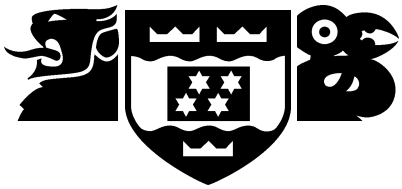
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

A Two Phase Genetic Programming Approach to Object Detection

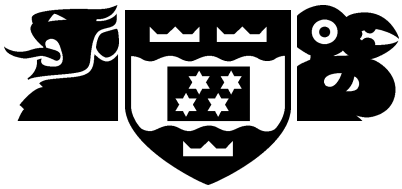
Mengjie Zhang, Peter Andreae, Urvesh Bhowan

Technical Report CS-TR-04/6
March 2004

School of Mathematical and Computing Sciences
Victoria University
PO Box 600, Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

A Two Phase Genetic Programming Approach to Object Detection

Mengjie Zhang, Peter Andreae, Urvesh Bhowan

Technical Report CS-TR-04/6
March 2004

Abstract

This paper describes two innovations that improve the efficiency and effectiveness of a genetic programming approach to object detection problems. The approach uses genetic programming to construct object detection programs that are applied, in a moving window fashion, to the large images to locate the objects of interest. The first innovation is to break the GP search into two phases with the first phase applied to a selected subset of the training data, and a simplified fitness function. The second phase is initialised with the programs from the first phase, and uses the full set of training data with a complete fitness function to construct the final detection programs. The second innovation is to add a program size component to the fitness function. This approach was applied to three object detection problems of increasing difficulty. The results indicate that the innovations increased both the effectiveness and the efficiency of the genetic programming search, and also that the genetic programming approach was more effective than a neural network approach.

Keywords Genetic programming, pixel statistics, false alarm area, program size, two-phase approach, multiclass object detection.

Author Information

Mengjie Zhang and Peter Andreae are academic staff members in computer science and Urvesh Bhowan was a postgraduate student in computer science, School of Mathematical and Computing Sciences, Victoria University of Wellington, New Zealand.

A Two Phase Genetic Programming Approach to Object Detection

Mengjie Zhang, Peter Andreae, and Urvesh Bhowan

School of Mathematical and Computing Sciences
Victoria University of Wellington,
P. O. Box 600, Wellington, New Zealand,
Email: {mengjie,pondy}@mcs.vuw.ac.nz

Abstract. This paper describes two innovations that improve the efficiency and effectiveness of a genetic programming approach to object detection problems. The approach uses genetic programming to construct object detection programs that are applied, in a moving window fashion, to the large images to locate the objects of interest. The first innovation is to break the GP search into two phases with the first phase applied to a selected subset of the training data, and a simplified fitness function. The second phase is initialised with the programs from the first phase, and uses the full set of training data with a complete fitness function to construct the final detection programs. The second innovation is to add a program size component to the fitness function. This approach was applied to three object detection problems of increasing difficulty. The results indicate that the innovations increased both the effectiveness and the efficiency of the genetic programming search, and also that the genetic programming approach was more effective than a neural network approach.

1 Introduction

Object detection tasks arise in a very wide range of applications, such as detecting faces from video images, finding tumours in a database of x-ray images, and detecting cyclones in a database of satellite images. In many cases, people (possibly highly trained experts) are able to perform the classification task well, but there is either a shortage of such experts, or the cost of people is too high. Given the amount of data that needs to be detected, automated object detection systems are highly desirable. However, creating such automated systems that have sufficient accuracy and reliability turns out to be very difficult.

Genetic programming (GP) is a relatively recent and fast developing approach to automatic programming [1, 2]. In GP, solutions to a problem are represented as computer programs. Darwinian principles of natural selection and recombination are used to evolve a population of programs towards an effective solution to specific problems. The flexibility and expressiveness of computer program representation, combined with the powerful capabilities of evolutionary search, makes GP an exciting new method to solve a great variety of problems.

There have been a number of reports on the use of genetic programming in object detection [3–9]. The approach we have used in previous work [8, 9] is to use a single stage approach (referred to as *the basic GP approach* here), where the GP is directly applied to the large images in a moving window fashion to locate the objects of interest. Past work has demonstrated the effectiveness of this approach on several object detection tasks.

