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

School of Mathematics, Statistics and Computer Science
Computer Science

Improving Fitness Function and Optimising Training Data in GP for Object Detection

Mengjie Zhang, Malcolm Lett

Technical Report CS-TR-06/8
February 2006

School of Mathematics, Statistics and Computer Science
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 Mathematics, Statistics and Computer Science
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

Improving Fitness Function and Optimising Training Data in GP for Object Detection

Mengjie Zhang, Malcolm Lett

Technical Report CS-TR-06/8
February 2006

Abstract

This paper describes an approach to the refinement of a fitness function and the optimisation of training data in genetic programming (GP) for object detection particularly object localisation problems. The fitness function uses the weighted F-measure of a genetic program and considers the localisation fitness values of the detected object locations. To investigate the training data with this fitness function, we categorise the training data into four types: *exact centre, close to centre, include centre*, and *background*. The approach is examined and compared with an existing fitness function on three object detection problems of increasing difficulty. The results suggest that the new fitness function outperforms the old one by producing far fewer false alarms and spending much less training time and that the first two types of the training examples contain most of the useful information for object detection. The results also suggest that the complete background type of data can be removed from the training set.

Keywords Fitness function, training examples, object detection, object classification, object recognition, object localisation

Author Information

Both authors are academic staff members and postgraduate students in computer science in the School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, New Zealand.

1 Introduction

As more and more images are captured in electronic form, the need for programs which can detect objects of interest in a database of images is increasing. For example, it may be necessary to detect all tumours in a database of x-ray images, find all cyclones in a database of satellite images, or detect all tanks or helicopters in a set of images [1, 2, 3, 4].

Genetic programming (GP) is a relatively recent and fast developing approach to automatic programming [5, 6, 7]. In GP, solutions to a problem can be represented in different forms but are usually interpreted as computer programs. Darwinian principles of natural selection and recombination are used to evolve a population of programs towards an effective solution to specific problems. The flexibility and expressiveness of computer program representation, combined with the powerful capabilities of evolutionary search, make GP an exciting new method to solve a great variety of problems. GP has been applied to a range of object recognition tasks with some success [5, 8, 9, 10, 11].

Finding a good fitness function for a particular object detection problem is an important but difficult task in developing a GP system. Various fitness functions have been devised for object detection, with varying success [5, 9, 11, 12, 13]. These tend to combine many parameters using scaling factors which specify the relative importance of each parameter, with no obvious indication of what scaling factors are good for a given problem. Many of these fitness functions for localisation require clustering to be performed to group multiple localisations of single objects into a single point before the fitness is determined [14, 13, 12]. Other measures are then incorporated in order to include information about the pre-clustered results (such as how many points have been found for each object). While some of these systems achieved good detection rates, many of them resulted in a large number of false alarms. In particular, the clustering process during the evolutionary process made the training time very long.

Organising training data is critical to any learning approaches. The previous approaches in object detection tend to use all possible positions of the large image in training an object detector. However, this usually requires a very long training time due to the use of a large number of positions on the background.

This paper aims to investigate a new fitness function and a new way to optimise the training data in GP for object detection, in particular localisation, with the goal of improving the detection performance and refining training examples. The approach will be examined on a sequence of object detection problems of increasing difficulty.

The remainder of this paper is organised as follows. Section 2 describes the GP approach to object detection. Section 3 describes data sets and experiment configurations. Section 4 presents the new fitness function. Section 5 investigates the training data. Finally, we draw conclusions in section 6.

2 GP Applied to Object Detection

The process for object detection is shown in Figure 1. A raw image is taken and a trained localiser applied to it, producing a set of points found to be the positions of these objects. Single objects could have multiple positions ("localisations"), however ideally there would be exactly one localisation per object. Regions of the image are then "cut out" at each of the positions specified. Each of these cutouts are then classified using the trained classifier.

This method treats all objects of multiple classes as a single "object of interest" class for the purpose of localisation, and the classification stage handles attaching correct class labels. Compared with the single-stage approach [10, 11], this approach has the advantage that the training is easier for both stages as a specific goal is focused on the training of each of the two

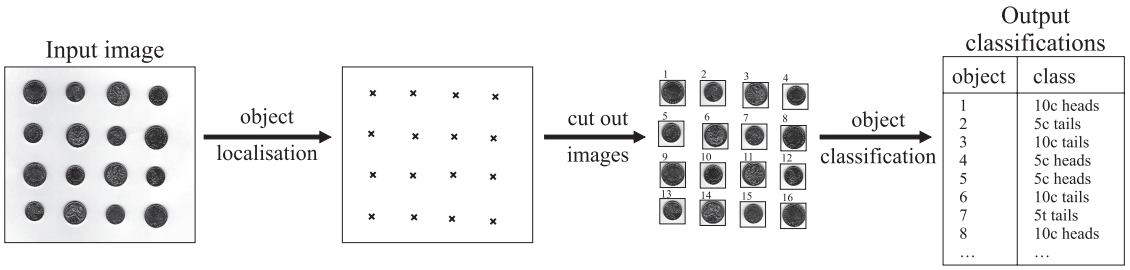


Figure 1: Object Detection Process.

stages. The first is tailored to achieving results as close to the object centres as possible (to achieve high "positional accuracy"), while the second is tailored to making all classifications correct (high "classification accuracy").

