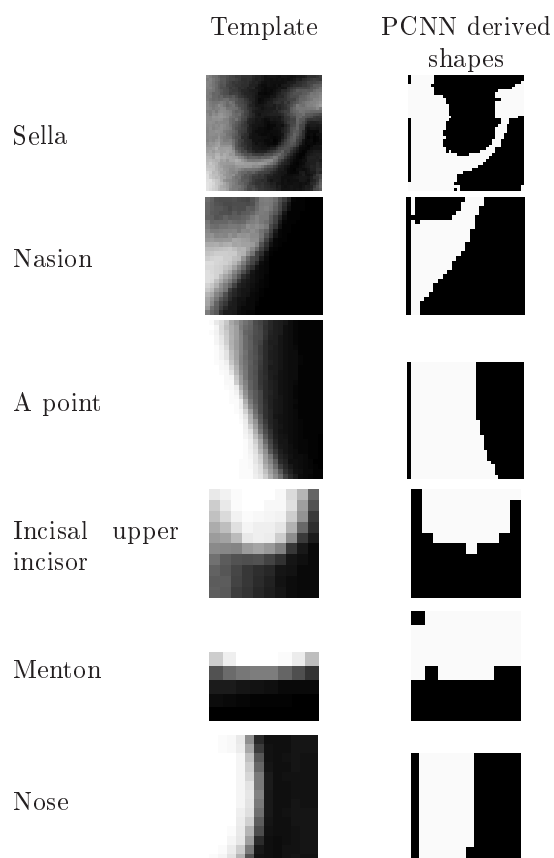


Table A.3: Templates computed using the output from a PCNN with the corresponding PCNN derived shapes. PCNN parameter settings are based on the values in Table A.2.

A.3 Cross Validation Results

Landmark	Cross validation			
	1	2	3	Avg
Menton	100	100	100	100
Sella	86.1	69.4	83.3	79.6
Incisal upper incisor	94.4	100	91.7	95.4
Nose mid	100	100	100	100
Nasion	77.8	80.6	94.4	84.3
A point	80.6	72.2	66.7	73.2
Upper lip	88.9			88.9 ¹

Table A.4: Cross validation results based on the detection performance for a range of landmark types using the parameter settings from Sections A.1 and A.2.

¹The detection performance for the upper lip is only based on a single-fold from a three-fold cross validation.

