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

School of Mathematical and Computing Sciences
Computer Science

A Domain Independent Approach to Multiclass Object Detection Using Genetic Programming

Mengjie Zhang, Victor Ciesielski and Peter Andreae

Technical Report CS-TR-02/04
May 2002

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 Domain Independent Approach to Multiclass Object Detection Using Genetic Programming

Mengjie Zhang, Victor Ciesielski and Peter Andreae

Technical Report CS-TR-02/04
May 2002

Abstract

This paper describes a domain independent approach to the use of genetic programming for object detection problems in which the locations of small objects of multiple classes in large images must be found. The evolved program is applied as a template, in moving window fashion, over the large images to locate the objects of interest. The paper develops three terminal sets based on domain independent pixel statistics computed from a square input field and considers two different function sets based on standard arithmetic and transcendental operations. The fitness function is designed based on the detection rate and the false alarm rate. A program classification map and a multiple binary map are developed for multiclass object classification problems to determine the class of an object in an input field. We have tested the method on three object detection problems of increasing difficulty with multiple different classes of interest. On images of easy and medium difficulty all objects are detected with no false alarms. On difficult images there are still significant numbers of errors, however the results are considerably better than those of a neural network based method for the same problems.

Keywords Machine learning, neural networks, genetic algorithms, object recognition, target detection, computer vision.

Author Information

The first author is a lecturer in computer science, Victoria University of Wellington, New Zealand. The second author is an associate professor of artificial intelligence, RMIT University, Melbourne, Australia. The third author is a senior lecturer in computer science, Victoria University of Wellington, New Zealand.

A Domain Independent Approach to Multiclass Object Detection Using Genetic Programming

Mengjie Zhang Victor Ciesielski Peter Andreae

Abstract

This paper describes a domain independent approach to the use of genetic programming for object detection problems in which the locations of small objects of multiple classes in large images must be found. The evolved program is applied as a template, in moving window fashion, over the large images to locate the objects of interest. The paper develops three terminal sets based on domain independent pixel statistics computed from a square input field and considers two different function sets based on standard arithmetic and transcendental operations. The fitness function is designed based on the detection rate and the false alarm rate. A program classification map and a multiple binary map are developed for multiclass object classification problems to determine the class of an object in an input field. We have tested the method on three object detection problems of increasing difficulty with multiple different classes of interest. On images of easy and medium difficulty all objects are detected with no false alarms. On difficult images there are still significant numbers of errors, however the results are considerably better than those of a neural network based method for the same problems.

This work not only extends genetic programming to multiclass object detection problems, but also shows how to use a single learned/evolved genetic program for both object classification and localisation. The results also show that mutation plays an important role in the genetic programming evolutionary process for multiple class object detection tasks. The object classification map developed in this approach can be used as a general classification strategy in genetic programming for multiple class classification problems.

Keywords Machine learning, neural networks, genetic algorithms, object recognition, target detection, computer vision.

---

[0]Dr. Mengjie Zhang is a lecturer in computer science at Victoria University of Wellington, New Zealand. Dr. Victor Ciesielski is an associate professor of artificial intelligence, RMIT University, Australia. Dr. Peter Andreae is a senior lecturer in computer science, Victoria University of Wellington, New Zealand. Dr. Mengjie Zhang is the corresponding author. His email address is: `mengjie@mcs.vuw.ac.nz`

1 Introduction

As more and more images are captured in electronic form, the need for programs which can find objects of interest in a database of images is increasing. For example, it may be necessary to find all tumors in a database of x-ray images, all cyclones in a database of satellite images or a particular face in a database of photographs. The common characteristic of such problems can be phrased as "Given $subimage_1, subimage_2$, ..., $subimage_n$ which are examples of the object of interest, find all images which contain this object and its location(s)". Figure 11 (page 23) shows examples of problems of this kind. In the problem illustrated by figure 11 (b), we want to find centers of all of the 5 cent and 20 cent coins and determine whether the head or the tail side is up. Examples of other problems of this kind include target detection problems [9, 36, 40] where the task is to find, say, all tanks, trucks or helicopters in an image. Unlike most of the current work in the object recognition area, where the task is to detect only objects of one class [9, 27, 28], our objective is to detect objects from a number of classes.

Several approaches have been applied to automatic object detection and recognition problems. Typically, they use multiple independent stages, such as preprocessing, edge detection, segmentation, feature extraction and object classification[6, 26], which often results in some efficiency and effectiveness problems. The final results rely too much upon the results of each stage. If some objects are lost in one of the early stages, it is very difficult or impossible to recover them in the later stage. To avoid these disadvantages, this paper introduces a single stage approach.

There have been a number of reports on the use of genetic programming in object detection and classification [29, 32]. [38] describes a genetic programming system for object detection in which the evolved functions operate directly on the pixel values. [34] describes a genetic programming system and a face recognition application in which the evolved programs have a local indexed memory. All of these approaches are based on detecting one class of objects or two class classification problems, that is, objects vs everything else. The approach in this paper will focus on object detection problems in which a number of objects in more than two classes of interest need to be localised and classified.

1.1 Outline of the Approach to Object Detection

A brief outline of the method is as follows:

1. Assemble a database of images in which the locations and classes of all of the objects of interest are manually determined. Split these images into a training set and a test set.

2. Determine an appropriate size ($n \times n$) of a square which will cover all single objects of interest to form the input field.

3. Invoke an evolutionary process with images in the training set to generate a program which can determine the class of an object in its input field.

4. Apply the generated program as a moving window template to the images in the test set and obtain the locations of all the objects of interest in each class. Calculate the detection rate and the false alarm rate on the test set as the measure of performance.

1.2 Goals

The overall goal of this paper is to investigate a learning/adaptive, single stage, domain independent approach to multiple class object detection problems without any preprocessing, segmentation and specific feature extraction. This approach is based on a genetic programming technique. Rather than using specific image features, pixel statistics are used as inputs to the evolved programs. Specifically, the following questions will be explored on a sequence of detection problems of increasing difficulty to determine the strengths and limitations of the method.

- What image features (pixel statistics) would make useful terminals?
- Would the four standard arithmetic operators be sufficient for the function set?
- How can the fitness function be constructed given that there are multiple classes of interest?
- How will performance vary with increasing difficulty of image detection problems?
- Will the performance be better than a neural network approach [41] on the same problems?

1.3 Structure

The remainder of this paper gives a brief literature survey, then describes the main components of this approach including the terminal set, the function set and the fitness function. After describing the three image databases used here, we present the experimental results and compare them with a neural network method. Finally, we analyse the results and the evolved programs and present our conclusions.

2 Literature Review

2.1 Object Detection

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 term *automatic target recognition* or *automatic object recognition* and some tasks of image analysis.

We classify the existing object detection systems into three dimensions based on whether the approach is segmentation free or not, domain independent or specific, and the number of object classes of interest in an image.

2.1.1 Segmentation Based vs Single Stage

According to the number of independent stages used in the detection procedure, we divide the detection methods into two categories:

- **Segmentation based approach**, which uses multiple independent stages for object detection. Most research on object detection involves four stages: *preprocessing, segmentation, feature extraction* and *classification* [5, 7, 10], as shown in figure 1. 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 algorithms or methods for these stages are generally domain specific. Learning paradigms such as neural networks and genetic algorithms/programming have usually been applied to the classification stage. In general, each independent stage needs a program to fulfill that specific task and accordingly multiple programs are needed for object detection problems. Success at each stage is critical to achieving good final detection performance. Detection of trucks and tanks in visible, multi-spectral infrared and synthetic aperture radar images [36] and recognition of tanks in cluttered images [6] are two examples.

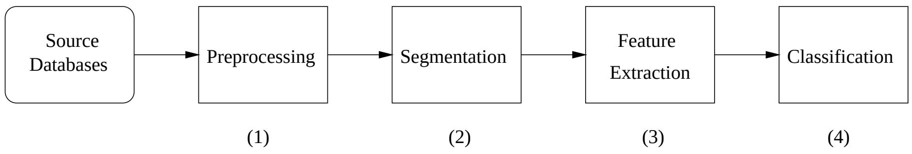


Figure 1: A typical procedure for object detection.

- **Single stage approach**, which uses only a single stage to detect the objects of interest in large images. There is only a single program produced for the whole object detection procedure in this approach. The major property of this approach is segmentation free. Detecting tanks in infrared images [40] and detecting small targets in cluttered images [30] based on a single neural network are examples of these kinds.

While most recent work on object detection problems concentrates on the segmentation based approach, this paper focuses on the single stage approach.

2.1.2 Domain Specific approach vs Domain Independent Approach

In terms of the generalisation of the detection systems, there are two major approaches:

- **Domain specific object detection**, which uses specific image features as inputs to the detector or classifier. These features, which are usually highly domain dependent, are extracted from entire images or segmented images. In a lentil grading and quality assessment system [39], for example, features such as brightness, colour, size and perimeter are extracted and used as inputs to a neural network classifier. This approach generally involves a time consuming investigation of good features for a specific problem and a hand-crafting of the corresponding feature extraction programs.

- **Domain independent object detection**, which usually use the raw pixels directly (no features) as inputs to the detector or classifier. In this case, feature selection, extraction and the hand-crafting of corresponding programs can be completely removed. This approach usually needs learning and adaptive techniques to learn features for the detection task. Directly using raw image pixel data as input to neural networks for detecting vehicles (tanks, trucks, cars, etc.) in infrared images [9] is such an example. However, a long learning/evolution time is usually required due to the large number of pixels. Furthermore, it generally requires a large number of training examples [3].