The object localisation stage is performed by means of a window which sweeps over the whole image, and for each position extracts the features and passes them to the trained localiser. The localiser then determines whether each position is an object or not (i.e. background).

Our work will focus on object localisation using genetic programming. Figure 2 shows an overview of this approach, which has a learning process and a 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. The programs are applied at many sampled positions within the images in the training set to detect the objects of interest. If the program localiser returns a value greater than or equal to zero, then this position is considered the centre of an object of interest; otherwise it is considered background. In the test procedure, the best evolved genetic program obtained in the learning process is then applied, in a moving window fashion, to the whole images in the test set to measure object detection performance.

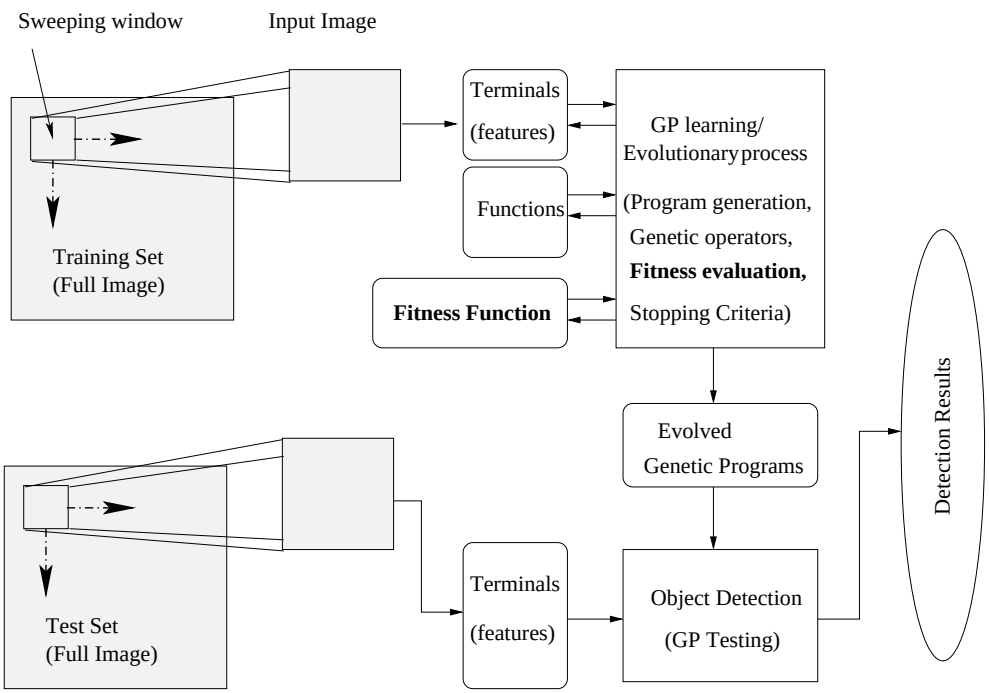
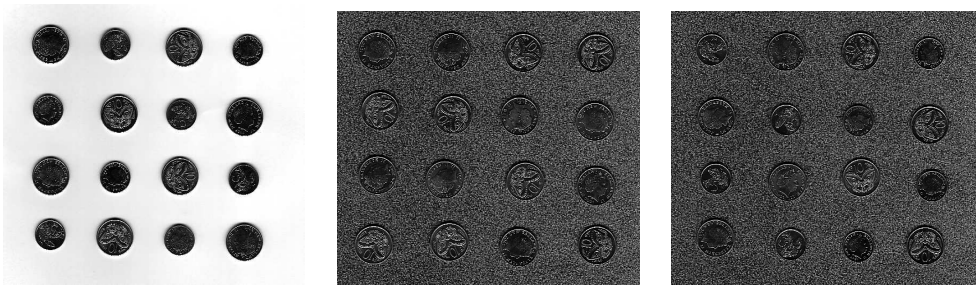


Figure 2: An overview of the GP approach for object detection.

3 Experiment Design and Configurations

3.1 Data Sets

We used three image data sets of New Zealand 5 and 10 cent coins in the experiments. Examples are shown in Figure 3. The data sets are intended to provide object localisation/detection problems of increasing difficulty. The first data set (*easy*) contains images of tails and heads of 5 and 10 cent coins against an almost uniform background. The second (*medium difficulty*) is of 10 cent coins against a noisy background, making the task harder. The third data set (*hard*) contains tails and heads of both 5 and 10 cent coins against a noisy background.



(a) (b) (c)

Figure 3: Sample images in the three data sets. (a) Easy; (b) Medium difficulty; (c) Hard.

We used 24 images for each data set in our experiments and equally split them into three sets: a training set for learning good genetic programs, a validation set for monitoring the training process to avoid overfitting, and a test set to measure object detection performance.