Appendix B

Test results

B.1 Sella landmark

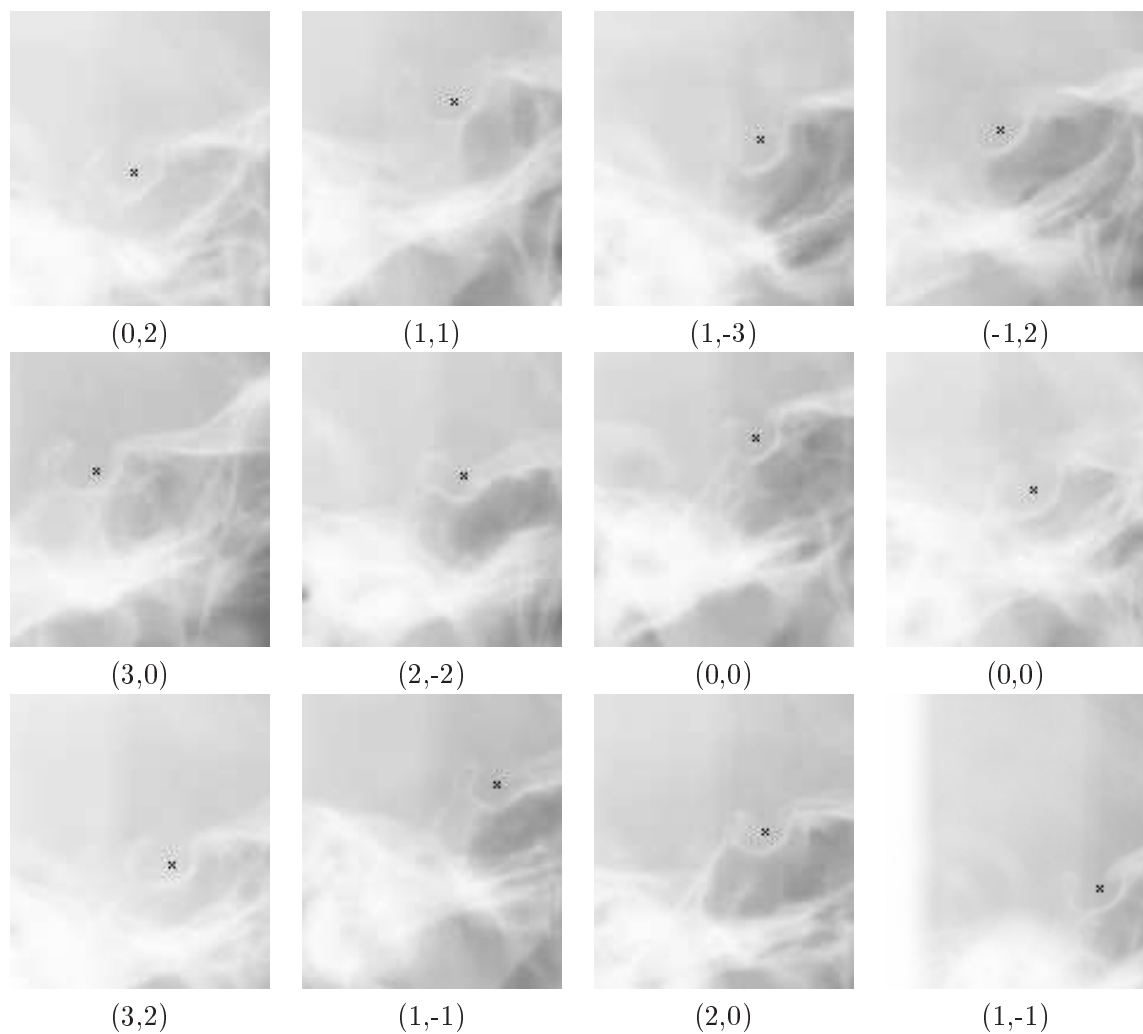


Figure B.1: A selection of images along with the predicted position for the sella landmark as is depicted by the cross. The positional error shown under each image is a measure of the predicted position relative to the actual position. Positional error is measured in pixels.

