Electronic Letters on Computer Vision and Image Analysis 6(1):27-43, 2007

Genetic Programming for Object Detection: A Two-Phase Approach with an Improved Fitness Function

Mengjie Zhang*, Urvesh Bhowan, Bunna Ny

School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, PO Box 600, Wellington, New Zealand

Received 17 August 2006; revised 10 January 2007; accepted 27 March 2007

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 is examined and compared with a neural network approach on three object detection problems of increasing difficulty. The results suggest that the innovations increase both the effectiveness and the efficiency of the genetic programming search, and also that the genetic programming approach outperforms a neural network approach for the most difficult data set in terms of the object detection accuracy.

Key Words: Artificial Intelligence approaches to Computer Vision, Object Recognition, Image Analysis, Genetic Programming, Neural Networks.

1 Introduction

Object detection and recognition tasks arise in a very wide range of applications [1, 2, 3, 4, 5, 6, 7], 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 [8, 9]. 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.

Correspondence to: <mengjie@mcs.vuw.ac.nz>

Recommended for acceptance by M. Bicego
ELCVIA ISSN:1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain

There have been a number of reports on the use of genetic programming in object detection [10, 11, 12, 13, 14, 15, 16]. The approach we have used in previous work [15, 16] 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.

While showing promise, this genetic programming approach still has some 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.

The goal of this paper is to investigate a study on improving GP techniques for object detection (rather than investigate an application of GP for object detection). Specifically, we investigate two innovations on the basic GP approach to address the problems described above. 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. We also examine the comprehensibility of the evolved genetic programs.

The rest of the paper is organised as follows. Section 2 gives some essential background of object detection and recognition and GP related work to object detection. Section 3 describes the main aspects of this approach. Section 4 describes the three image data sets and section 5 presents the experimental results. Section 6 draws the conclusions and gives future directions.

2 Background

This section provides some essential background, including a brief overview of the object recognition and detection with related methods, and a brief overview of related work in GP to object detection and recognition and image analysis.

2.1 Object Detection/Recognition and Related Methods

The term *object detection* here refers to the detection of small objects in large images. This includes both *object classification* and *object localisation*. *Object classification* refers to the task of discriminating between images of different kinds of objects, where each image contains only one of the objects of interest. *Object localisation* refers to the task of identifying the positions of all objects of interest in a large image. The object detection problem is similar to the commonly used terms *automatic target recognition* and *automatic object recognition*.

Traditionally, most research on object recognition involves four stages: *preprocessing, segmentation, feature extraction* and *classification* [17, 18]. The preprocessing stage aims to remove noise or enhance edges. In the segmentation stage, a number of coherent regions and "suspicious" regions which might contain objects are usually located and separated from the entire images. The feature extraction stage extracts domain specific features from the segmented regions. Finally, the classification stage uses these features to distinguish the classes of the objects of interest. The features extracted from the images and objects are generally domain specific such as high level relational image features. Data mining and machine learning algorithms are usually applied to object classification.

Object detection and recognition has been of tremendous importance in many application domains. These domains include military applications [19, 20, 10], shape matching [2], human face and visual recognition [1, 5, 21, 22], natural scene recognition [4], agricultural product classification [23], handwritten character recognition [24, 25], medical image analysis [26], postal code recognition [27, 28], and texture classification [29].

Since the 1990s, many methods have been employed for object recognition. These include different kinds of neural networks [30, 31, 28, 32, 33], genetic algorithms [34, 35], decision trees [36], statistical methods such as Gaussian models and Naive Bayes [37, 36], support vector machines [37, 36], genetic programming [38, 13, 22, 39], and hybrid methods [40, 41, 42].

2.1.1 Performance Evaluation

Object detection performance is usually measured by *detection rate* and *false alarm rate*. The detection rate (DR) refers to the number of small objects correctly reported by a detection system as a percentage of the total number of actual objects in the image(s). The false alarm rate (FAR), also called false alarms per object [43], refers to 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 image(s). Note that the detection rate is between 0 and 100%, while the false alarm rate may be greater than 100% for difficult object detection problems.

2.2 GP Main Characteristics: GP vs GAs

GP is an approach to automatic programming, in which a computer can construct and refine its own programs to solve specific tasks. First popularised by Koza [9] in 1992, GP has become another main genetic paradigm in evolutionary computation (EC) in addition to the well known *genetic algorithms* (GAs).

Compared with GAs, GP has a number of characteristics. While the standard GAs use bit strings to represent solutions, the forms evolved by GP are generally trees or tree-like structures. The standard GA bit strings use a fixed length representation while the GP trees can vary in length. While the GAs use a binary alphabet to form the bit strings, the GP uses alphabets of various sizes and content depending on the problem domain. These trees are made up of internal nodes and leaf nodes, which have been drawn from a set of primitive elements that are relevant to the problem domain. Compared with a bit string to represent a given problem, the trees can be much more flexible.