However, this genetic programming approach is not without problems. One problem is that the training time was often very long, even for relatively simple object detection problems. A second problem is that the evolved programs are often hard to understand or interpret. We have identified two causes of these problems: the programs are usually quite large and contain much redundancy, and the cost of the fitness function is high. We believe that the size and redundancy of the programs contributes to the long training times and may also reduce the quality of the resulting detectors by unnecessarily increasing the size of the search space and reducing the probability of finding an optimal detector program. Evaluating the fitness of a candidate detector program in the basic GP approach involves applying the program to each possible position of a window on all the training images, which is quite expensive. An obvious solution is to apply the program to only a small subset of the possible window positions, but it is not obvious how to choose the subset. A poor choice could bias the evolution towards programs that are sub-optimal on the real data.

This paper describes two innovations on the basic GP approach to address these problems. The first is to split the GP evolution into two phases, using a different fitness function and just a subset of the training data in the first phase. The second is to augment the fitness function in the second phase by a component that biases the evolution towards smaller, less redundant programs. We consider the effectiveness and efficiency of this approach by comparing it with the basic GP approach and a neural network approach. We also examine the comprehensibility of the evolved genetic programs.

The rest of the paper is organised as follows. Section 2 describes the main aspects of this approach. Section 3 describes the three image data sets and section 4 presents the experimental results. Section 5 draws the conclusions and gives future directions.

2 The Approach

2.1 Overview of the Approach

Figure 1 shows an overview of this approach, which has two phases of learning and a testing procedure. In the first learning phase, the evolved genetic programs were initialised randomly and trained on object examples cut out from the large images in the training set. This is just an object classification task, which is simpler than the full object detection task. This phase therefore uses a fitness function which maximises classification accuracy on the object cutouts.

In the second phase, a second GP process is initialised with the programs generated by the first phase, and trained on the full images in the training set by

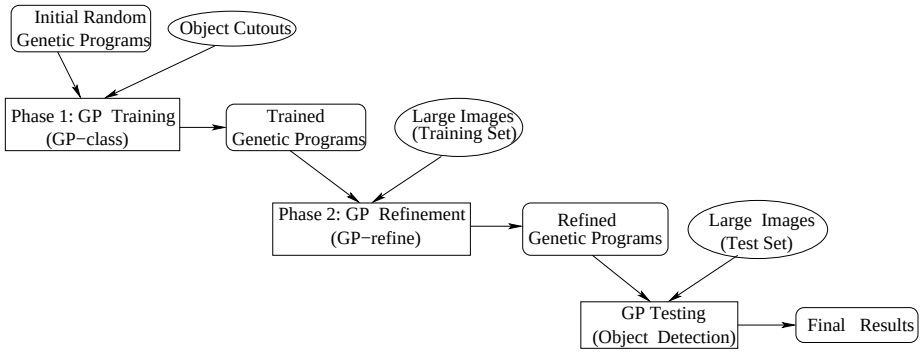


Fig. 1. An overview of the two phase GP approach.

applying the programs to a square input field ("window") that was moved across the images to detect the objects of interest. This phase uses a fitness function that maximises *detection* performance on the large images in the training set. In the test procedure, the best refined genetic program is then applied to the entire images in the test set to measure object detection performance.

Because the object classification task is simpler than the object detection task, we expect the first phase to be able to find good genetic programs much more rapidly and effectively than the second phase. Also, the fitness function is much easier to evaluate, so that a more extensive evolution can be performed in the same time. Although simpler, the object classification task is closely related to the detection task, so we believe that the genetic programs generated by the first phase are likely to be very good starting points for the second phase, allowing the more expensive evolutionary process to concentrate its effort in the more optimal part of the search space.

Since the number of possible programs increases exponentially with the size of the programs, the difficulty of finding an optimal program also increases with the size of the programs. In the second phase, we added a program size component to the fitness function to bias the search towards simpler functions, which we expected would increase both the efficiency and the effectiveness of the evolutionary search. It will also have a tendency to remove redundancy (since a program with redundancy will be less fit than an equivalent program with the redundancy removed), making the programs more comprehensible.

2.2 Program Structure, Terminal Set, and Function Set

In this system, we used tree structures to represent genetic programs.