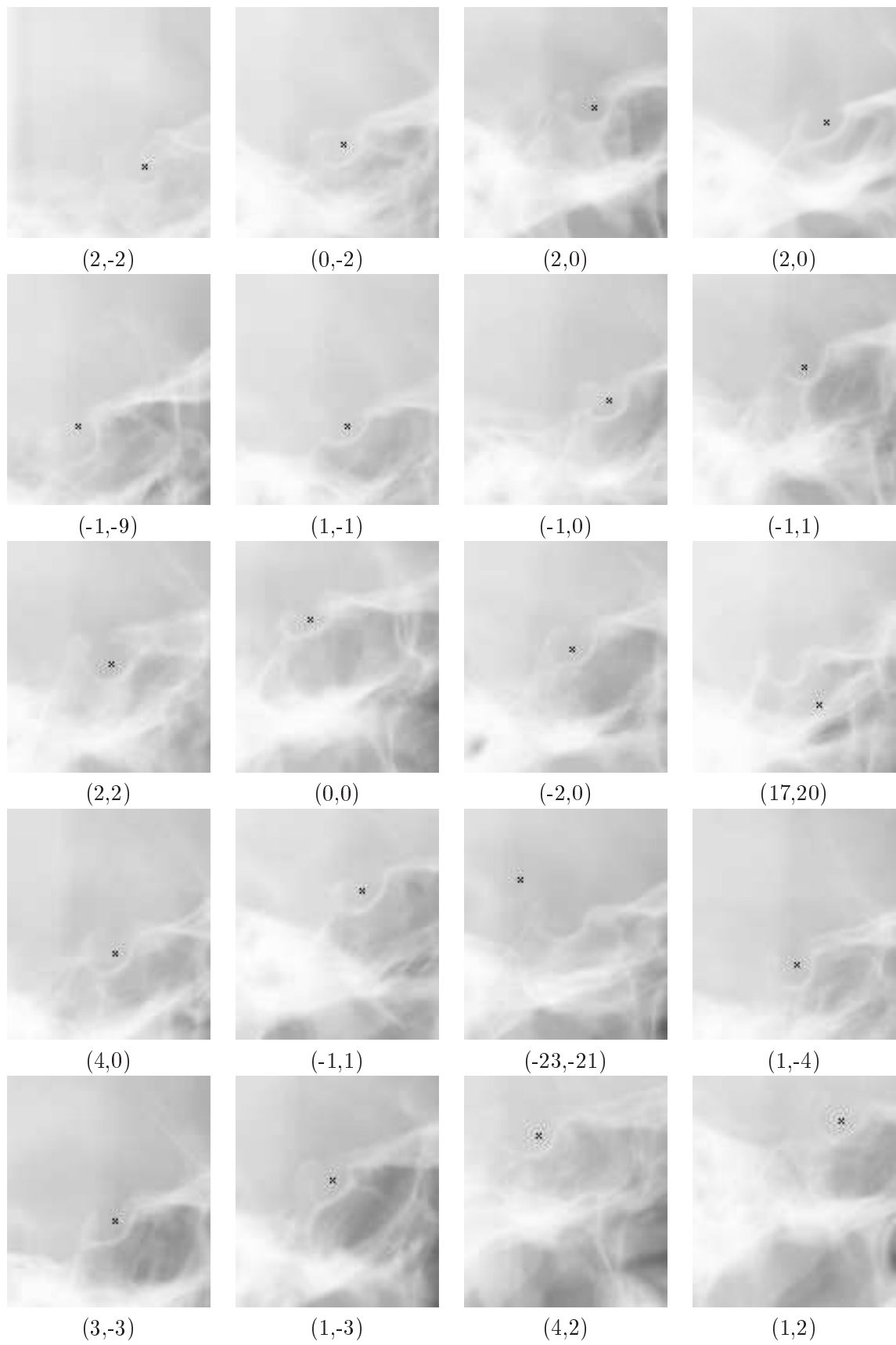


Figure B.1 (continued)

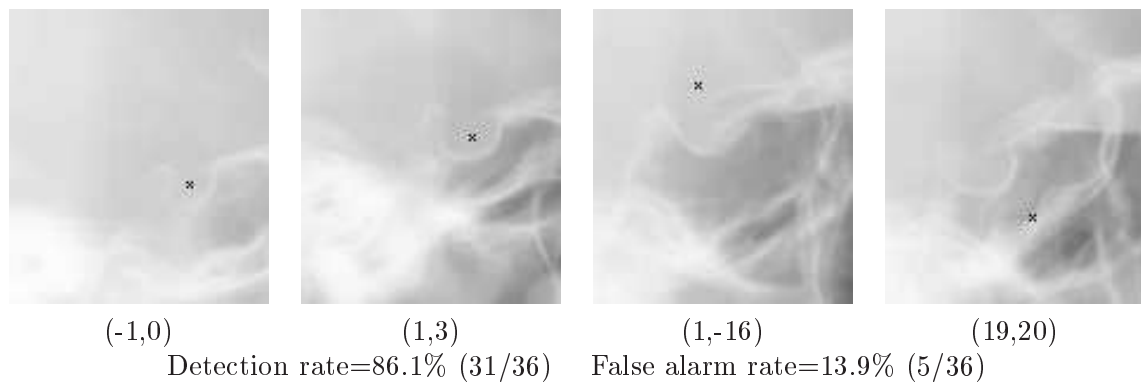


Figure B.1 (continued)

Fitness score=19.1781

(/ (/ (* (- (/ (* M₁ M₁) (+ S₃ M₂)) S₄) M₁) (+ (+ (+ (+ S₁ (+ (- M₄ M₁) M₆))) M₂) (+ (- (- M₄ M₁) M₁) M₆)) M₂)) (+ M₁ M₆))

Figure B.2: The sella landmark detection program used to predicted the position of the landmarks in Figure B.1