2.3 GP Related Work to Object Detection

Since the early 1990s, there has been only a small amount of work on applying genetic programming techniques to object classification, object detection and other image recognition problems. This in part reflects the fact that genetic programming is a relatively young discipline compared with, say, neural networks and genetic algorithms.

In terms of the number of classes in object classification, there are two categories: *binary classification problems*, where there are only two classes of objects to be classified, and *multi-class classification problems*, where more than two classes of images are involved. While GP has been widely applied to binary classification problems [38, 44, 10, 45], it has also been applied to multi-class classification problems [46, 22, 47, 16, 15, 48].

In terms of the representation of genetic programs, different forms of genetic programs have been developed in GP systems for object classification and image recognition. The main program representation forms include tree or tree-like or numeric expression programs [8, 49, 46, 48], graph based programs [8], linear GP [50], linear-graph GP [51], and grammar based GP [52].

The use of GP in object/image recognition and detection has also been investigated in a variety of application domains. These domains include military applications [10, 45], English letter recognition [24], face/eye detection and recognition [53, 22, 39], vehicle detection [38, 13] and other vision and image processing problems [12, 54, 9, 14, 55, 56].

Since the work to be presented in this paper focuses on the use of genetic programming techniques for object detection, table 1 lists the recent research to overview the GP related work based on the applications and the first authors.

Problems	Applications	Authors	Source
Object Detection	Orthodontic landmark detection	Ciesielski et al.	[63]
	Ship detection	Howard et al.	[38]
	Mouth detection	Isaka	[64]
	Small target detection	Benson	[11]
	Vehicle detection	Howard et al.	[13]
	Medical object detection	Zhang et al.	[48, 15]
Object Classification	Tank detection	Tackett	[10, 45]
	Letter recognition	Andre	[24]
		Koza	[49]
	Face recognition	Teller et al.	[22]
	Small target classification	Stanhope et al.	[57]
		Winkeler et al.	[39]
	Shape recognition	Teller 7et al.	[47]
	Eye recognition	Robinson et al.	[53]
	Texture classification	Song et al.	[58, 59, 60, 44, 29]
	Medical object classification	Loveard et al.	[61, 46]
	Shape and coin recognition	Zhang et al.	[62]
Other Vision Problems	Edge detection	Lucier et al.	[65]
	San Mateo trail	Koza	[9]
	problem	Koza	[66]
	Image analysis	Howard et al.	[54]
		Poli	[67]
	Model Interpretation	Lindblad et al.	[14]
	Stereoscopic Vision	Graae et al.	[12]
	Image compression	Nordin et al.	[55]

Table 1: Object recognition and detection related work based on genetic programming.

3 The Approach

3.1 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 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. The process of the second phase and the GP testing procedure are shown in figure 2.

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

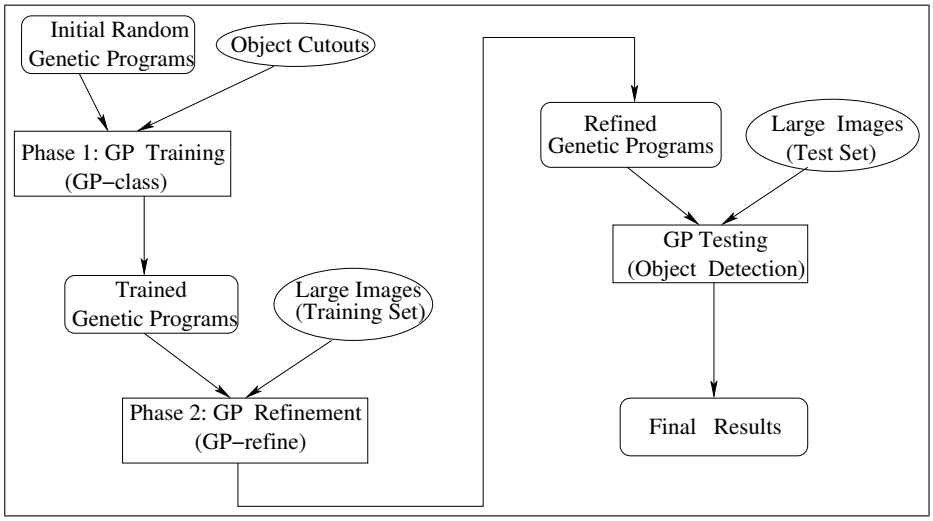


Figure 1: An overview of the two phase GP approach.