For object detection problems, terminals generally correspond to image features. Instead of using global features of an entire input image window, we used a number of statistical properties of local square and circular region features as terminals, as shown in figure 2. The first terminal set consists of the means and standard deviations of a series of concentric square regions centred in the input image window, which was used in the *shape* data set (see section 3). The

second terminal set consists of the means and standard deviations of a series of concentric circular regions, which was used in the two coin data sets. For each terminal set, we also used a random constant as an additional terminal.

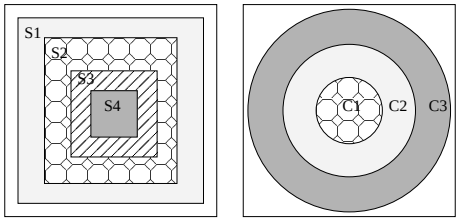


Fig. 2. Local square and circular features as terminals.

In the function set, the four standard arithmetic operators and a conditional operator were used to form the non-terminal nodes:

$$FuncSet = \{+, -, *, /, if\}$$

The $+$, $-$, and $*$ operators have their usual meanings — addition, subtraction and multiplication, while / represents "protected" division which is the usual division operator except that a divide by zero gives a result of zero. Each of these functions takes two arguments. The *if* function takes three arguments. The first argument, which can be any expression, constitutes the condition. If the first argument is positive, the *if* function returns its second argument; otherwise, it returns its third argument. The *if* function allows a program to contain a different expression in different regions of the feature space, and allows discontinuous programs, rather than insisting on smooth functions.

2.3 Object Classification Strategy

The output of a genetic program is a floating point number. Generally genetic programs can perform one class object detection tasks quite well where the division between positive and negative numbers of a genetic program output corresponds to the separation of the objects of interest (of a single class) from the background (non-objects). However, for multiple class object detection problems, where two or more classes of objects of interest are involved, the standard genetic programming classification strategy mentioned above cannot be applied.

In this approach, we used a different strategy called *program classification map*, as shown in equation 1, for the multiple class object detection problems [10]. Based on the output of an evolved genetic program, this map can identify which class of the object located in the current input field belongs to. In this map, m refers to the number of object classes of interest, v is the output value of the evolved program and T is a constant defined by the user, which plays a role of a threshold.

$$\text{Class} = \begin{cases} background, & v \leq 0 \\ \text{class 1}, & 0 < v \leq T \\ \text{class 2}, & T < v \leq 2T \\ \cdots & \cdots \\ \text{class i}, & (i-1) \times T < v \leq i \times T \\ \cdots & \cdots \\ \text{class m}, & v > (m-1) \times T \end{cases} \tag{1}$$

2.4 Fitness Function

We used two fitness functions for the two learning phases. In the first phase, we used the classification accuracy directly as the fitness function to maximise object classification accuracy. In the second phase, we used a multi-objective fitness function.

The goal of object detection is to achieve both a high detection rate and a low false alarm rate. In genetic programming, this typically needs a multi-objective fitness function. A fitness function we used in previous work [9] is:

$$fitness(DR, FAR) = K_1 * (1 - DR) + K_2 * FAR + K_3 * FAA \tag{2}$$

where DR is the Detection Rate (the number of small objects correctly reported by a detection system as a percentage of the total number of actual objects in the images), FAR is the False Alarm Rate (also called *false alarms per object*, the number of non-objects incorrectly reported as objects by a detection system as a percentage of the total number of actual objects in the images), and FAA is the False Alarm Area (the number of false alarm pixels which are not object centres but are incorrectly reported as object centres before clustering). The parameters K_1, K_2 and K_3 reflect the relative weighting of DR, FAR, and FAA.

A problem with this fitness function is that it has no bias towards smaller programs. When a short program and a long program produce the same detection rate and the same false alarm rate (perhaps because one is the same as the other but with additional, redundant elements), the GP system will randomly choose one for reproduction, mutation or crossover during the evolutionary process. If the longer programs are selected, the evolutionary process will tend to continue to produce long programs, which will slow the evolution down and decrease the chance of finding a good program. Also, the longer programs are usually very difficult to interpret. Therefore, for the experiments in this paper, we have augmented above fitness function with a *program size* component:

$$fitness = K_1 \cdot (1 - DR) + K_2 \cdot FAR + K_3 \cdot FAA + K_4 \cdot ProgSize \tag{3}$$

where $progSize$ is the number of terminals and non-terminals in the program.