A special case is to use a small number of domain independent, pixel level features (referred to as *pixel statistics*), such as *mean* and *variance* of some portions of an image [13]. While this still involves feature extraction in some cases, such features could be chosen from a preexisting feature library. Moreover, the programs for this kind of feature extraction are usually standard and can be simply selected from some existing program packages.

2.1.3 Multiple Class vs Single Class

Regarding the number of object classes of interest in an image, there are two main types of detection problems:

- **One class object detection problem**, where there are multiple objects in each image, however they belong to a single class. One special case in this category is that there is only one object of interest in each source image. In nature, these problems contain a binary classification problem: *object* vs *non-object*, also called *object* vs *background*. Examples are detecting small targets in thermal infrared images [30] and detecting a particular face in photograph images [21].
- **Multiple class object detection problem**, where there are multiple object classes of interest each of which has multiple objects in each image. Detection of handwritten digits in zip code images [20] is an example of this kind.

In general, multiple class object detection problems are more difficult than one class detection problems. This paper is focused on detecting multiple objects from a number of classes in a set of images, which is particularly difficult. Most research in object detection which has been done so far belongs to the one class object detection problem.

2.2 Performance Evaluation

In this paper, we use the *detection rate* and *false alarm rate* to measure the performance of multiclass object detection problems. 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 or *false alarms/object* [30], 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.

The main goal of object detection is to obtain a high detection rate and a low false alarm rate. There is, however, a tradeoff between them for a detection system. Trying to improve the detection rate often results in a increase in false alarm rate, and vice versa. Detecting objects in images with very cluttered backgrounds is an extremely difficult problem where false alarm rates of 200-2000% (that is the detection system suggests that there are 20 times as many objects as there really are) are common [28, 30].

Most research which has been done in this area so far only presents the results of the classification stage (only the final stage in figure 1, on page 5) and assumes that all other stages have been properly done. However, the results presented in this paper are the performance for the whole detection problem (including the localisation and classification). Thus, it is inappropriate to simply compare the results in this paper with those obtained by other systems which are developed only for the classification problem. It is also inappropriate to directly compare the performance on multiple class detection problems achieved in this paper with the performance on one class detection problems reported in other papers.

2.3 Genetic Programming

Genetic programming (GP) is an approach to automatic programming, in which a computer can construct and refine its own programs to solve specific tasks. First introduced by Koza [17] in 1992, GP has become another genetic learning/search paradigm based on the use of Darwinian evolution 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 genetic programming 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.

Constructing a GP system involves making design decisions for a number of elements of the GP system, including the representation of programs, the construction of an initial population of programs, the evaluation of programs and the construction of new population of programs.

2.3.1 Program Representation

Much of the GP work was done using LISP or LISP-like representations of the programs. A sample computer program for the algebraic equation $(x-1)-x^3$ can be represented in LISP as the S-expression `(- (- x 1) (* x (* x x)))`. The tree representation is shown in figure 2.

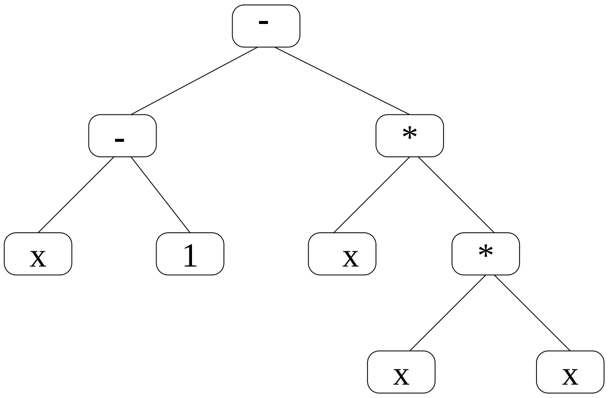


Figure 2: A simple tree representation for a sample LISP program.

The programs are constructed from *a terminal set* and *a function set* which vary according to the problem domain. Terminals and functions are also called *primitives*, and the terminal set and the function set are combined to form a *primitive set* [12].

Function Set: Functions in a function set form the internal nodes of the tree representation of a program. In general, there are two kinds of functions used in genetic programming. The first class refers to standard functions, such as the four arithmetic operations. The second class comprises specific functions which vary with the problem domain.

Terminal Set: Terminals have no arguments and form the leaves of the parse tree. Typically, terminals represent the inputs to the GP program, the constants supplied to the GP program, or zero-argument functions with side-effects executed by the GP program [2]. In any case, a terminal returns an actual numeric value without having to take an input.

2.3.2 Program Generation

There are several ways of generating programs to initialise a GP population, including *full, grow* and *ramped half-and-half* [17]. In the full method, functions are selected as the (internal) nodes of the program until a given depth of the program tree is reached. Then terminals are selected to form the leaf nodes. This ensures that full, entirely balanced trees are constructed. When the grow method is used, nodes are selected from either functions or terminals. If a terminal is selected, the generation process is terminated for the branch and moves on to the next non-terminal branch in the tree. In the ramped half-and-half method, both the full and grow methods are combined. Half of the programs generated for each depth value are created by using the grow method and the other half using the full method.

2.3.3 Fitness Function

Fitness is the measure of how well a program has learnt to predict the output from the input during simulated evolution. The fitness of a program generated by the evolutionary process is computed according to the fitness function. The fitness function should be designed to give graded and continuous feedback about how well a program in a population performs on the training set.

2.3.4 Selection Mechanism

The selection mechanism determines which evolved program will be used for the genetic operators to produce new individuals for the next generation during the evolutionary process. Two of the most commonly used selection methods are *proportional selection* and *tournament selection*.

In the proportional selection method [17], an individual in a population will be selected according to the proportion of its own fitness to the total sum of the fitness of all the individuals in the population. Programs with low fitness scores would have a low probability of having any genetic operators applied to them and so would most likely be removed from the population. Programs which perform particularly well in an environment will have a very high probability of being selected.

The tournament selection method [2] is based on competition within only a subset of the population, rather than the whole population. A number of programs are selected randomly according to the tournament size and a selective competition takes place. The better individuals in the tournament are allowed to replace the worse individuals. In the smallest possible tournament, two individuals can compete. The winner is allowed to reproduce with mutation and the result is returned to the population, replacing the loser of the tournament.

2.3.5 Genetic Operators

There are three fundamental genetic operators: reproduction, mutation and crossover.

Reproduction is the basic engine of Darwinian theory [17], which involves just simply copying the selected program from the current generation to the new generation. This allows good programs to survive during evolution.

Mutation operates only on a single selected program and introduces new genetic code in the new generation. This operator removes a random subtree of a selected program, then puts a new subtree in the same place. The goal here is to keep the diversity of the population in evolution.

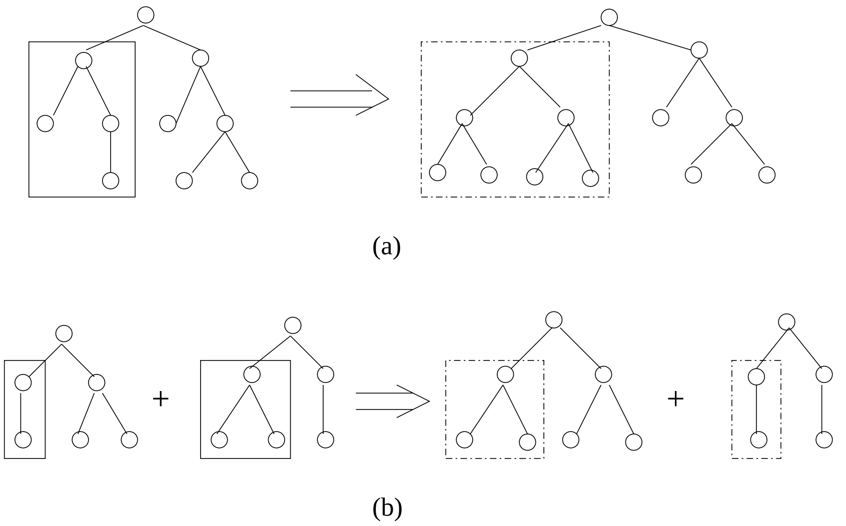


Figure 3: Effect of genetic operators in genetic programming. (a) Mutation in GP: Replaces a random subtree; (b) Crossover in GP: Swaps two random subtrees.

Crossover takes advantage of different selected programs within a population, attempting to integrate the useful attributes from them. The crossover operator combines the genetic material of the two selected parents by swapping a subtree of one parent with a subtree of the other, and introducing two newly formed programs into the population in the next generation.

2.4 Related Work – GP for 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 vision problems. This in part reflects the fact that genetic programming is a relatively young discipline compared with, say, neural networks.

2.4.1 Object Classification

Tackett [32, 33] uses genetic programming to assign detected image features to a *target* or *non-target* category. Seven primitive image features and twenty statistical features are extracted and used as the terminal set. The four standard arithmetic operators and a logic function are used as the function set. The fitness function is based on the classification result. The approach was tested on US Army NVEOD Terrain Board imagery, where vehicles such as tanks need to be classified. The genetic programming method outperformed both a neural network classifier and a binary tree classifier on the same data, producing lower rates of false positives for the same detection rates.

Andre [1] uses genetic programming to evolve functions that would traverse an image, calling upon co-evolved detectors in the form of hit-miss matrices to guide the search. These hit-miss matrices are evolved with a two dimensional genetic algorithm. These evolved functions can be used to discriminate between two letters or to recognise single digits.

Koza [19] (in chapter 15) uses a "turtle" to walk over a bitmap landscape. This bitmap is to be classified either as a letter "L", a letter "I", or neither of them. The turtle has access to the values of the pixels in the bitmap by moving over them and calling a detector primitive. The turtle uses a decision tree process, in conjunction with negative primitives, to walk over the bitmap and decide which category a particular landscape falls into. Using automatically defined functions as local detectors and a constrained syntactic structure, some perfect scoring classification programs were found. Further experiments showed that detectors can be made for different sizes and positions of letters, although each detector has to be specialised to a given combination of these factors.