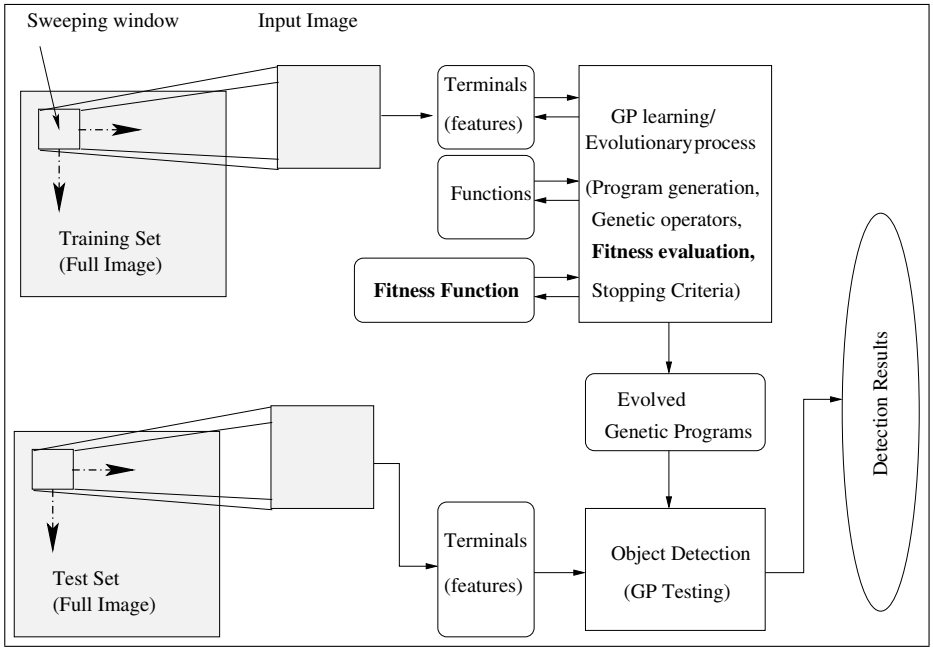


Figure 2: The second phase of GP training (GP-refine) and the GP testing procedure.

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.

3.2 Terminal set and function set

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 3. 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 4). 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.

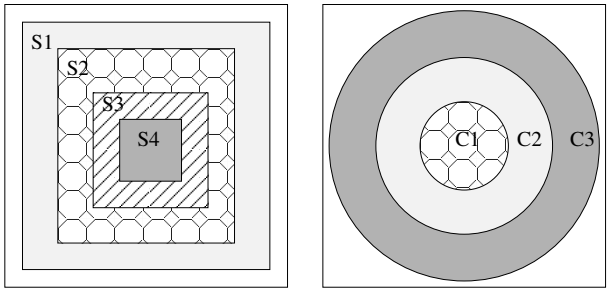


Figure 3: Local square and circular features as terminals.

Notice that these features are certainly not the best for these particular problems. However, our goal is to investigate the two-phase and the program size ideas rather than finding good features for a particular task, which is beyond the scope of this paper. Accordingly, instead of using some complex features such as the SIFT features [68, 7], haar wavelets and orientation histogram features [1], we used these simple features to keep the problem complexity low.

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.

3.3 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 three or more classes of objects of interest are involved, the standard genetic programming classification strategy mentioned above cannot be directly 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 [48]. 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.

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

3.4 Fitness functions

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 more complex fitness function to be described below to maximise object detection accuracy.

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 either a multi-objective fitness function or a single-objective fitness function that can integrate the effects of the multiple objectives.

While a real multi-objective fitness function can be used for object detection as in [69], the GP community typically takes the latter approach — usually uses a single-objective fitness function that can reflect the effects of the multiple objectives for a particular problem such as object detection [13, 38, 16, 70]. An example existing fitness function of this kind used in our previous work [16] (and similar ideas also used in other work [13, 38, 70]) is:

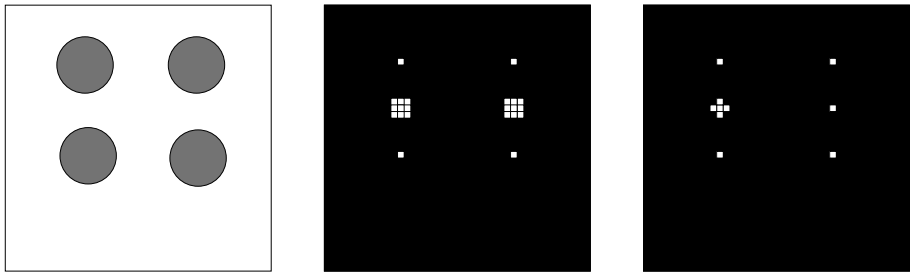
$$fitness(DR, FAR) = W_d * (1 - DR) + W_f * FAR \quad (2)$$

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) and 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). The parameters W_d, W_f reflect the relative importance between the detection rate and the false alarm rate.

Although such a fitness function accurately reflects the performance measure of an object detection system, it is not smooth. In particular, small improvements in an evolved genetic program may not be reflected in any change to the fitness function. The reason is the clustering process that is essential for the object detection — as the sliding window is moved over a true object, the program will generally identify an object at a cluster of window locations where the object is approximately centered in the window. It is important that the set of positions is clustered into the identification of a single object rather than the identification of a set of objects on top of each other.