Evaluating the fitness of a program requires several steps. The first is to apply the program as a moving $n \times n$ window template (where n is the size of the input image window) to each of the training images and label each possible window position with the class of object determined by the program (including "background"). We refer to the data structure containing these labels as a "detection map".

The detection map will typically contain a cluster of pixels labelled by a class name at places that the program could detect an object. Since the pixels in a cluster typically are part of the same object, the second step clusters the pixels with a simple scanning algorithm to identify the clusters corresponding to distinct objects. The third step matches the locations of these detected objects against the locations of the true objects in the images to determine the number of objects that were correctly identified, and the number of false alarms. The final step uses these numbers and the size of the program to compute the value of the fitness function.

Notice that adding the program size constrain to the fitness function is a kind of *parsimony pressure* technique [11–13]. Early work on this issue resulted in diverse opinions: some researchers think using parsimony pressure could improve performance [12], while some others thinks this could lead to premature convergence [13]. Although our approach is different from the early work, it might still face a risk of early convergence. Therefore, we used a very small weight (K_4) for the program size in our fitness function relative to K_1 and K_2 (see table 1).

2.5 Parameters and Termination Criteria

The ramped half-and-half method [1, 2] was used for generating the programs in the initial population and for the mutation operator. The proportional selection mechanism and the reproduction [10], crossover and mutation operators [1] were used in the learning process.

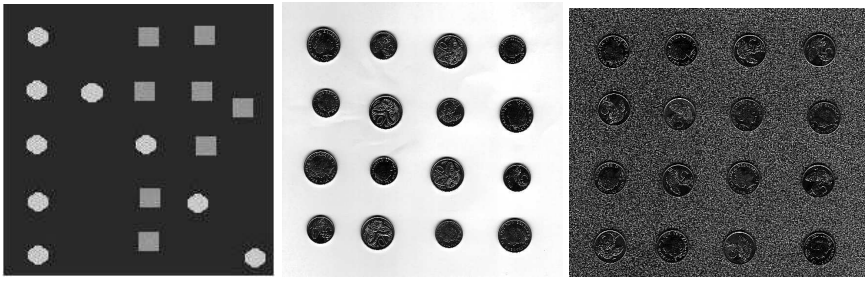
Important parameter values used in the experiments are shown in table 1.

Table 1. Parameters used for GP training for the three databases.

Parameter Kind	Parameter Name	Shape	Coins	Heads/tails
Search Parameters	population-size	800	1000	1600
	initial-max-depth	2	2	5
	max-depth	6	7	8
	max-generations	50	150	200
	input-size	20×20	72×72	62×62
Genetic Parameters	reproduction-rate	2%	2%	2%
	cross-rate	70%	70%	70%
	mutation-rate	28%	28%	28%
Fitness Parameters	T	100	80	80
	K1	5000	5000	5000
	K2	100	100	100
	K3	10	10	10
	K4	1	1	1

The learning/evolutionary process is run for a fixed number (*max-generations*) of generations, unless it finds a program that solves the problem perfectly (100% detection rate and no false alarms), or there is no increase in the fitness for 10 generations, at which point the evolution is terminated early.

3 Image Data Sets



No. of images: 10 Object size:18×18	No. of images: 20 Object size:70× 70	No. of images: 20 Object size:60×60
(Shape)	(Coins)	(Heads/tails)

Fig. 3. Object Detection Problems

We used three data sets in the experiments. Example images are given in figure 3. These data sets provide object detection problems of increasing difficulty. Data set 1 (Shape) was generated to give well defined objects against a uniform background. The pixels of the objects were generated using a Gaussian generator with different means and variances for different classes. There are two classes of small objects of interest in this database: circles and squares. Data set 2 (Coins) was intended to be somewhat harder and consists of scanned images of New Zealand coins. There are two object classes of interest: the 5 cent coins and 10 cent coins. These coins are a mixture of head up or tail up and accordingly has a greater variance than data set 1. The objects in each class have a similar size but are located at arbitrary positions and with different rotations. Data set 3 (Heads/tails) also contains two object classes of interest, but the detection task is significantly more difficult. The task is detecting the head side and the tail side of New Zealand 5 cent coins with various orientations from a non-uniform background. Given the low resolution of the images, this detection task is very difficult — even humans cannot distinguish the classes perfectly.