B.2 Nasion

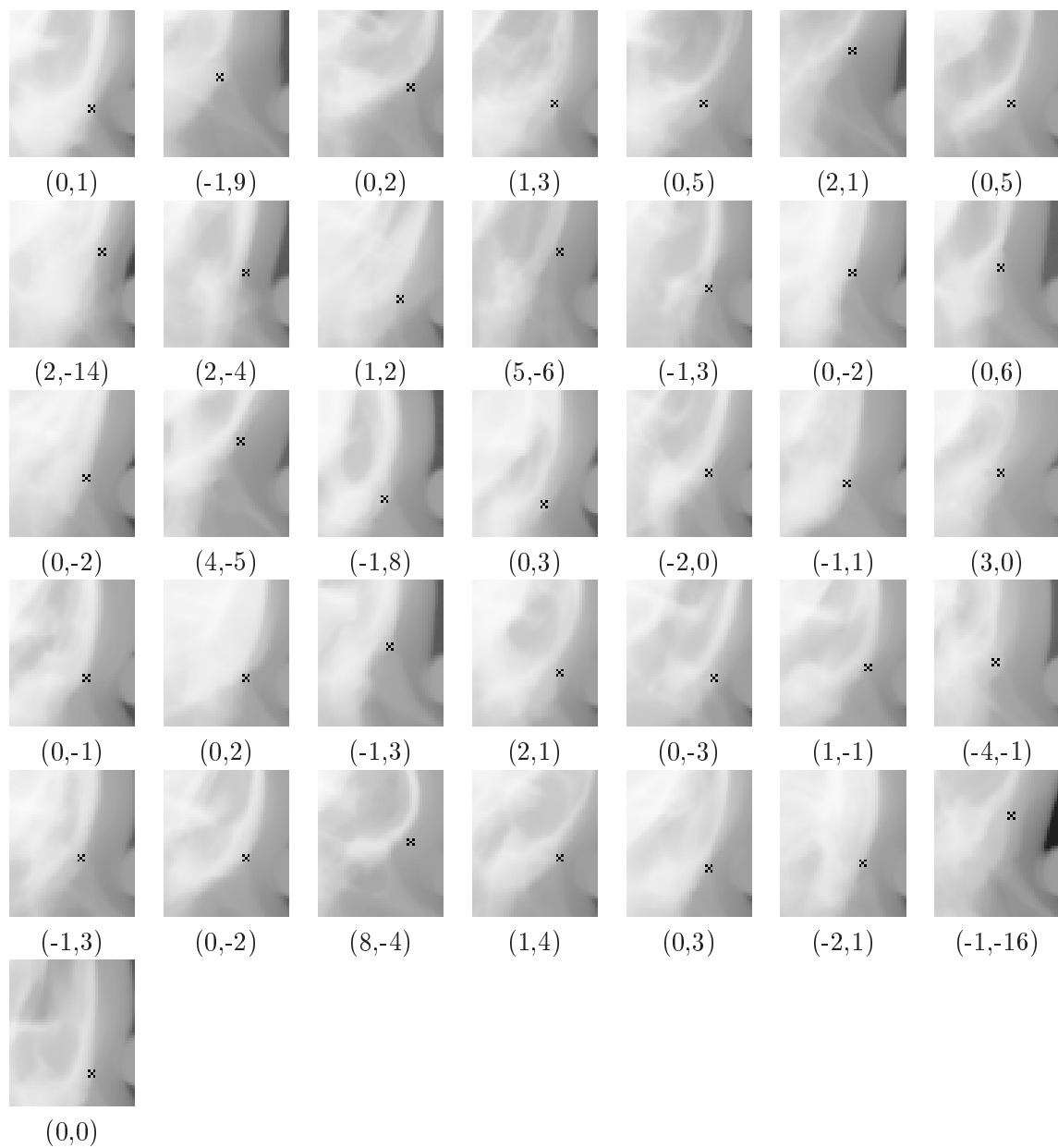


Figure B.3: A selection of images along with the predicted position for the nasion landmark as is depicted by the cross. The positional error shown under each image is a measure of the predicted position relative to the actual position. Positional error is measured in pixels.

Fitness=9.45946

(* (/ (/ (- (- (+ S₃ M₄) (/ M₇ S₂)) (+ (+ S₁ S₂) (+ M₂ S₁))) (* (* (/ M₇ S₁) S₇) (/ (- M₆ (* S₂ (+ M₅ S₄))) S₁))) (* (/ (- (* (* M₇ M₄) (* M₇ S₃)) (+ (+ (+ S₂ 127.194) M₁) S₂)) (- (/ (/ S₆ S₁) M₁) (* M₂ S₅))) (* (+ M₄ (+ (/ M₇ M₅) 127.194)) (- (- M₁ S₆) (/ S₁ 121.876)))))) (* (- (- (* (/ M₇ S₁) S₇) (/ S₁ (+ M₆ (+ S₄ 127.194)))) (/ (* M₄ S₃) (+ (/ (/ (+ S₄ S₂) S₆) S₂) (* M₆ (+ (+ M₁ M₂) S₂)))))) (- (* (- (+ M₆ M₁) (- S₇ 4.97753)) M₂) (+ S₁ S₇))))

Figure B.4: The nasion landmark detection program used to predicted the position of the landmarks in Figure B.3