Suppose we obtained two genetic programs from the population. Program 1 incorrectly identified a large cluster of locations as an object and Program 2 identified a smaller cluster of locations (as shown in figures 4 (b) and (c)). In terms of object detection, the program 2 was clearly better than program 1 since program 2 only produced six false alarm pixels but program 1 produced 18 false alarm pixels. However, the above fitness function grouped the two clusters of different numbers of false alarm pixels as the same number (which is two) of false positives for both programs. Thus the two programs have exactly the same FAR since both of them have two false positives. Accordingly, a fitness function based solely on DR and FAR cannot correctly rank these two programs, which means that the evolutionary process will have difficulty for selecting better programs. To deal with this problem, the False Alarm Area (FAA, the number of false alarm pixels which are not object centres but are incorrectly reported as object centres before clustering) was added to the fitness function.

Another problem of using this fitness function is that some genetic programs evolved are very long. When a short program and a long program produce the same detection rate and the same false alarm rate, the GP system will randomly choose one for reproduction, mutation or crossover during the evolutionary process. If the long



(a) (b) (c)

Figure 4: Sample object detection maps. (a) Original image; (b) Detection map produced by Program 1; (c) Detection map produced by program 2.

programs are selected, the evolution for the rest of the learning process will be slow. More importantly, the good building blocks in these long programs will have a much greater chance to be destroyed than in the short programs (Gedanken experiment in GP [8]), which could lead to poor solutions by the evolutionary process. This is mainly because this fitness function does not include any hints about the size of programs.

3.4.1 The new fitness function

To smooth the fitness function so that small improvement in genetic programs could be reflected and to consider the effect of program size, we added two measures, *false alarm area* and *program size* to the fitness function.

The new fitness of a genetic program is calculated as follows.

1. Apply the program as a moving $n \times n$ window template (n is the size of the input image window) to each of the training images and obtain the output value of the program at each possible window position, as shown in Figure 2. Label each window position with the 'detected' object according to the object classification strategy. Call this data structure a detection map.

2. Find the centres of *objects of interest only* by the following clustering process:

   - Scan the detection map from the up-left corner "pixel by pixel" for detected *objects of interest* (those "pixels" marked as the "background" class are skipped). When an object of a class of interest at a particular location is encountered, mark that location point as the centre of the object and skip pixels in $n/2 \times n/2$ square to right and below this location. In this way, all the locations ("pixels") considered "detected objects" by the genetic program within the square of then $n/2 \times n/2$ size will be "clustered" as a single object. The square size $n/2 \times n/2$ was chosen as half of the moving sweeping window size in order not to miss any detected object. This process will continue in the right cross and down directions until all the locations in the detection map are scanned or skipped. The locations marked by this process are considered the centres of the objects for the classes of interest detected by the genetic program.

3. Match these detected objects with the known locations of each of the desired/target objects and their classes. Here, we allow location error of $TOLERANCE$ pixels in the x and y directions. We have used a value of 2 for $TOLERANCE$. For example, if the coordinates of a known object centre are *(21, 19)* and the coordinates of a detected object centre are *(22, 21)*, we consider that the object has been correctly located.

4. Calculate the detection rate DR, the false alarm rate FAR, and the false alarm position FAA of the evolved program.

5. Count the size of the program by adding the number of terminals and the number of functions in the program.

6. Compute the fitness of the program according to equation 3.

$$fitness = K_1 \cdot (1 - DR) + K_2 \cdot FAR + K_3 \cdot FAP + K_4 \cdot ProgSize \quad (3)$$

where K_1, K_2, K_3, and K_4 are constant weighting parameters which reflect the relative importance between detection rate, false alarm rate, false alarm area, and program size.

We expect that the new fitness function can reflect both small and large improvement in genetic programs and can bias the search towards simpler functions. We also expected this would increase both the efficiency and the effectiveness of the evolutionary search. It will also have a tendency to reduce redundancy, making the programs more comprehensible.

Notice that adding the program size constrain to the fitness function is a kind of *parsimony pressure* technique [71, 72, 73]. Early work on this issue resulted in diverse opinions: some researchers think using parsimony pressure could improve performance [72], while some others thinks this could lead to premature convergence [73]. 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 2).

3.5 Parameters and termination criteria

In this system, we used tree structures and Lisp S-expressions to represent genetic programs [9]. The ramped half-and-half method [8, 9] was used for generating the programs in the initial population and for the mutation operator. The proportional selection mechanism and the reproduction [48], crossover and mutation operators [8] were used in the learning process.

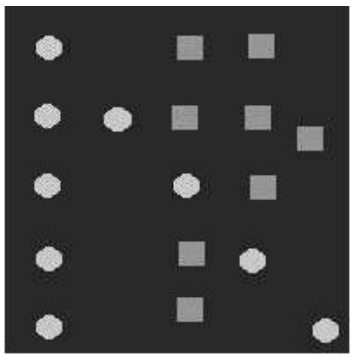
Important parameter values used in the experiments are shown in table 2. These parameter values were obtained using the existing heuristics in GP plus some minor effort on empirical search via experiments.