Teller and Veloso [34] use a genetic programming method based on the PADO language to perform face recognition tasks on a database of face images in which the evolved programs have a local indexed memory. The approach was tested on a discrimination task between 5 classes of images [35] and achieved up to 60% correct classification for images without noise.

Robinson and McIlroy [25] apply genetic programming techniques to the problem of eye location in grey-level face images. The input data from the images is restricted to a 3000 pixel block around the location of the eyes in the face image. This approach produced promising results over a very small training set, up to 100% true positive detection with no false positives, on a three image training set. Over larger sets the genetic programming approach performed less well however, and could not match the performance of neural network techniques.

Winkeler and Manjunath [38] produce genetic programs to locate faces in images. Face samples are cut out and scaled, then pre-processed for feature extraction. The statistics gleaned from these segments are used as terminals in genetic programming which evolves an expression returning how likely a pixel is to be part of a face image. Separate experiments process the grey scale image directly, using low level image processing primitives and scale-space filters.

2.4.2 Object Detection

All the reported genetic programming based object detection approaches belong to the *one class object detection* category. In these detection problems, there is only one object class of interest in the large images.

Howard, Roberts and Brankin [13] present a genetic programming approach to automatic detection of ships in low-resolution synthetic aperture radar imagery. A number of random integer/real constants and pixel statistics are used as terminals. The four arithmetic operators and *min* and *max* operators constitute the function set. The fitness is based on the number of the true positives and false positive objects detected by the evolved program. A two stage evolution strategy was applied in this approach. In the first stage, genetic programming evolved a detector that could correctly distinguish the target (ship) pixels from the non-target (ocean) pixels. The best detector was then applied to the entire image and produced a number of false alarms. In the second stage, a brand new run of genetic programming was tasked to discriminate between the clear targets and the false alarms as identified in the first stage and another detector was generated. This two stage process resulted in two detectors that were then fused using the *min* function. These two detectors return a real number, which if greater than zero denotes a ship pixel, and if zero or less denotes an ocean pixel. The approach was tested on images chosen from commercial SAR imagery, a set of 50m and 100m resolution images of the English Channel taken by the European Remote Sensing satellite. One of the 100m resolution images was used for training, two for validation and two for testing. The training was quite successful with perfect detection rate and no false alarms, while there was only one false positive in each of the two test images and the two validation images which contained 22, 22, 48 and 41 true objects.

Isaka [16] uses genetic programming to locate mouth corners in small (*50×40*) images taken from images of faces. Processing each pixel independently using an approach based on relative intensities of surrounding pixels, the genetic programming approach was shown to perform comparably to a template matching approach on the same data.

A list of object detection related work based on genetic programming is shown in table 1.

3 GP Adapted to Multiclass Object Detection

3.1 The Genetic Programming System

In this section, we describe our approach to a genetic programming system for multiple class object detection problems. Figure 4 shows an overview of this approach, which has a learning process and a

Problems	Applications	Authors	Year	Source
Object Classification	Tank detection (classification)	Tackett	1993	[32]
		Tackett	1994	[33]
	Letter recognition	Andre	1994	[1]
		Koza	1994	[19]
	Face recognition	Teller et al.	1995	[34]
	Small target classification	Stanhope et al.	1998	[31]
		Winkeler et al.	1997	[38]
	Shape recognition	Teller et al.	1995	[35]
	Eye recognition	Robinson et al.	1995	[25]
Object Detection	Ship detection	Howard et al.	1999	[13]
	Mouth detection	Isaka	1997	[16]
	Small target detection	Benson	2000	[4]
	Vehicle detection	Howard et al.	2002	[15]
Other Vision Problems	Edge detection	Lucier et al.	1998	[22]
	San Mateo trail problem	Koza	1992	[17]
		Koza	1993	[18]
	Image analysis	Howard et al.	2001	[14]
	Image compression	Nordin et al.	1996	[23]

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

testing procedure. In the learning/evolutionary process the evolved genetic programs use a square input field which is large enough to contain each of the objects of interest and are applied, in a moving window fashion, to the entire images in the training set to detect the objects of interest. In the test procedure, the best evolved genetic program obtained in the learning process is then applied to the entire images in the test set to measure object detection performance.

The learning/evolutionary process in our GP approach is summarised as follows:

1. Initialise the population.

2. Repeat until a termination criterion satisfied:

   2.1 Evaluate the individual programs in the current population. Assign a fitness to each program.

   2.2 Until the new population is fully created, repeat the following:

   - Select programs in the current generation.
   - Perform genetic operators on the selected programs.
   - Insert the result of the genetic operations into the new generation.

3. Present the best individual in the population as the output — the learned/evolved genetic program.

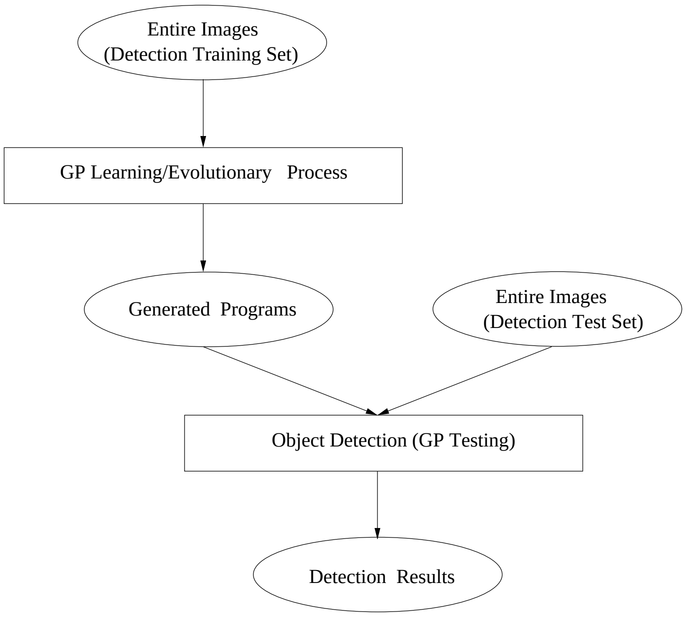


Figure 4: An overview of the GP approach for multiple class object detection.

In this system, we used the tree-like program structure (section 2.3.1) to represent genetic programs. The ramped half-and-half method was used for generating the programs in the initial population and for the mutation operator (section 2.3.2). The proportional selection mechanism (section 2.3.4) and the reproduction, crossover and mutation operators (section 2.3.5) were used in the learning process.

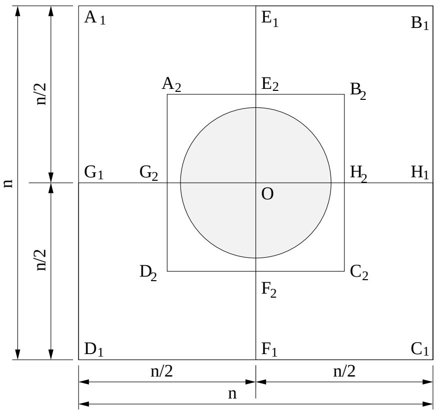
In the remainder of this section, we address the other aspects of the learning/evolutionary system: (1) Determination of the terminal set; (2) Determination of the function set; (3) Development of a classification strategy; (4) Construction of the fitness measure; (5) Selection of the input parameters and determination of the termination strategy.

3.2 The Terminal Sets

For object detection problems, terminals generally correspond to image features. In our approach, we designed three different terminal sets: local rectilinear features, circular features and "pixel features". In all these cases, the features are statistical properties of regions of the image, and we refer to them as pixel statistics.

3.2.1 Terminal Set I — Rectilinear Features

In the first terminal set, twenty pixel statistics, *F1 to F20* in table 2, are extracted from the input field as shown in figure 5. The input field must be sufficiently large to contain the biggest object and some



Squares:

$A_1B_1C_1D_1$, $A_2B_2C_2D_2$,

$A_1E_1OG_1$, $E_1B_1H_1O$,

$G_1OF_1D_1$, $OH_1C_1F_1$

Rows and Columns (Lines):

G_1H_1, E_1F_1, G_2H_2, E_2F_2

Size of the Lines:

$G_2H_2 = A_2B_2 = E_2F_2 = B_2C_2$:

User defined; Default $= n/2$

Figure 5: The input field and the image regions and lines for feature selection in constructing terminals.

background, yet small enough to include only a single object. In this way, the evolved program, as a detector, could automate the "human eye system" of identifying pixels/object centres which stand out from their local surroundings.

In figure 5, the grey filled circle denotes an object of interest, the square $A_1B_1C_1D_1$ represents the input field. The five smaller squares represent local regions from which pixel statistics will be computed. The four central lines (rows and columns) are also used for a similar purpose.[1] The mean and standard deviation of the pixels comprising each of these regions are used as two separate features. There are 6 regions giving 12 features, F_1 to F_{12}. Also we use pixels along the main axes (four lines) of the input field, giving features F_{13} to F_{20}.

In addition to these pixel statistics, we use a terminal which generates a random number in the range [0,255]. This corresponds to the range of pixel intensities in grey level images.

These pixel statistics have the following characteristics:

- They are symmetrical.
- Local regional features (from small squares and lines) are included. This assists the finding of object centres in the sweeping procedure — if the evolved program is considered as a moving window template, the match between the template and the sub-image forming the input field will be better when the moving template is close to the centre of an object.
- They are domain independent and easy to extract. These features belong to the pixel level and can be part of a domain independent pre-existing feature library of terminals from which the genetic

[1] These lines can be considered special local regions. If the input field size n is an even number, each of these "lines" is a rectangle consisting of two rows or two columns of pixels.