B.3 A-point landmark

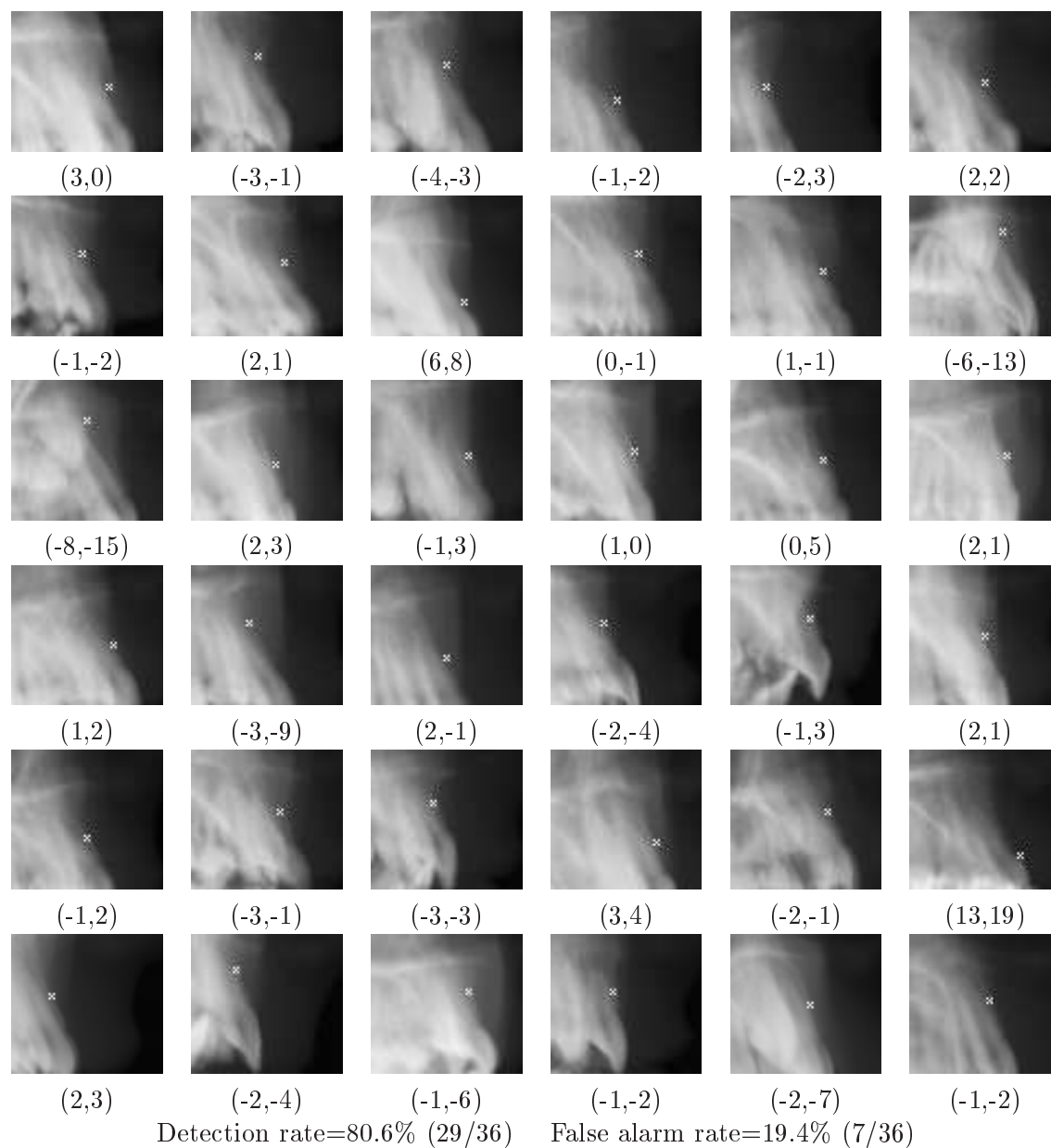


Figure B.5: A selection of images along with the predicted position for the A-point landmark as is depicted by the cross. The positional error shown under each image is a measure of the predicted position relative to the actual position. Positional error is measured in pixels.

Fitness=27.027

(* (* (/ (- 239.272 (- S₂ M₆)) (+ M₇ M₁)) (* (* (/ (+ (/ (- S₂ S₁) M₇) M₆) (+ (+ (+ S₄ M₁) S₆) S₆)) 178.822) (- S₄ 108.699))) (+ M₆ (* (+ M₆ (* (/ (- S₂ S₁) M₇) M₄) (- S₂ S₁))))))))

Figure B.6: The A-point landmark detection program used to predicted the position of the landmarks in Figure B.5