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

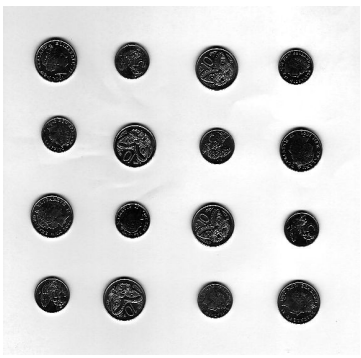
Table 2: Parameters used for GP training for the three databases.

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.

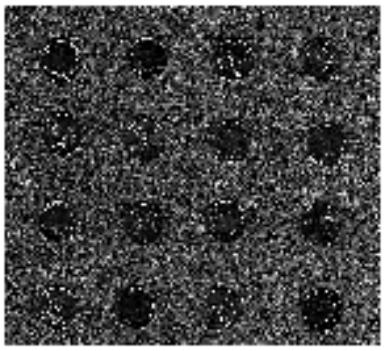
4 Image Data Sets



No. of images: 10
Object size:18×18
(Shape)



No. of images: 20
Object size:70× 70
(Coins)



No. of images: 20
Object size:60×60
(Heads/tails)

Figure 5: Object detection problems.

We used three data sets in the experiments. Example images are given in figure 5. 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. Since the sizes of the two classes (5-cent coins vs 10-cent coins) are quite different, it should not be very difficult to distinguish between the two classes. 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. The coins are placed in different locations with significantly different orientations. In addition, the background was generated using a Gaussian generator with the same mean (100) but a very large standard deviation (120), making the background more complex. Given the low resolution (75pt) of the images, this detection task is actually 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. To avoid the "lucky partitioning" of the these data set, the partitioning process of training and test sets was randomly repeated ten times for each of the three data sets and the average results are reported in the next section.

5 Results and discussion

5.1 Object detection results

The detection results of the two phase GP approach for the three image data sets are shown in table 3. These results are compared with the basic GP approach [15, 74] and a neural network approach [75, 76] using the same set of features. The basic GP approach is similar to the new GP approach described in this paper, except that it uses the old fitness function without considering the program size and false alarm areas (equation 2) and that genetic programs are learned from the full training images directly, which is a single stage approach[15, 74]. In the neural network approach[75, 76], a three layered feed forward neural network is trained by the back propagation algorithm [77] without momentum using an online learning scheme and fan-in factors [78]. For all

the three approaches, the experiments are repeated 50 times and the average results on the *test set* are presented in this section.

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

Table 3: Object detection results achieved by different approaches.

As can be seen from table 3, 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 [74], 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 on this data set using the same set of features. However, this might be partially because the features used here carried intrinsic bias towards the neural network approach and/or partially because the neural networks were not tuned, pruned or optimised [30, 69]. While further discussion here on this topic is beyond the goal of this paper, we are interested in carrying out further investigation in the future.

5.2 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 will investigate which of the factors is the most important for object detection in the future.

5.3 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:

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

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 3, 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.

6 Conclusions

Rather than investigating an application of GP for object detection, the goal of this paper is to investigate a study on improving GP techniques for object detection. The goal has been successfully achieved by developing a two phase GP approach and a new fitness function with constraints on program size. We investigated the effectiveness and efficiency of the two phase GP approach and the comprehensibility of genetic programs evolved using the new fitness function. 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. The new GP approach also achieved better detection accuracy than a neural network approach on the second coin data set using the same set of features. While a detailed comparison between the two approaches is beyond the goal of this paper, we are interested in doing further investigation in the future.

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.

This paper was focused on improving GP techniques rather than investigating applications of GP on object detection. However, it would be interesting to test the new GP approach developed in this paper on some more difficult, real world object detection tasks such as those in the Caltech 101 data set and the retina data set in the future.

Acknowledgement

This work was supported in part by the national Marsden Fund of Royal Society of New Zealand (05-VUW-017) and the University Research Fund (6/9, 7/39) at Victoria University of Wellington, New Zealand.

References

[1] F. Schwenker, Andreas Sachs, G. Palm, and H.A. Kestler. Orientation histograms for face recognition. In Schwenker F. and Marinai S., editors, *Artificial neural Networks in Pattern Recognition*, volume 4087 of *LNAI*, pages 253–259, Berlin Heidelberg New York, 2006. Springer Verlag.

[2] Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondence. In *Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Volume 1,*, pages 26–33. IEEE Computer Society Press, 2005.

[3] Alex. D. Holub, Max Welling, and Pietro Perona. Combining generative models and fisher kernels for object class recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV), Volume 1*, pages 136–143. IEEE Computer Society Press, 2005.