In the experiments, we used one, three, and five images as the training set and used five, ten and ten images as the test set for the *Shape, Coins*, and *Heads/tails* data sets, respectively.

4 Results and Discussion

4.1 Object Detection Results

The detection results of the two phase GP approach for the three image data sets are shown in table 2. These results are compared with the basic GP approach [8, 14] and a neural network approach [15] using the same set of features. For

Table 2. Object detection results achieved by different approaches.

Image Data Set		Shape	Coins	Heads/tails	
				heads	tails
Best Detection Rate(%)		100	100	100	100
Best False Alarm Rate (%)	Two-phase GP Approach	0	0	0	55
	Basic GP Approach	0	0	0	100
	Neural Networks	0	0	9.4	134.1

all cases, the experiments were repeated 10 times and the average results on the *test set* were presented.

As can be seen from table 2, all the three approaches achieved ideal results for the shape and the Coins data sets, reflecting the fact that the detection problems in the two data sets are relatively easy and that the two terminal sets are appropriate for the two data sets (note that other terminal sets did not achieve ideal results [14], but this is beyond the scope of this paper). For the difficult Heads/tails data set, none of the three methods resulted in ideal performance. However, the two phase GP approach described in this paper achieved the best performance. Notice also that both GP approaches achieved better results than the neural network approach using the same set of features.

4.2 Training Time and Program Size

Although both of the GP approaches achieved better results than the neural networks overall, the time spent on the training/refining process are quite different. For the *Coins* data set, for example, the basic GP approach used 17 hours on average to find a good genetic program, whereas the two phase GP approach used only 11 hours on average. For the *Heads/tails* data set, the two phase GP approach found good programs after 23 hours on average (of which the first phase only took only two to three minutes). The basic GP approach, on the other hand, took an average of 45 hours. The first phase is so fast because the size of the training data set is small, and the task of discriminating the classes of objects (when centered in the input window) is quite simple. However, the programs it finds appear to be very good starting points for the more expensive second phase, which enables the evolution in the second phase to concentrate its search in a much more promising part of the search space.

In addition, the sizes of the programs (the number of terminals plus the number of functions in a program) evolved by the two phase GP approach were also found to be shorter than those evolved by the basic GP approach. For the *Coins* data set, for example, the program size in the two phase GP approach averages 56 nodes, in contrast to 107 nodes for the basic GP approach. Both the good initial programs and the bias towards smaller programs would contribute to this result; we have not yet identified which of the factors is the most important.

4.3 Comprehensibility of Genetic Programs

To check the effectiveness of the new fitness function at improving the comprehensibility of the programs, an evolved genetic program in the *shape* data set is shown below:

(/ (if (/ (- $F_{4\mu}$ T) $F_{4\mu}$) $F_{3\mu}$ (* (- $F_{4\mu}$ $F_{2\mu}$) $F_{1\sigma}$)) (/ $F_{4\mu}$ $F_{4\mu}$))

This program detector can be simplified as follows:

(if (- $F_{4\mu}$ T) $F_{3\mu}$ (* (- $F_{4\mu}$ $F_{2\mu}$) $F_{1\sigma}$))

where $F_{i\mu}$ and $F_{i\sigma}$ are the mean and standard deviation of region i (see figure 2, left) of the window, respectively, and T is a predefined threshold. This program can be translated into the following rule:

```
if (F4μ > T) then
  value = F3μ;
else
  value = (F4μ - F2μ) * F1σ;
```

If the sweeping window is over the background only, $F_{4\mu}$ would be smaller than the threshold (100 here), the program would execute the "else" part. Since $F_{4\mu}$ is equal to $F_{2\mu}$ in this case, the program output will be zero. According to the classification strategy — object classification map, this case would be correctly classified as *background*. If the input window contains a portion of an object of interest and some background, $F_{4\mu}$ would be smaller than $F_{2\mu}$, which results in a negative program output, corresponding to class *background*. If $F_{4\mu}$ is greater than the threshold T, then the input window must contain an object of interest, either for *class1* or for *class2*, depending the value of $F_{3\mu}$.