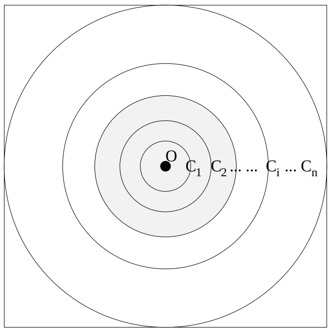
Pixel Statistics		Regions and Lines of Interest
Mean	SD	
F_1	F_2	big square $A_1B_1C_1D_1$
F_3	F_4	small central square $A_2B_2C_2D_2$
F_5	F_6	upper left square $A_1E_1OG_1$
F_7	F_8	upper right square $E_1B_1H_1O$
F_9	F_{10}	lower left square $G_1OF_1D_1$
F_{11}	F_{12}	lower right square $OH_1C_1F_1$
F_{13}	F_{14}	central row of the big square G_1H_1
F_{15}	F_{16}	central column of the big square E_1F_1
F_{17}	F_{18}	central row of the small square G_2H_2
F_{19}	F_{20}	central column of the small square E_2F_2

Table 2: Twenty pixel statistics. (SD: standard deviation)

programming evolutionary process is expected to automatically learn and select only those relevant to a particular domain. This is quite different from the traditional image processing and computer vision approaches where the problem specific features are often needed.

- The number of these features is fixed. In this approach, the number of features is always twenty no matter what size the input field is. This is particularly useful for the generalisation of the system implementation.

3.2.2 Terminal Set II — Circular Features



Features		Local Boundaries
Mean	Stan. Dev.	
F_1	F_2	central pixel
F_3	F_4	circular boundary C_1
F_5	F_6	circular boundary C_2
...	...	... ...
$F_{(2i+1)}$	$F_{(2i+2)}$	circular boundary C_i
...	...	... ...
$F_{(2n+1)}$	$F_{(2n+2)}$	circular boundary C_n

Figure 6: The input field and the image boundaries for feature extraction in constructing terminals.

The second terminal set is based on a number of circular features, as shown in figure 6. The features were computed based on a series of concentric circles centred in the input field. This terminal set focused on boundaries rather than regions. The gap between the radii of two neighbouring circles is one pixel. For instance, if the input field is 19×19 pixels, then the number of central circles will be $19/2+1=10$ (the central pixel is considered as a circle with a zero radius); accordingly, there would be 20 features. Compared with the rectilinear terminal set, the number of these circular features in this terminal set depends on the size of the input field.

3.2.3 Terminal Set III — "Pixels"

The goal of this terminal set is to investigate the use of raw pixels as terminals in genetic programming. To decrease the computation cost, we considered a *2×2* square, or 4 pixels, as a single "pixel". The average value of the four pixels in the square was used as the value of this "pixel", as shown in figure 7.

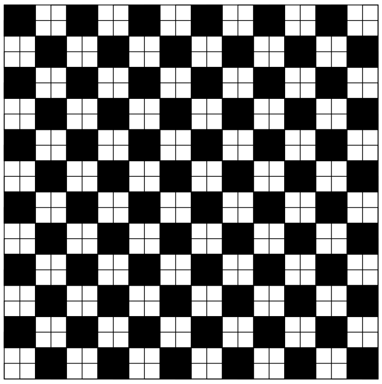


Figure 7: "Pixel" terminals.

3.3 The Function Sets

We used two different function sets in the experiments: four arithmetic operations only, and combination of arithmetic and transcendental functions.

3.3.1 Function Set I

In the first function set, the four standard arithmetic operations are used to form the non-terminal nodes:

$$FuncSet1 = \{+, -, *, /\}$$

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. This function set was designed to investigate whether the four standard arithmetic functions are sufficient for the multiple class object detection problems.

A generated program consisting of the four functions and a number of rectilinear terminals is shown in figure 8.[2] This program performed particularly well for the coin images.

$$\frac{F16}{F14} + F5 + \frac{\frac{F11}{F14 \cdot F20}}{F11} + F12 - F14 - (F9 \cdot F11 \cdot F1 \cdot F10 - F9 \cdot F17) \cdot \frac{F5}{F18} - [F17 + (F11 + F12) \cdot F20 + (F2 + 145.765 - \frac{F6}{F11}) \cdot (133.082 - F17) \cdot \frac{F11}{F14 \cdot F20}] + [(F6 - F5 - F3 \cdot F6) \cdot \frac{F1 + 145.765 + F16 \cdot F10}{F18} - F12] \cdot [F17 + (F17 + F12) \cdot F20 + F14 \cdot F12 \cdot (F1 + F12 - F17)]$$

Figure 8: A generated program for the coin detection problem

3.3.2 Function Set II

We also designed a second function set. We hypothesized that convergence might be quicker if the function values were close to the range $(-1, 1)$ and more functions might lead to better results if the four arithmetic functions were not sufficient. We introduced some transcendental functions, that is, the absolute function *dabs*, the trigonometric sine function *sin*, the logarithmetic function *log*, and the exponent (to base e) function *exp*, to form the second function set:

$$FuncSet2 = \{+, -, *, /, dabs, sin, log, exp\}$$

3.4 Object Classification Strategy

The output of a genetic program in a standard GP system is a floating point number. Generally genetic programs can perform one class object detection tasks quite well where the division between negative and non-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). This is similar to the binary classification problems in the standard genetic programming where the division between negative and non-negative numbers acts as a natural boundary for a distinction between the two classes. Thus,

[2] The *LISP* form of this program is:

```
(+ (- (+ (+ (/ F16 F14) F5) (+ (/ (/ F11 (* F14 F20)) F11) (- F12 F14))) (- (* (-
(* (* (* F9 F11) F1) F10) (* F9 F17)) (/ F5 F18)) (- (+ (+ F17 (* (+ F11 F12) F20))
(* (- (+ F2 145.765) (/ F6 F11)) (- 133.082 F17))) (/ F11 (* F14 F20))))) (* (- (*
(- (- F6 F5) (* F3 F6)) (/ (+ (+ F1 145.765) (* F16 F10)) F18)) F12) (+ (+ F17 (*
(+ F17 F12) F20)) (* (+ F14 F12) (- (+ F1 F12) F17)))))
```

genetic programs generated by the standard genetic programming evolutionary process primarily have the ability to represent and process binary classification or one class object detection tasks. However, for multiple class object detection problems described here, where there are more than two classes of objects of interest are involved, the standard genetic programming classification strategy mentioned above cannot be applied.

In this approach, we develop a different strategy called *program classification map*, as shown in figure 9, for the multiple class object detection problems. 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.

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

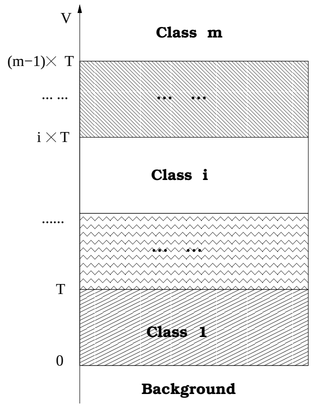


Figure 9: Mapping of program output to an object classification.

3.5 The Fitness Function

Since the goal of object detection is to achieve both a high detection rate and a low false alarm rate, we should consider a multi-objective fitness function in our GP system for multiple object object detection problems. In this approach, the fitness function is based on integration of the detection rate and the false alarm rate on the images in the training set during the learning process. Figure 10 shows the object detection procedure and how the fitness of an evolved genetic program is obtained.

The fitness of a genetic program is obtained as follows:

1. Apply the program as a moving $n \times n$ window template (n is the size of the input field) to each of the training images and obtain the output value of the program at each possible window position. Label each window position with the 'detected' object according to the object classification strategy

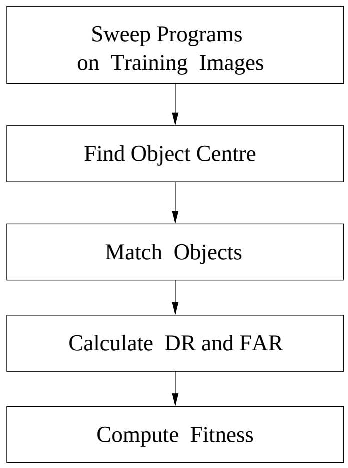


Figure 10: Object detection and Fitness Calculation

described in figure 9. Call this data structure a detection map. An object in a detection map is associated with a floating point program output.

2. Find the centres of *objects of interest only.* This is done as follows:

   - Scan the detection map for an object of interest. When one is found mark this point as the centre of the object and continue the scan $n/2$ pixels later in both horizontal and vertical directions.

3. Match these detected objects with the known locations of each of the desired true objects and their classes. A match is considered to occur if the detected object is within *tolerance* pixels of its known true location. Here, the *tolerance* is a constant parameter defined by the user.

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

5. Compute the fitness of the program as shown in equation 1.

$$fitness(FAR, DR) = W_f \times FAR + W_d \times (1 - DR) \tag{1}$$

   where W_f and W_d are constant weights which reflect the relative importance of false alarm rate versus detection rate.[3]

[3]Theoretically, W_f and W_d could be replaced by a single parameter since they have only one degree of freedom. However, the two cases of using a single and double parameters have different effects for stopping the evolutionary process. For convenience, we use two parameters.

With this design, the smaller the fitness, the better the performance. Zero fitness is the ideal case, which corresponds to the situation in which all of the objects of interest in each class are correctly found by the evolved program without any false alarms.

3.6 Main Parameters

Once a genetic programming system has been created, one must choose a set of parameters for a run. Based on the roles they play in the learning/evolutionary process, we group these parameters into three categories: search parameters, genetic parameters, and fitness parameters.