[4] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Volume 2*, pages 2169– 2178. IEEE Computer Society Press, June 2006.

[5] Hao Zhang, Alex Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Volume 2*, pages 2126– 2136. IEEE Computer Society Press, June 2006.

[6] Jim Mutch and David G. Lowe. Multiclass object recognition with sparse, localized features. In *Proceedings of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Volume 1*, pages 11–18. IEEE Computer Society Press, June 2006.

[7] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[8] 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*. San Francisco, Calif. : Morgan Kaufmann Publishers; Heidelburg : Dpunkt-verlag, 1998. Subject: Genetic programming (Computer science); ISBN: 1-55860-510-X.

[9] John R. Koza. *Genetic programming : on the programming of computers by means of natural selection*. Cambridge, Mass. : MIT Press, London, England, 1992.

[10] 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*, pages 303–309, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann.

[11] Karl Benson. Evolving finite state machines with embedded genetic programming for automatic target detection within SAR imagery. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 1543–1549, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 July 2000. IEEE Press.

[12] Cristopher T. M. Graae, Peter Nordin, and Mats Nordahl. Stereoscopic vision for a humanoid robot using genetic programming. In Stefano Cagnoni, Riccardo Poli, George D. Smith, David Corne, Martin Oates, Emma Hart, Pier Luca Lanzi, Egbert Jan Willem, Yun Li, Ben Paechter, and Terence C. Fogarty, editors, *Real-World Applications of Evolutionary Computing*, volume 1803 of *LNCS*, pages 12–21, Edinburgh, 17 April 2000. Springer-Verlag.

[13] Daniel Howard, Simon C. Roberts, and Conor Ryan. The boru data crawler for object detection tasks in machine vision. In Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and G"unther Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279 of *LNCS*, pages 220–230, Kinsale, Ireland, 3-4 April 2002. Springer-Verlag.

[14] F. Lindblad, P. Nordin, and K. Wolff. Evolving 3d model interpretation of images using graphics hardware. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation, CEC2002*, Honolulu, Hawaii, 2002.

[15] Mengjie Zhang and Victor Ciesielski. Genetic programming for multiple class object detection. In Norman Foo, editor, *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence (AI'99)*, pages 180–192, Sydney, Australia, December 1999. Springer-Verlag Berlin Heidelberg. Lecture Notes in Artificial Intelligence (LNAI Volume 1747).

[16] Mengjie Zhang, Peter Andreae, and Mark Pritchard. Pixel statistics and false alarm area in genetic programming for object detection. In Stefano Cagnoni, editor, *Applications of Evolutionary Computing, Lecture Notes in Computer Science, LNCS Vol. 2611*, pages 455–466. Springer-Verlag, 2003.

[17] Terry Caelli and Walter F. Bischof. *Machine Learning and Image Interpretation*. Plenum Press, New York and London, 1997. ISBN 0-306-45761-X.

[18] Earl Gose, Richard Johnsonbaugh, and Steve Jost. *Pattern Recognition and Image Analysis*. Prentice Hall PTR, Upper Saddle River, NJ 07458, 1996. ISBN 0-13-236415-8.

[19] Ayanna Howard, Curtis Padgett, and Carl Christian Liebe. A multi-stage neural network for automatic target detection. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 231–236, Anchorage, Alaska, 1998. 0-7803-4859-1/98.

[20] Yee Chin Wong and Malur K. Sundareshan. Data fusion and tracking of complex target maneuvers with a simplex-trained neural network-based architecture. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1024–1029, Anchorage, Alaska, May 1998. 0-7803-4859-1/98.

[21] D. Valentin, H. Abdi, and O'Toole. Categorization and identification of human face images by neural networks: A review of linear auto-associator and principal component approaches. *Journal of Biological Systems*, 2(3):413–429, 1994.

[22] 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*, volume 990 of *LNAI*, pages 165–176, Berlin, 3–6 October 1995. Springer Verlag.

[23] P. Winter, W. Yang, S. Sokhansanj, and H. Wood. Discrimination of hard-to-pop popcorn kernels by machine vision and neural network. In *ASAE/CSAE meeting*, Saskatoon, Canada, Sept. 1996. Paper No. MANSASK 96-107.

[24] David Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In Kenneth E. Kinnear, editor, *Advances in Genetic Programming*, pages 477–494. MIT Press, 1994.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[26] Brijesh Verma. A neural network based technique to locate and classify microcalcifications in digital mammograms. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1790–1793, Anchorage, Alaska, 1998. 0-7803-4859-1/98, IEEE.

[27] Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hibbard. Handwritten digit recognition: application of neural network chips and automatic learning. *IEEE Communications Magazine*, pages 41–46, November 1989.

[28] Dick de Ridder, Aarnoud Hoekstra, and Robert P. W. Duin. Feature extraction in shared weights neural networks. In *Proceedings of the Second Annual Conference of the Advanced School for Computing and imaging, ASCI*, pages 289–294, Delft, June 1996.