While this program detector can be relatively easily interpreted and understood, the programs obtained using the old fitness function are generally hard to interpret due to the length of the programs and the redundancy. By carefully designing the fitness function to constrain the program size, the evolved genetic programs appear to be more comprehensible.

5 Conclusions

The goal of this paper was to investigate the effectiveness and efficiency of the two phase GP approach and the comprehensibility of genetic programs evolved using the new fitness function with the constraints on program size. The approach was tested on three object detection problems of increasing difficulty and achieved good results.

We developed a two phase approach to object detection using genetic programming. Our results suggest that the two phase approach is more effective and more efficient than the basic GP approach and more effective than a neural network approach using the same set of features.

We modified the fitness function by including a measure of program size. This resulted in genetic program detectors that were better quality and more comprehensible. It also reduced the search computation time.

Although this approach considerably shortens the training times, the training process is still relatively long. We intend to explore better classification strategies and add more heuristics to the genetic beam search to the evolutionary process.

While the programs evolved by the two phase GP approach with the new fitness function are considerably shorter than the basic GP approach, they usually still contain some redundancy. Although we suspect that this redundancy reduces the efficiency and the effectiveness of the evolutionary search, it is also possible that redundancy plays an important role in the search. We are experimenting with simplification of the programs during the evolutionary process to remove the redundancy, and will be exploring whether it reduces training speed and improves program quality.

References

1. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications. Morgan Kaufmann Publishers (1998)
2. Koza, J.R.: Genetic programming : on the programming of computers by means of natural selection. Cambridge, Mass. : MIT Press, London, England (1992)
3. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In Forrest, S., ed.: Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann (1993) 303–309
4. Benson, K.: Evolving finite state machines with embedded genetic programming for automatic target detection within SAR imagery. In: Proceedings of the 2000 Congress on Evolutionary Computation, IEEE Press (2000) 1543–1549
5. Graae, C.T.M., Nordin, P., Nordahl, M.: Stereoscopic vision for a humanoid robot using genetic programming. In Cagnoni, S., et al. eds.: Real-World Applications of Evolutionary Computing. Volume 1803 of LNCS., Springer-Verlag (2000) 12–21
6. Howard, D., Roberts, S.C., Ryan, C.: The boru data crawler for object detection tasks in machine vision. In Cagnoni, S., et al. eds.: Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002. Volume 2279 of LNCS. Springer-Verlag (2002) 220–230
7. Lindblad, F., Nordin, P., Wolff, K.: Evolving 3d model interpretation of images using graphics hardware. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation, IEEE Press (2002)
8. Zhang, M., Ciesielski, V.: Genetic programming for multiple class object detection. In Foo, N., ed.: Proceedings of the 12th Australian Joint Conference on Artificial Intelligence, LNAI Vol. 1747, Springer-Verlag (1999) 180–192
9. Zhang, M., Andreae, P., Pritchard, M.: Pixel statistics and false alarm area in genetic programming for object detection. In Cagnoni, S., ed.: Applications of Evolutionary Computing, Lecture Notes in Computer Science, LNCS Vol. 2611, Springer-Verlag (2003) 455–466
10. Zhang, M., Ciesielski, V., Andreae, P.: A domain independent window-approach to multiclass object detection using genetic programming. EURASIP Journal on Signal Processing, Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis **2003** (2003) 841–859

11. Smith, P.W.H.: Controlling code growth in genetic programming. In John, R., Birkenhead, R., eds.: Advances in Soft Computing, De Montfort University, Leicester, UK, Physica-Verlag (2000) 166–171
12. Dallaway, R.: Genetic programming and cognitive models. Technical Report CSRP 300, School of Cognitive & Computing Sciences, University of Sussex,, Brighton, UK (1993)
13. Soule, T., Foster, J.A.: Effects of code growth and parsimony pressure on populations in genetic programming. Evolutionary Computation **6** (1998) 293–309
14. Bhowan, U.: A domain independent approach to multi-class object detection using genetic programming. BSc Honours research project/thesis, school of mathematical and computing sciences, victoria university of wellington. (2003)
15. Ny, B.: Multi-cclass object classification and detection using neural networks. BSc Honours research project/thesis, school of mathematical and computing sciences, victoria university of wellington (2003)