3.6.1 Search Parameters

The search parameters used here include the number of individuals in the population (*population-size*), the maximum depth of the randomly generated programs in the initial population (*initial-max-depth*), the maximum depth permitted for programs resulting from crossover and mutation operations (*max-depth*), and the maximum generations the evolutionary process can run (*max-generations*). These parameters control the search space and when to stop the learning process. In theory, the larger these parameters, the more chance of success. In practice, however, it is impossible to set them very large due to the limitations of the hardware and high cost of computation.

There is another search parameter, the size of the input field (*input-size*), which decides the size of the moving window in which a genetic program is computed in the program sweeping procedure.

3.6.2 Genetic Parameters

The genetic parameters decide the number of genetic programs used/produced by different genetic operators in the mating pool to produce new programs in the next generation. These parameters include the percentage of the best individuals in the current population that are copied unchanged to the next generation (*reproduction-rate*), the percentage of individuals in the next generation that are to be produced by crossover (*cross-rate*), the percentage of individuals in the next generation that are to be produced by mutation (*mutation-rate* = *100%* − *reproduction-rate* − *cross-rate*), the probability that in a crossover operation two terminals will be swapped *cross-term*, and the probability that in a crossover operation random subtrees will be swapped (*cross-func* = *100%* − *cross-term*).

3.6.3 Fitness Parameters

The fitness parameters include a threshold parameter (T) in the object classification algorithm, a tolerance parameter (*tolerance*) in object matching, and two constant weight parameters (W_f and W_d) reflecting the relative importance of the detection rate and the false alarm rate in obtaining the fitness of a genetic program.

3.6.4 Parameter Values

Good selection of these parameters is crucial to success. The parameter values are usually very different for various tasks. However, there does not seem to be a reliable way of *a priori* deciding these parameter values. To obtain good results, these parameter values were carefully chosen through an empirical search in experiments, as shown in table 3.

Parameter Kinds	Parameter Names	Easy Images	Coin Images	Retina Images
	population-size	100	500	700
Search	initial-max-depth	4	5	6
	max-depth	8	12	20
Parameters	max-generations	100	150	150
	input-size	14×14	24×24	16×16
	reproduction-rate	10%	1%	2%
Genetic	cross-rate	65%	74%	73%
	mutation-rate	25%	25%	25%
Parameters	cross-term	15%	15%	15%
	cross-func	85%	85%	85%
	T	100	100	100
Fitness	W_f	50	50	50
Parameters	W_d	1000	1000	3000
	tolerance (pixels)	2	2	2

Table 3: Parameters used for GP training for the three databases.

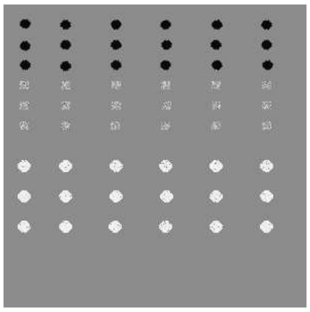
For detecting circles and squares in the easy images, for example, we set the population size to 100. On each iteration, 10 programs are created by reproduction, 65 programs by crossover, and 25 by mutation. Of the 65 crossover programs, 10 (15%) are generated by swapping terminals and 55 (85%) by swapping subtrees. The programs are randomly initialised with a maximum depth of 4 at the beginning and the depth can be increased to 8 during the evolutionary process. We also use 100, 50, 1000 and 2 as the constant parameters T, W_f, W_d and *tolerance*, which are used for the program classification and the calculation of the fitness function. The maximum generation permitted for the evolutionary process is 100 for this detection problem. The size of the input field is the same as that used in the neural network approach, that is, 14×14.

3.7 Termination Criteria

In this approach, the learning/evolutionary process is terminated when one of the following conditions is met:

- The detection problem has been solved on the training set, that is, all objects in each class of interest in the training set have been correctly detected with no false alarms. In this case, the fitness of the best individual program is zero.
- The number of generations reaches the pre-defined number, *max-generations*.

4 The Image Databases



Number of Images: 10
Object Classes: 3
Image Size 700x700

Easy (Circles and Squares)
(a)

Number of Images: 20
Object Classes: 4
Image Size 640x480

Medium Difficulty (Coins)
(b)

Number of Images: 15
Object Classes: 2
Image Size 1024x1024

Very Difficult (Retinas)
(c)

Figure 11: Object Detection Problems of Increasing Difficulty

We used three different databases in the experiments. Example images and key characteristics are given in figure 11. The databases were selected to provide detection problems of increasing difficulty. Database 1 (Easy) 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 each class. There are three classes of small objects of interest in this database: black circles ($class1$), grey squares ($class2$) and white circles ($class3$). The coin images (database 2) were intended to be somewhat harder and were taken with a CCD camera over a number of days with relatively similar illumination. In these images the background varies slightly in different areas of the image and between images and the objects to be detected are more complex, but still regular. There are four object classes of interest: the head side of 5 cent coins (class $head005$), the head side of 20 cent coins (class $head020$), the tail side of 5 cent coins (class $tail005$) and the tail side of 20 cent coins (class $tail020$). All the objects in each class have a similar size. They are located at arbitrary positions and with some rotations. The retina images (database 3) were taken by a professional photographer with special apparatus at a clinic and contain very irregular objects on a very cluttered background. The objective is to find two classes of retinal pathologies – haemorrhages and micro aneurisms. To give a clear view of representative samples of the target objects in the retina images, one sample piece of these images is presented in figure 12. In this figure, haemorrhage and micro-aneurism examples are labeled using white surrounding squares.

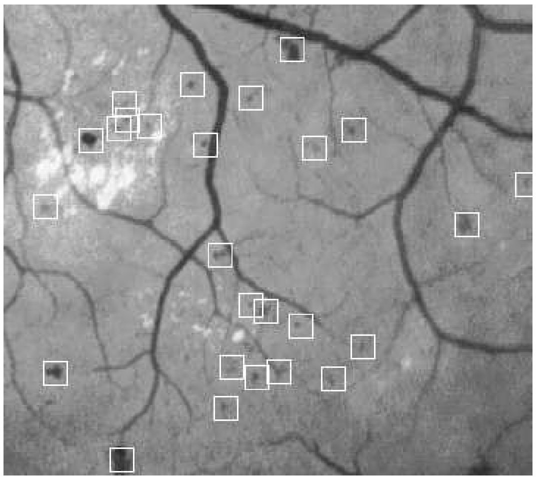


Figure 12: An enlarged view of one piece of the retina images

5 Experimental Results

We did three groups of experiments, as shown in Table 4. The first group of experiments is based on the first two terminal sets (rectilinear features and circular features) and the first function set (the four standard arithmetic functions). The second group of experiments uses the third terminal set consisting of raw "pixels" and the first function set. The third group of experiment uses the first terminal set consisting of rectilinear features and the second function set consisting of additional transcendental functions than the first function set.

Experiments	Terminal Sets	Function Sets
I	TermSet1 (Rectilinear)	FuncSet1
	TermSet2 (Circular)	FuncSet1
II	TermSet3 (Pixels)	FuncSet1
III	TermSet1 (Rectilinear)	FuncSet2

Table 4: Three groups of experiments

In these experiments, 4 out of 10 images in the easy image database are used for training and 6 for testing. For the coin images, 10 out of 20 are used for training and 10 for testing. For the retina images, 10 are used for training and 5 for testing. The results presented in this section were achieved by performing the learned/evolved genetic programs on the images in the test set.

5.1 Experiment I

This group of experiments constitutes the major experiments. The main goal here is to investigate whether this GP approach can be applied to multiple class object detection problems of increasing difficulty. The parameters used in these experiments can be seen from table 3 (section 3.6.4). The average performance of the best 10 genetic programs (evolved from 10 runs) for the easy and the coin databases, and the average performance of the best 5 genetic programs (out of 5 runs, due to the high computational cost) for the retina images are presented.

The results are compared with those obtained using a neural network approach for object detection on the same databases [24, 41]. The neural network method used was the same as that shown in section 1.1 (page 2), except that the evolutionary process was replaced by a network training process in step 3 and the generated genetic program was replaced by a trained network.

5.1.1 Easy Images

Table 5 shows the best results of the GP approach with the two different terminal sets (*GP1* with *TermSet1*, *GP2* with *TermSet2*) and the neural network (NN) method for the easy images. For *class1 (black circles)* and *class3 (grey circles)*, all the three methods achieved a 100% detection rate with no false alarms. For *class2 (grey squares),* the two GP methods also achieved 100% detection rate with zero false alarms. However the NN method had a false alarm rate of 91.2% at a detection rate of 100%.

Easy Images		Object Classes		
		Class1	Class2	Class3
Best Detection Rate(%)		100	100	100
False Alarm Rate (%)	NN	0	91.2	0
	GP1	0	0	0
	GP2	0	0	0

Table 5: Comparison of the object detection results for the easy images: the GP approaches vs the NN approach. (Input field size = 14×14; Repetitions = 10.)

5.1.2 Coin Images

Coin Images		Object Classes			
		head005	tail005	head020	tail020
Best Detection Rate(%)		100	100	100	100
False Alarm Rate (%)	NN	0	0	182	37.5
	GP1	0	0	0	0
	GP2	0	0	38.4	26.7

Table 6: Comparison of the object detection results for the coin images: the GP approaches vs the NN approach. (Input field size = 24×24; Repetitions = 10.)