B.4 Incisal upper incisal landmark

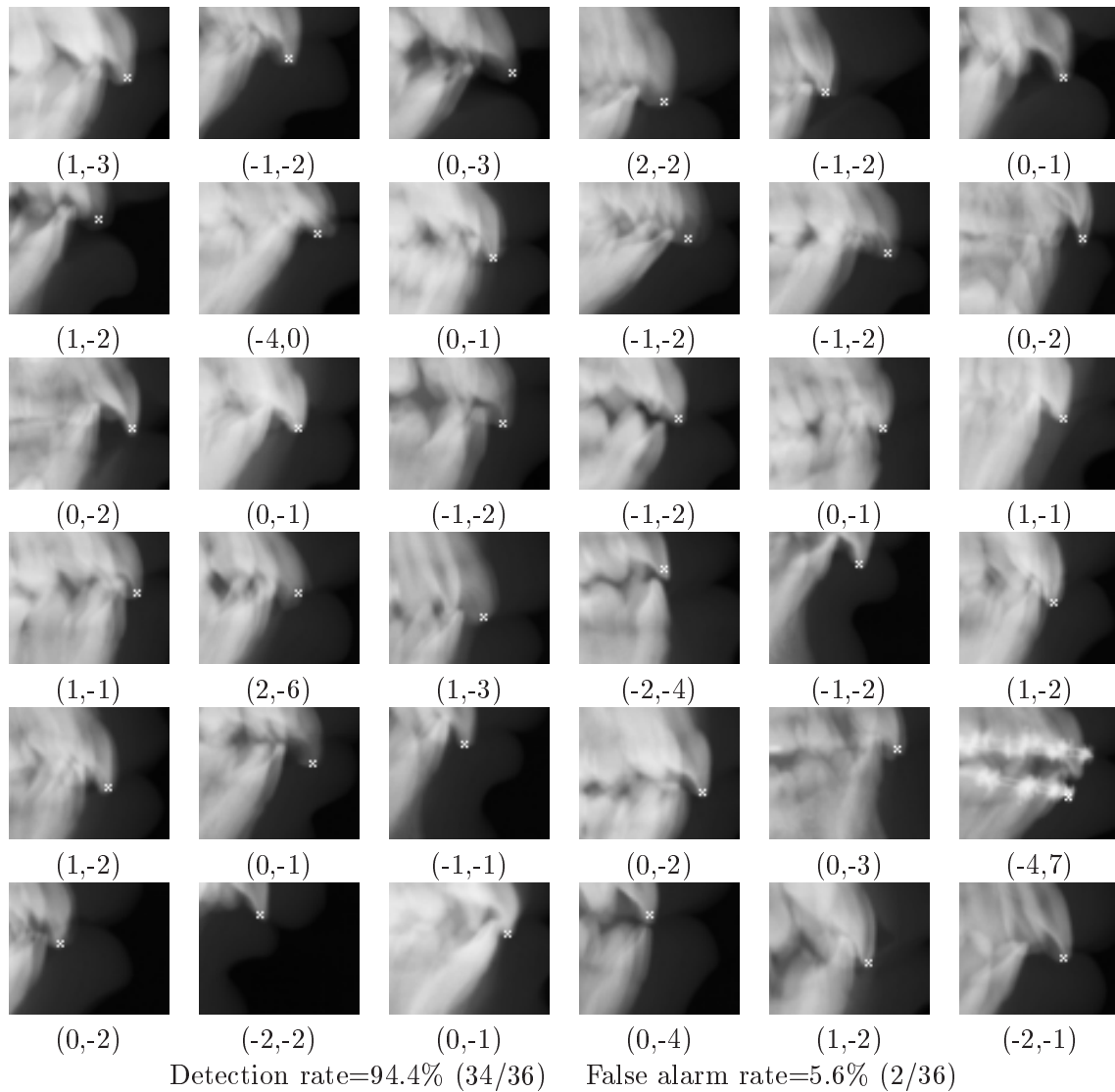


Figure B.7: A selection of images along with the predicted position for the incisal upper incisor landmark as is depicted by the cross. The positional error shown under each image is a measure of the predicted position relative to the actual position. Positional error is measured in pixels.

Fitness=4.05405

(/ (* (/ S₇ M₂) M₇) (/ (- (/ (* (* M₇ S₅) M₂) 219.642) (- (* (/ S₇ M₂) (+
 (/ S₇ M₂) (+ S₅ S₅))) 90.9186)) (* (- 76.1776 S₂ S₅)))

Figure B.8: The incisal upper incisor landmark detection program used to predicted the position of the landmarks in Figure B.7