[29] Andy Song, Thomas Loveard, and Victor Ciesielski:. Towards genetic programming for texture classification. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, pages 461–472. Springer Verlag, 2001.

[30] Russell D. Reed and Robert J. Marks II. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Cambridge, MA: The MIT Press, 1999. ISBN 0-262-18190-8.

[31] M. R. Azimi-Sadjadi, D. Yao, Q. Huang, and G. J. Dobeck. Underwater target classification using wavelet packets and neural networks. *IEEE Transactions on Neural Networks*, 11(3):784–794, May 2000.

[32] C. Stahl, DaimlerChrysler Aerospace, and P. Schoppmann. Advanced automatic target recognition for police helicopter missions. In F. A. Sadjadi, editor, *Proceedings of SPIE Volume 4050, Automatic Target Recognition X*, April 2000. [4050-30].

[33] T. Wessels and C. W. Omlin. A hybrid system for signature verification. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Volume V*, Como, Italy, July 2000.

[34] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler. Using learning to facilitate the evolution of features for recognising visual concepts. *Evolutionary Computation*, 4(3):297–312, 1997.

[35] Jeng-Sheng Huang and Hsiao-Chung liu. Object recognition using genetic algorithms with a Hopfield's neural model. *Expert Systems with Applications*, 13(3):191–199, 1997.

[36] Stuart Russell and Peter Norvig. *Artificial Intelligence, A modern Approach*. Prentice Hall, 2nd edition, 2003.

[37] Margaret H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003.

[38] Daniel Howard, Simon C. Roberts, and Richard Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311, 1999.

[39] 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*, pages 330–335, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[40] Peter G. Korning. Training neural networks by means of genetic algorithms working on very long chromosomes. *International Journal of Neural Systems*, 6(3):299–316, September 1995.

[41] Victor Ciesielski and Jeff Riley. An evolutionary approach to training feed forward and recurrent neural networks. In L. C. Jain and R. K. Jain, editors, *Proceedings of the Second International Conference on Knowledge Based Intelligent Electronic Systems*, pages 596–602, Adelaide, April 1998.

[42] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, May 1997.

[43] Mukul V. Shirvaikar and Mohan M. Trivedi. A network filter to detect small targets in high clutter backgrounds. *IEEE Transactions on Neural Networks*, 6(1):252–257, Jan 1995.

[44] Andy Song, Vic Ciesielski, and Hugh 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 CEC2002*, pages 243–248. IEEE Press, 2002.

[45] Walter Alden Tackett. *Recombination, Selection, and the Genetic Construction of Computer Programs*. PhD thesis, Faculty of the Graduate School, University of Southern California, Canoga Park, California, USA, April 1994.

[46] Thomas Loveard and Victor Ciesielski. Representing classification problems in genetic programming. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1070–1077, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.

[47] Astro Teller and Manuela Veloso. PADO: Learning tree structured algorithms for orchestration into an object recognition system. Technical Report CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.

[48] Mengjie Zhang, Victor Ciesielski, and Peter Andreae. 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(8):841–859, 2003.

[49] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, Mass. : MIT Press, London, England, 1994.

[50] Wolfgang Kantschik, Peter Dittrich, Markus Brameier, and Wolfgang Banzhaf. Metaevolution in graph GP. In Riccardo Poli, Peter Nordin, William B. Langdon, and Terence C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'99*, volume 1598 of *LNCS*, pages 15–28, Goteborg, Sweden, 26-27 May 1999. Springer-Verlag.

[51] Wolfgang Kantschik and Wolfgang Banzhaf. Linear-graph GP—A new GP structure. In Evelyne Lutton, James A. Foster, Julian Miller, Conor Ryan, and Andrea G. B. Tettamanzi, editors, *Proceedings of the 4th European Conference on Genetic Programming, EuroGP 2002*, volume 2278, pages 83–92, Kinsale, Ireland, 3-5 2002. Springer-Verlag.

[52] P. A. Whigham. Grammatically-based genetic programming. In Justinian P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41, Tahoe City, California, USA, 9 July 1995.

[53] Gerald Robinson and Paul McIlroy. Exploring some commercial applications of genetic programming. In T. C. Fogarty, editor, *Evolutionary Computation, Volume 993, Lecture Note in Computer Science*. Springer-Verlag, 1995.

[54] Daniel Howard, Simon C. Roberts, and Conor Ryan. Evolution of an object detection ant for image analysis. In Erik D. Goodman, editor, *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 168–175, San Francisco, California, USA, 9-11 July 2001.

[55] Peter Nordin and Wolfgang Banzhaf. Programmatic compression of images and sound. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 345–350, Stanford University, CA, USA, 1996. MIT Press.