Experiments with coin images gave similar results to the easy images. These are shown in table 6. Detecting the heads and tails of 5 cents (class *head005, tail005*) appears to be relatively straight forward. All the three methods achieved a 100% detection rate without any false alarms. Detecting heads and tails of 20 cent coins (class *head020, tail020*) is more difficult. While the neural network method resulted in many false alarms, the two GP methods had much better results. In particular, the GP1 method achieved the ideal results, that is, all the objects of interest were correctly detected without any false alarms for all the four object classes.

5.1.3 Retina Images

The results for the retina images are summarised in table 7. Compared with the results for the other image databases, these results are not satisfactory.[4] However, the false alarm rate is greatly improved over the neural network method.

The results over the three databases show similar patterns: the genetic programming based method always gave a lower false alarm rate than the neural network approach for the same detection rate. While the *GP2* also gave the ideal results for the easy images, it produced a higher false alarm rate

[4] With the current techniques applied in this area, detecting objects in images with a highly cluttered background is an extremely difficult problem [28, 30]. In fact, these results are quite competitive to other methods for very difficult detection problems. As a young discipline, it is quite promising for genetic programming to achieve such results.

Retina Images		Object Classes	
		haem	micro
Best Detection Rate(%)		73.91	100
False Alarm Rate (%)	NN	2859	10104
	GP1	1357	588
	GP2	1857	732

Table 7: Comparison of the object detection results for the retina images: the GP approaches vs the NN approach. (Input field size = 16×16; Repetitions = 5.)

on both the coin and the retina images than the *GP1* method. This suggests that the local rectilinear features are more effective for these detection problems than the circular features.

5.1.4 Training Times

We did these experiments on a 4 processor ULTRA-SPARC4. The training times for the three databases are very different due to various degrees of difficulty of the detection problems. The average training times used in the GP evolutionary process ($GP1$) for the easy, the coin and the retina images are 2 minutes, 36 hours and 93 hours, respectively.[5] This is much longer than the NN method, which took 2 minutes, 35 minutes and 2 hours on average. However, the GP method gave much better detection results on all the three databases. This suggests that the GP method is particularly applicable to tasks where accuracy is the most important factor and training time is seen as relatively unimportant.

5.2 Experiment II

Instead of using rectilinear and circular features (pixel statistics) in experiment I, experiment II directly uses the "pixel" values as terminals (the third terminal set). For the input field sizes of *14×14*, *24×24* and *16×16* for the easy, the coin, and the retina images, the number of terminals are 49 (*7×7*), 144 (*12×12*), and 64 (*8×8*), respectively. For the easy images, the learning took about 70 hours on a 4 processor ULTRA-SPARC4 machine to reach perfect detection performance and 78 generations were taken. The population size used was 1000, the maximum depth of the program was 30, the maximum initial depth 10, the maximum number of generations 100. For the coin images and the retina images, the situation was worse. Since a large number of terminals were used, the maximum depth of the program trees was increased to 50 for the coin images and 60 for the retina images. The population size for both databases used was 3000 with a maximum number of generations of 100. The evolutionary

[5]Even if the training time for difficult problems is very long, the time spent on applying the learned genetic program to the test set is usually very short, say, from several seconds to about one minute.

process took three weeks to complete 50 generations for the coin images and five weeks to complete 50 generations for the retina images. The best detection results were overall 22% false alarm rate at a 100% detection rate for the coin images, and about 850% false alarm rate at a detection rate of 100% for micro aneurisms in the retina images.

While these results are worse than those obtained by the *GP1* and *GP2* using the rectilinear and circular features, they are still better than the neural network approach. A major advantage of the pixel approach is that hand-crafting of programs for feature selection and extraction can be completely avoided and interesting, good features can be automatically constructed by the evolutionary process. If we use a larger population (e.g. 10000 or 50000), a larger program size (e.g. 100) and larger number of generations (e.g. 300), the results could be better according to our experience. While this is not possible to investigate with the current hardware we use, it shows a promising future direction with the improvement and development of more powerful hardware, for example, parallel or genetic hardware.

5.3 Experiment III

Instead of using the four standard arithmetic functions, this experiment focused on using the extended function set ($FuncSet2$), as shown in section 3.3.2. The parameters shown in table 3 (section 3.6.4) were used in this experiment. The best detection results for the three databases are shown in table 8.

	Easy Images			Coin Images				Retina Images	
	class1	class2	class3	head005	tail005	head020	tail020	haem	micro
Best Detection Rate (%)	100	100	100	100	100	100	100	73.91	100
False Alarm Rate (%)	0	0	0	0	0	0	0	1214	463

Table 8: Results with the second function set.

As can be seen from table 8, this function set also gave ideal results for the easy and the coin images and a better result for the retina images. The best detection rates for detecting *micro* is 100% with a corresponding false alarm rate of 463%. The best detection rate for *haem* is still 73.91% but the false alarm rate is reduced to 1214%. In addition, convergence was slightly faster for training the coin and retina images. This suggests that *dabs, sin, log* and *exp* are particularly useful for more difficult problems.

6 Discussions

6.1 Analysis of Results on the Retina Images

The genetic programming based approach achieved the ideal results on the easy images and the coin images, but resulted in some false alarms on the retina images, particularly for the detection of objects in class *haem* in which the false alarm rate was very high and more than a quarter of the real objects in this class were not detected by the evolved genetic program.

We identified two possible reasons for the results on the retina images being worse than the results on the easy and the coin images. The first reason concerns the complexity of the background. In the easy and coin images, the background is relatively uniform, whereas in the retina images it is highly cluttered. In particular, the background of the retina images contains many objects, such as veins and other anatomical features, that are not members of the two classes of interest (micro aneurisms and haemorrhages). These objects of non-interest must be classified as "background", in just the same way as the genuine background. The more complex the boundary between classes in the input space, the more complex an evolved program has to be to distinguish the classes. It may be that the more complex background class in the retina images requires a more complex evolved program than the GP system was able to discover. It may even be that the set of terminals and functions is not adequate/sufficient to represent an evolved program to distinguish the objects of interest from such a rich background.

The second possible reason concerns the variation in size of the objects. In the easy and coin images, all of the objects in a class have similar sizes, whereas in the retina images, the sizes of the objects in each class vary. This variation means that the evolved genetic program must cover a more complicated region of the input space. The sizes of the *micro* objects vary from 3×3 to 5×5 pixels and the sizes of the *haem* objects vary from 6×6 to 14×14 pixels. Given the size of the input field (16×16) and the choice of terminals, the variance in the size of the *haem* objects is particularly problematic, since it ranges from just one quarter of the input field (hence entirely inside the central detection region) to almost the entire input field. The fact that the performance on the *haem* class is worse than the performance on the *micro* class (especially in experiment III) provides additional evidence that the size variation is a cause of the poor performance.

The first reason suggests that the current approach is limited on images containing cluttered backgrounds. One possible modification to address this limitation explore is to evolve multiple programs rather than a single program, either having a separate program for each class of interest, or have several programs to exclude different parts of the background. Another possible modification is to extend the terminal set and/or function set to enrich the expressive power of the evolved programs.

The second reason suggests that the current approach has limited applicability to scale invariant detection problems. This would not be surprising, given the current set of terminals and functions. In particular, although the pixel statistics used in the rectilinear and circular terminal sets are robust to small variations in scale, they are not robust to large variations. We will explore alternative pixel statistics that are more robust to scale variations, and also function sets that would allow disjunctive programs that could better represent classes that contained objects of several different size ranges.

6.2 Analysis of Evolved Programs

This section gives a brief analysis of the best generated programs for the three databases. The genetic programs evolved by the *GP1* in experiment *I* are used as examples.

6.2.1 Easy Images

Figure 13 shows three sample good evolved programs for the easy images.[6] All of these programs achieved the ideal results: all of the circles and squares were correctly detected with no false alarms.

Program 1:	$\dfrac{\dfrac{F_3 \cdot F_{14} \cdot F_{15}}{F_6 \cdot F_{14}} - F_{19} - F_7 \cdot F_{10} \cdot F_{17} \cdot F_{16} \cdot F_{18} + \dfrac{F_5}{F_5} \cdot F_{14} + \dfrac{F_3}{F_5}}{\dfrac{F_3}{F_5} \cdot \dfrac{\frac{F_5}{F_6}}{\frac{F_{11}}{F_{15}}} + \dfrac{\frac{F_{19}}{F_6}}{\frac{F_5}{F_{15}}}}$
Program 2:	$\dfrac{F_{18}}{F_3 + F_5 - F_3 \cdot F_5} - F_4 \cdot F_{16} \cdot \dfrac{F_{18} \cdot (F_5 + F_{18}) - (F_7 + F_4) + F_{10} - F_{19}}{F_4 \cdot F_{16}}$
Program 3:	$\dfrac{(F_{16} + F_7) \cdot F_{15} \cdot F_4}{F_{19} - \dfrac{F_{13} \cdot F_5}{\frac{F_{18}}{F_{10}}} + F_{11}} \cdot \dfrac{F_9}{F_9 \cdot F_4}$

Figure 13: Three sample generated programs for simple object detection in the easy images.

[6]These programs were the direct mathematical conversion of the original LISP format programs evolved by the evolutionary process. The LISP format of the first program is, for example, `(/ (+ (- (- (/ (* (* F3 F14) F15) (* F6 F14)) F19) (* (* (* (* F7 F10) F17) F16) F18)) (+ (* (/ F5 F5) F14) (/ F3 F5))) (+ (* (/ F3 F5) (/ (/ F5 F6) (/ F11 F15))) (/ (/ F19 F6) (/ F5 F15))))`. Note that we did not simplify them — Simplification of evolved genetic programs is beyond the goal of this paper.