B.5 Menton landmark

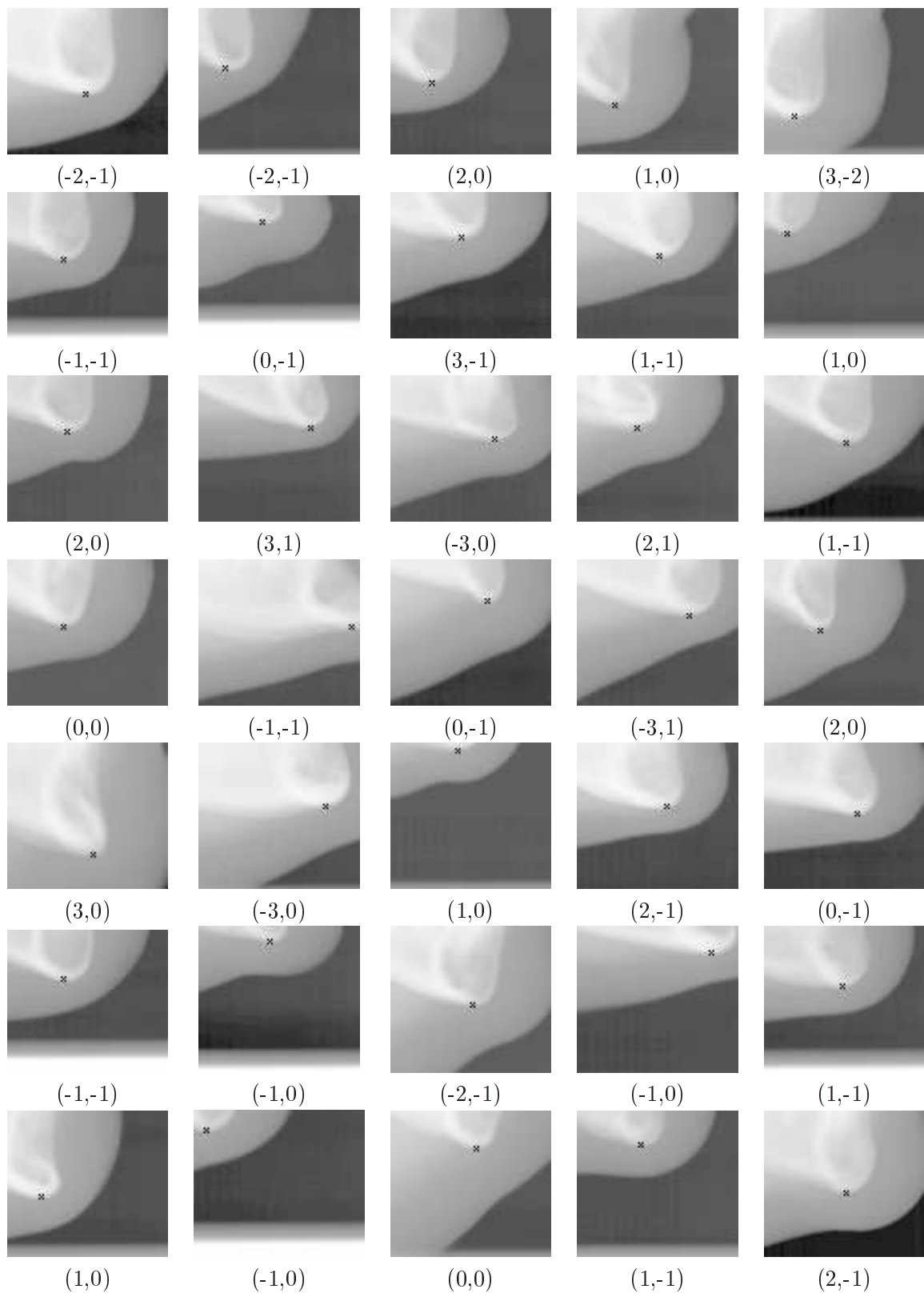
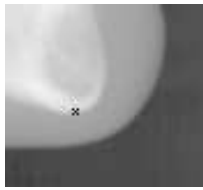


Figure B.9: A selection of images along with the predicted position for the menton landmark as is depicted by the cross. The positional error shown under each image is a measure of the predicted position relative to the actual position. Positional error is measured in pixels.



(1,-1)

Detection rate=100% (36/36) False alarm rate=0% (0/36)

Figure B.9 (continued)

Fitness=0

(- (- (* M₄ (- (- (- 98.0502 M₇) S₂) S₂)) (/ (- (* S₄ S₄) (* M₁ S₇)) (/ S₄ S₄))) (/ (- (* S₄ S₄) (* M₁ S₇)) (/ S₄ S₄)))

Figure B.10: The menton landmark detection program used to predicted the position of the landmarks in Figure B.9