[56] Riccardo Poli. Genetic programming for feature detection and image segmentation. In T. C. Fogarty, editor, *Evolutionary Computing*, number 1143 in Lecture Notes in Computer Science, pages 110–125. Springer-Verlag, University of Sussex, UK, 1-2 April 1996.

[57] Stephen A. Stanhope and Jason M. Daida. Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery. In V. William Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming*, volume 1447 of *LNCS*, pages 735–744, Mission Valley Marriott, San Diego, California, USA, 25-27 March 1998. Springer-Verlag.

[58] Andy Song. *Texture Classification: A Genetic Programming Approach*. PhD thesis, Department of Computer Science, RMIT University, Melbourne, Australia, 2003.

[59] Andy Song and Vic Ciesielski. Texture analysis by genetic programming. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 2092–2099, Portland, Oregon, 20-23 June 2004. IEEE Press.

[60] 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*, pages 2126–2133, Canberra, 8-12 December 2003. IEEE Press.

[61] Thomas Loveard. *Genetic Programming for Classification Learning Problems*. PhD thesis, RMIT University, School of Computer Science and Information Technology, 2003.

[62] Mengjie Zhang and Will Smart. Multiclass object classification using genetic programming. In Guenther R. Raidl, Stefano Cagnoni, Jurgen Branke, David W. Corne, Rolf Drechsler, Yaochu Jin, Colin Johnson, Penousal Machado, Elena Marchiori, Franz Rothlauf, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*, volume 3005 of *LNCS*, pages 367–376, Coimbra, Portugal, 5-7 April 2004. Springer Verlag.

[63] Victor Ciesielski, Andrew Innes, Sabu John, and John Mamutil. Understanding evolved genetic programs for a real world object detection problem. In Maarten Keijzer, Andrea Tettamanzi, Pierre Collet, Jano I. van Hemert, and Marco Tomassini, editors, *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of *Lecture Notes in Computer Science*, pages 351–360, Lausanne, Switzerland, 30 March - 1 April 2005. Springer.

[64] Satoru Isaka. An empirical study of facial image feature extraction by genetic programming. In John R. Koza, editor, *the Genetic Programming 1997 Conference*, pages 93–99. Stanford Bookstore, Stanford University, CA, USA, July 1997. Late Breaking Papers.

[65] Bradley J. Lucier, Sudhakar Mamillapalli, and Jens Palsberg. Program optimisation for faster genetic programming. In *Genetic Programming – GP'98*, pages 202–207, Madison, Wisconsin, July 1998.

[66] John R. Koza. Simultaneous discovery of reusable detectors and subroutines using genetic programming. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 295–302, Morgan Kauffman, 1993.

[67] Riccardo Poli. Genetic programming for image analysis. In John R. Koza, David E. Goldberg, and David B. Fogel a nd Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First A nnual Conference*, pages 363–368, Stanford University, CA, USA, 28–31 1996. MIT Press.

[68] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[69] Alexander Gepperth and Stefan Roth. Applications of multi-objective structure optimization. *Neurocomputing*, 69(7-9):701–713, 2006.

[70] Mark E. Roberts and Ela Claridge. Cooperative coevolution of image feature construction and object detection. In Xin Yao, Edmund Burke, Jose A. Lozano, Jim Smith, Juan J. Merelo-Guervós, John A. Bullinaria, Jonathan Rowe, Peter Tiňo Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *LNCS*, pages 902–911, Birmingham, UK, 18-22 September 2004. Springer-Verlag.

[71] P. W. H. Smith. Controlling code growth in genetic programming. In Robert John and Ralph Birkenhead, editors, *Advances in Soft Computing*, pages 166–171, De Montfort University, Leicester, UK, 2000. Physica-Verlag.

[72] Richard Dallaway. Genetic programming and cognitive models. Technical Report CSRP 300, School of Cognitive & Computing Sciences, University of Sussex,, Brighton, UK, 1993. In: Brook & Arvanitis, eds., 1993 The Sixth White House Papers: Graduate Research in the Cognitive & Computing Sciences at Sussex.

[73] 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.

[74] Urvesh Bhowan. 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., October 2003.

[75] Mengjie Zhang and Victor Ciesielski. Using back propagation algorithm and genetic algorithm to train and refine neural networks for object detection. In Trevor Bench-Capon, Giovanni Soda, and A Min Tjoa, editors, *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA'99)*, pages 626–635, Florence, Italy, August 1999. Springer-Verlag. Lecture Notes in Computer Science, (LNCS Volume 1677).

[76] Bunna Ny. Multi-cclass object classification and detection using neural networks. BSc Honours research project/thesis, school of mathematical and computing sciences, victoria university of wellington, October 2003.

[77] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP research group, editors, *Parallel distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations*, chapter 8. The MIT Press, Cambridge, Massachusetts, London, England, 1986.

[78] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard W. Hubbard, and L. D. Jackel. Handwritten zip code recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan Kaufmann, San Mateo, CA, 1990.