There are several things we can note about these programs. Firstly, the programs are not trivial, and are decidedly non-linear. It is hard to interpret even these programs for the easy images. Secondly, the programs use many, but not all, of the terminals, but do not use any constants. There are no groups of the terminals that are unused — both the means and standard deviations of both the square regions and the lines are used in the programs, so it does not appear that any of the terminals could be safely removed. Thirdly, although the programs are not in their simplest form (for example, the factor F_5/F_5 could be removed from the first program), there is not a large amount of redundancy, so that the GP search is finding reasonably efficient programs.

6.2.2 Coin Images

In addition to the program shown in figure 8, we present another generated program in figure 14, which also performed perfectly for the coin images.

$$\frac{\begin{array}{c}\dfrac{F_{10}\cdot F_{12}-F_9-F_2+F_{12}\cdot\dfrac{F_{10}\cdot F_{12}-F_9}{87.251}-[F_2-(\dfrac{F_{12}\cdot F_{12}-F_{17}-F_2}{F_1}+\dfrac{87.251}{\frac{F_{19}}{F_5}})]}{F_{11}\cdot(\dfrac{F_9}{F_{16}}-\dfrac{F_{17}-F_2+F_{12}\cdot F_{12}-F_{11}}{F_{16}})-F_{15}+F_8} \\ F_{15}\cdot\dfrac{F_{15}}{F_8}\end{array}}{F_{17}-F_2}+\frac{F_9}{\frac{F_{13}-F_{15}}{F_1}}+\frac{87.251}{\frac{F_{19}}{F_5}}+F_{10}\cdot F_{12}-F_9$$

Figure 14: A sample generated program for regular object detection in the coin images.

Compared with those for the easy images, these programs are more complex, which reflects the greater difficulty of the detection problem in the coin images. One difference is that these programs also contain constants. The set of possible programs is considerably expanded by allowing constants as well as the terminals, but the search for good values for the constants is difficult. Our current GP is biased so that constants are only introduced rarely, but it is clear that the detection problem on the coin images is sufficiently difficult to require some of these constants.

6.2.3 Retina Images

One evolved genetic program for the retina images is presented in figure 15. (The program is presented in LISP format rather standard format because of its complexity.)

This program is much more complex than any of the programs for the easy and the coin images. The program uses all 20 terminals, and 8 constants. It does not seem possible to make any meaningful

```
(* (* (- (/ F6 (+ (* (/ (* F2 (/ (* F6 (+ F1 (- F10 F15)))
                                 (- (- F18 F17) (- F19 87.05))))
                        (+ 17.0792 (+ F9 F14)))
                     (/ (+ F19 (* (+ (+ F11
                                        (- (* (- (- F15 F18) (+ 40.58 F16))
                                              (- (* F13 (+ (/ 57.64 F16) F13))
                                                 (- F9 F6)))
                                           (/ (* F3 F1) F1)))
                                     (* (- (* (- (/ (+ (+ F18 (+ (/ (/ F14 F6)
                                                                   (+ F6 F1))
                                                                89.70))
                                                       (* F10 F12)) F2) F9)
                                              (+ (+ F16 14.75) F9)) F18)
                                        (/ (/ F13 F1) (* (+ F6 F12) F9))))
                               (+ F16 F8)))
                         (+ (- (- (+ (/ F10 (* F9 F6)) F13) F10) F18)
                            (+ (* (- (+ F1 F2) (+ F17 F8)) F5)
                               (* (* F20 F16) F10)))))
                   (* (+ (- (* (+ F11
                                  (+ (* F14 F3)
                                     (/ F15 (/ (+ (* F2 14.5251)
                                                  (* (* (/ (* F18
                                                             (/ (* F2 F13) F15))
                                                           F1)
                                                        (/ (/ F11 F13) (/ F7 F5)))
                                                     (+ (+ F18 (* F2 F13))
                                               (/ F8 F12))))
                                         F17)))) F11) F16)
                         (* (- F1 (+ F3 F8)) F5))
                      (/ (+ (- F7 F20) F18) F20))))
             (* (* (* (* F2 F13) F2)
                   (/ (* F4 (/ (* F2 F13) F15)) (* F18 F12)))
                (* F14 F2)))
          (+ (+ (- (+ (- F19 F3) F2) F7) (- (+ F8 F17) F18))
             (/ (+ F15 60.10)
                (* (* F1 (/ (/ F12 (- (+ (/ (/ F12 F13) (/ F15 F5)) F17) F18))
                            (/ F7 F5))) F8))))
       (* (/ (* F10 (/ (* F2 F13) F15)) F18)
          (* (* (* (* F2 F2) (/ (/ (/ F18 (+ F1 F2)) F13)
                                (/ (/ (- F15 96.16) (* F4 14.53)) F5))) F4)
             (/ (/ F12 F13) (/ F1 (+ (/ F10 F1) F4))))))
```

Figure 15: A sample generated program for very difficult detection problems in the retina images

interpretation of this program. It may be that with high level, domain specific features and domain specific functions, it would be possible for the GP system to construct simpler and more interpretable programs; however, this would against one of the goals of this paper which is to investigate domain independent features.

Even the best programs for the retina images gave quite a high number of false alarms, and it appears that the 20 terminals and four standard arithmetic functions are not sufficient for constructing programs for such difficult detection problems. Nonetheless, the program above still had much better performance than a neural network with the same input features.

6.3 Analysis of Classification Strategy

As described in figure 9, we used a program classification map as the classification strategy. In this map, a constant T was used to give "fixed" size ranges for determining the classes of those objects from the output of the program. The parameter can be regarded as a *threshold* or a class boundary parameter. Using just a single value for T forces most of the classes to have an equal possible range in the program output, which might lead to a relatively long time of evolution. A natural question to raise is whether we can replace the single parameter T with a set of parameters, say, $T_1, T_2, ..., T_m$, one for each class of interest.

To answer this question, we ran a set of experiments on the easy images with three parameters, T_1, T_2 and T_3, for the thresholds in the program classification map. The experiments showed that some sets of values of the parameters resulted in an ideal performance but other sets of values did not. Also, the learning/evolutionary process converged very fast with some sets of values but very slowly with others. However, the results of the experiments gave no guidelines for selecting a good set of values for these parameters. In some cases, using separate parameters for each threshold may lead to a better performance than using a single parameter, but appropriate values for the parameters need to be empirically determined. In practice, this is difficult because there is no *a priori* knowledge in most cases for setting these parameters.

We also tried an alternative classification strategy, which we called *multiple binary map*, to classify multiple classes of objects. In this method, we convert a multiple classification problem to a set of binary classification problems. Given a problem L with m classes $L = \{c_1, c_2, ..., c_m\}$, the problem is decomposed into $L_1 = \{c_1, other\}$, $L_2 = \{c_2, other\}$, ..., $L_m = \{c_m, other\}$, where c_i denotes the ith class of interest and *other* refers to the class of non-objects of interest. In this way, a multiple class object detection problem is decomposed into a set of one class object detection tasks and genetic programming is applied to each of the sub sets to obtain the detection result for a particular class of interest. We tested this method on the detection problems in the three image databases and the results were similar to those of the original experiments. In other words, genetic programming with this classification strategy works for multiple class object detection problems. One disadvantage of this method is that several genetic programs have to be evolved. On the other hand, the genetic programs may be simpler which may reduce the training time for each program. In fact, for the coin images problem, a considerably shorter total training time was required to create a set of one class programs than to create a single multiple class program. A more detailed discussion of this method is outside the goal of paper, and is left to future work.

6.4 Analysis of Crossover and Mutation Rates

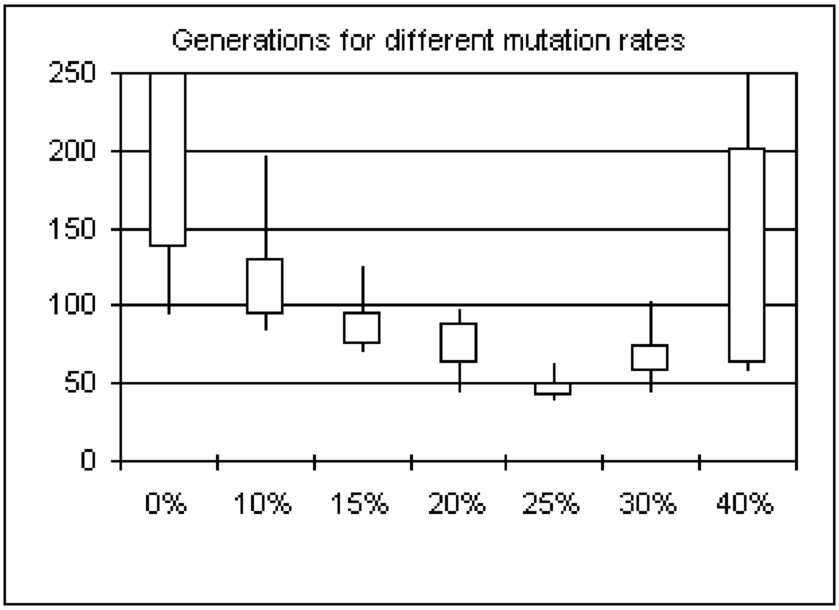


Figure 16: convergence *vs* mutation rate

Some genetic programming researchers argue that mutation is useless and should not be used in genetic programming [17], while some others insist that a high mutation rate would help the GP evolution converge [8, 11]. To investigate the effects of mutation in genetic programming for multiclass object detection problems, we carried out ten experiments for different rates of mutation *vs* crossover on the easy images, as shown in figure 16. The reproduction rate was held constant at 10%, and the mutation rate varied from 0% to 40%. The graph shows the distribution of the number of generations to convergence by a box-and-whisker plot with the limits of the central box at the 30% and 70% percentiles. With both 0% and 40% mutation, the search sometimes did not converge within the limit of 250 generations. There was a clear effect of the mutation rate on the number of generations to convergence. The best mutation rate was 25%, where only 48 generations on average were required to find a good solution, with slower convergence at both lower and higher mutation rates. Experiments on the coin and the retina images gave a similar trend. This suggests that, in genetic programming for multiple class object detection problems described in this paper, mutation plays an important role of keeping the diversity of the population and the convergence could be sped up when an appropriate mutation rate was used. However, such a good mutation rate is generally task dependent, and 15%-30% is a good choice for similar tasks.