B.6 Nose landmark

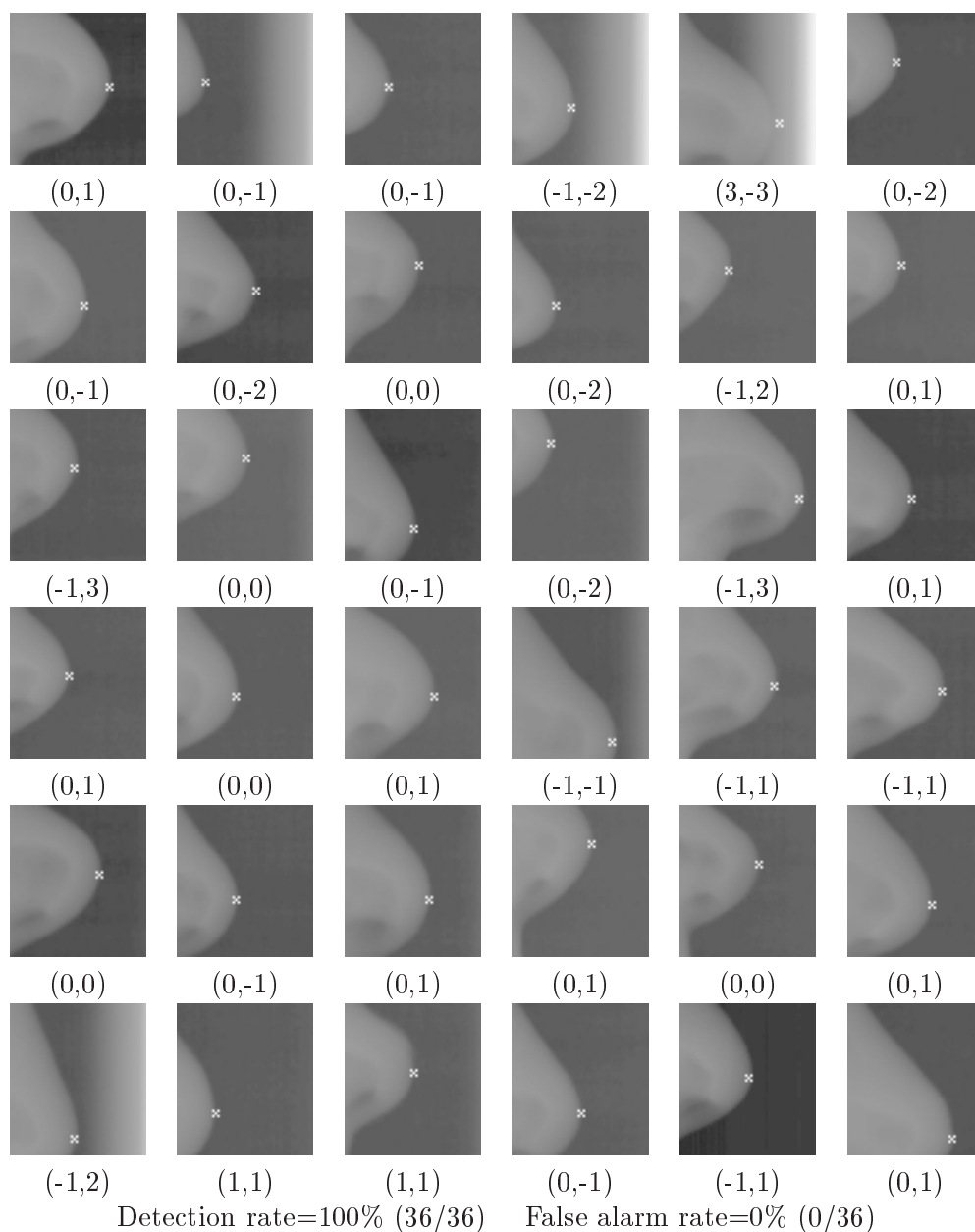


Figure B.11: A selection of images along with the predicted position for the nose landmark as is depicted by the cross. The positional error shown under each image is a measure of the predicted position relative to the actual position. Positional error is measured in pixels.

Fitness=0

```
(/ (+ (+ (- (- (- M3 M6) (- (- (- M6 M6) (- M7 M4)) M4)) (- (+ M7 S7) (- S4
M2))) (* M1 (* (- S3 S1) (+ S5 M3)))) (* (+ (- S7 S6) (+ M6 S3)) (- (/ M3
M7) (/ 10.7579 M4)))) (+ (* (+ (* S3 S7) (+ M3 M6)) (+ (+ S1 (* M3 S2)) (+
M4 S4))) (- S7 S6)))
```

Figure B.12: The nose landmark detection program used to predicted the position of the landmarks in Figure B.11