6.5 Analysis of Reproduction

In early genetic programming, the reproduction rule did a probabilistic selection of genetic programs from the current population based on their fitness and allowing them to survive by copying them into the new population. The better the fitness, the more likely the individual program is to be selected [19, 37]. However, this mechanism does not guarantee that the best program will survive. An alternative reproduction rule is one that removes the probabilistic element, and simply reproduces the best n genetic programs from the current population. We ran experiments on the easy images with both reproduction rules and plotted the best fitness in each generation (see figure 17). The dotted curve shows the best fitness with the probabilistic reproduction rule. Over the 100 generations, there are four clear intervals (at generation 7, 22, 45 and 67) where the fitness got worse rather than better, which delayed the convergence of learning. In contrast, the deterministic reproduction rule had a steady improvement in fitness. Furthermore, the deterministic reproduction rule converged on an ideal program after just 71 generations, while the probabilistic reproduction rule had still not converged on an ideal program after 100 generations. (In fact, the fitness did not improve at all during the final 30 generations!) Clearly, the new reproduction rule greatly improved the training speed and convergence.

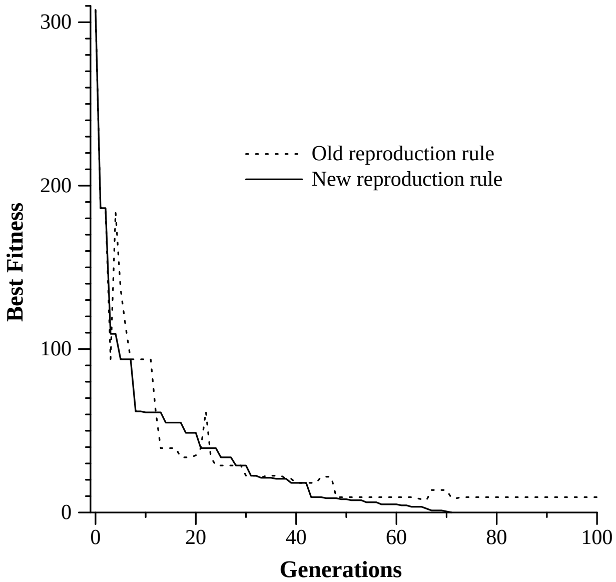


Figure 17: Training easy images based on the old and the new reproduction rules

7 Conclusions

The goal of this paper was to develop a domain independent, learning/adaptive approach for detecting small objects of multiple classes in large images based on genetic programming. This goal was achieved by the use of genetic programming with a set of domain independent pixel statistics as terminals, a number of standard operators as functions, and a linear combination of the detection rate and false alarm rate as the fitness measure. A secondary goal was to compare the performance of this method with a neural network method.

The paper differs from most work in object detection in two ways. Most work addresses the one class problem, that is, *object* versus *non-object*, or *object* versus *background*. This paper has shown a way of solving a multiple class object detection problem without breaking it into a collection of one class problems. Also, most current research uses different algorithms in multiple independent stages to solve the localisation problem and the classification problem; in contrast, this paper uses a single learned genetic program for both object classification and object localisation in multiple class object detection problems.

The paper has shown that genetic programming can be successfully applied to multiple class object detection problems. The genetic programming method gave better detection performance than a neural network based approach for the same problem set. The approach appears to be applicable to detection problems of varying difficulty.

The classification strategy for multiple class detection/classification problems is more difficult than one class object detection problems or binary classification problems. The paper presented two strategies — the program classification map and multiple binary maps — for multiple class problems. Although developed in this paper for multiclass object detection problems, they are general strategies for multiple class classification problems.

The experiments showed that mutation does play an important role in the three multiple class object detection tasks. This is in contrast to Koza's early claim that genetic programming does not need mutation. For genetic programming applied to multiple object object detection problems, the experiments suggest that a 15%-30% mutation rate would be a good choice.

The experiments also identified some limitations of the particular approach taken in the paper. The first limitation concerns the choice of input features and the function set. For the simple and medium difficulty object detection problems, the 20 regional/rectilinear features and four standard arithmetic functions performed very well; however, they were not adequate for the most difficult object detection task. In particular, they were not adequate for detecting classes of objects with a range of sizes. Further work will be required to discover more effective domain independent features and function sets, especially ones that provide some size invariance.

A second limitation is the high training time required. One aspect of this training time is the experimentation required to find good values of the various parameters for each different problem. The genetic programming method appears to be applicable to multiple class object detection tasks where accuracy is the most important factor and training time is seen as relatively unimportant. Further experimentation may discover more effective ways of setting the parameters which will reduce the training time.

Subject to these limitations, the paper has demonstrated that genetic programming can be used effectively for the multiple class detection problem and provides more evidence that genetic programming has a great potential for application to a variety of problems in the real world.

Acknowledgement

We would like to thank Dr. James Thom at RMIT university and Dr. Zhi-Qiang Liu at the University of Melbourne for a number of useful discussions. Thanks also to Peter Wilson whose basic genetic programming package was used in this project and to Chris Kamusinski who provided and labelled the retina images.

References

[1] 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.

[2] 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.

[3] E. B. Baum and D. Haussler. What size net gives valid generalisation? *Neural Computation*, 1:151–160, 1989.

[4] 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.

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

[6] David P. Casasent and Leonard M. Neiberg. Classifier and shift-invariant automatic target recognition neural networks. *Neural Networks*, 8(7/8):1117–1129, 1995.

[7] Olivier Faugeras. *Three-Dimensional Computer Vision – A Geometric Viewpoint.* The MIT Press, 1993. ISBN 0-262-06158-9.

[8] Matthias Fuchs. Crossover versus mutation: An empirical and theoretical case study. In John R. Koza, Wolfgang Banzhaf, and et al., editors, *Genetic Programming 1998, Proceedings of the third annual conference*, pages 78–85, University of Wisconsin, Madison, July 1998. Morgan Kaufmann Publisher.

[9] Paul D. Gader, Joseph R. Miramonti, Yonggwan Won, and Patrick Coffield. Segmentation free shared weight neural networks for automatic vehicle detection. *Neural Networks*, 8(9):1457–1473, 1995.

[10] 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.

[11] K. Harries and P. Smith. Exploring alternative operators and search strategies in genetic programming. In John R. Koza, Wolfgang Banzhaf, and et al., editors, *Proceedings of the second international conference on Genetic Programming (GP-97)*, pages 147–155. Morgan Kaufmann Publisher, 1997.

[12] Christopher Harris. *An Investigation into the Application of Genetic Programming Techniques to Signal Analysis and Feature Detection.* PhD thesis, the University of London, London, UK, September 1997. Department of Computer Science, University College London.

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

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[21] Shang-Hung Lin, Sun-Yuan Kung, and Long-Ji lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks*, 8(1):114–132, Jan 1997.

[22] 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.

[23] 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.

[24] Nagendra Rai. Pixel statistics in neural networks for object detection. Minor thesis, RMIT University, Department of Computer Science, 2001.

[25] 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.

[26] Steven K. Rogers, John M. Colombi, Curtis E. Martin, James C. Gainey, Ken H. Fielding, Tom J. Burns, Dennis W. Ruck, Matthew Kabrisky, and Mark Ocley. Neural networks for automatic target recognition. *Neural Networks*, 8(7/8):1153–1184, 1995.

[27] H. L. Roitblat, W. W. L. Au, P. E. Nachtigall, R. Shizumura, and G. Moons. Sonar recognition of targets embedded in sediment. *Neural Networks*, 8(7/8):1263–1273, 1995.

[28] Michael W. Roth. Survey of neural network technology for automatic target recognition. *IEEE Transactions on neural networks*, 1(1):28–43, March 1990.

[29] Jamie R. Sherrah, Robert E. Bogner, and Abdesselam Bouzerdoum. The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. 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 304–312, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] Allen M. Waxman, Michael C. Seibert, Alan Gove, David A. Fay, Ann Marie Bernandon, Carol Lazott, William R. Steele, and Robert K. Cunningham. Neural processing of targets in visible, multispectral ir and sar imagery. *Neural Networks*, 8(7/8):1029–1051, 1995.

[37] Peter Wilson. Development of genetic programming strategies for use in the robocup domain. Technical report, Department of Computer Science, RMIT, 1998. Honours Thesis.

[38] 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.

[39] Philip Winter, Shahab Sokhansanj, Hugh C.Wood, and William Crerar. Quality assessment and grading of lentils using machine vision. In *Agricultural Institute of Canada Annual Conference*, Saskatoon, SK S7N 5A9, Canada, July 1996. Canadian Society of Agricultural Engineering. CASE paper No. 96-310.

[40] Yonggwan Won, Paul D. Gader, and Patrick C. Coffield. Morphological shared-weight networks with applications to automatic target recognition. *IEEE Transactions on neural networks*, 8(5):1195–1203, September 1997. ISSN 1045-9227.

[41] Mengjie Zhang and Victor Ciesielski. Centred weight initialisation in neural networks for object detection. In Jenny Edwards, editor, *Australian Computer Science Communications (Computer Science 99), Vol. 21, No. 1*, pages 39–50, Auckland, January 1999. Springer. Proceedings of the Twenty Second Australasian Computer Science Conference.