3.2 GP System Configurations

In this system, we used tree structures to represent genetic programs [6]. The ramped half-and-half method [5] was used for generating programs in the initial population and for the mutation operator. The proportional selection mechanism and the reproduction, crossover and mutation operators were used in evolution.

For object detection problems, terminals generally correspond to image features. In this approach, the features are extracted by calculating the mean and standard deviation of pixel values within several circular regions. This set of features has the advantages of being rotationally invariance. In addition, we also used a constant terminal. Note that finding a good set of features is beyond the goal of this paper.

The function set contains the four standard arithmetic and a conditional operation.

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

The $+$, $-$, and $*$ operators are usual addition, subtraction and multiplication, while $/$ represents "protected" division. The *if* function returns its second argument if the first argument is positive or returns its third argument otherwise.

Construction of a new fitness function and investigation of the training data are the main focuses of this paper, which will be described in the next sections.

We used a population of 500 genetic programs for evolution in each experiment run. The reproduction rate, crossover rate and mutation rate were 5%, 70% and 25%, respectively. The program size was initialised to 4 and it could increase to 8 during evolution. The system run 50 generations unless it found a solution, in which case the evolution was terminated

early. A total number of 100 runs were performed for each fitness function on each data set and the average results are presented in the next two sections.

4 Fitness Function

During the evolutionary process for object detection, we expect that the evolved genetic programs only detect the objects when the sweeping window is centred over these objects. However, in the usual case, these evolved genetic programs will also detect some "objects" not only when the sweeping window is within a few pixels of the centre of the target objects, but also when the sweeping window is centred over a number of cluttered pieces of background. Clearly, these "objects" are not those we expected but are false alarms.

As the goal is to detect the target objects with no or a small number of false alarms, many GP systems uses a combination of detection rate and false alarm rate or recall and precision as the fitness function. For example, a previous GP system uses the following fitness function [10]:

$$fitness_{CBF} = A \cdot (1 - DR) + B \cdot FAR + C \cdot FAA \tag{1}$$

where DR, FAR, and FAA are detection rate (the number of small objects correctly reported by a detection system as a percentage of the total number of actual objects in the images), false alarm rate (also called *false alarms per object*, the number of non-objects incorrectly reported as objects by a detection system as a percentage of the total number of actual objects in the images), and false alarm area (the number of false alarm pixels which are not object centres but are incorrectly reported as object centres before clustering), respectively, and A, B, C are constant weights which reflect the relative importance of detection rate versus false alarm rate versus false alarm area.

Different evolved genetic programs typically result in different numbers of false alarms and such differences should be reflected when these programs are evaluated by the fitness function. For example, some requirements shown in figure 4 should be considered. In this figure, the circles are target objects and squares are large images or regions. A cross (x) represents a detected object. In each of the five cases, the program associated with the left figure should be considered better than that with the right.

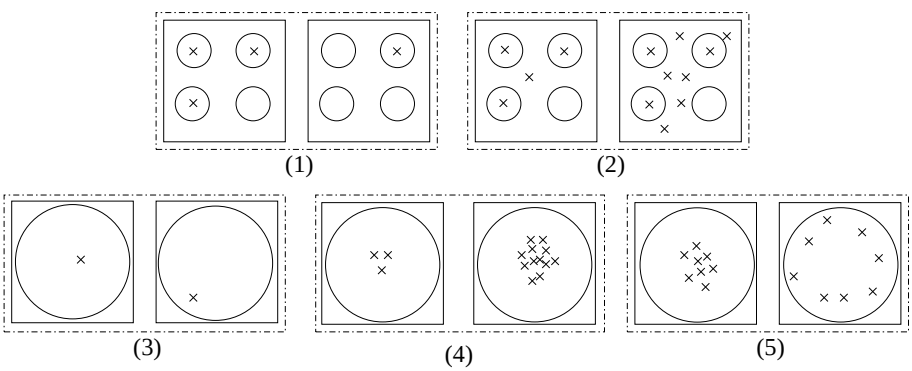


Figure 4: Examples of the design considerations of the fitness function.

While this fitness function has considered case 1, and partially considered cases 2 and 4, it does not take into accounts of cases 3 and 5. Since this method used clustering before calculating the fitness, we refer to it as *clustering based fitness*, or CBF for short.

4.1 A New Fitness Function — RLWF

During the object localisation process, a genetic program might consider many pixel positions in an image as object centres and we call each object centre localised in an image by a genetic program a *localisation*.

Unlike the previous fitness function CBF, the new fitness function is based on a "Relative Localisation Weighted F-measure" (RLWF), which attempts to acknowledge the worth/goodness of individual localisations made by the genetic program. Instead of using either correct or incorrect to represent a localisation, each localisation is allocated a weight (referred to as the *localisation fitness, LF*) which represents its individual worth and counts towards the overall fitness.

Each weight is calculated based on its relative location, or the distance of the localisation from the centre of the closest object, as shown in Equation 2.

$$\mathrm{LF}(x,y) = \begin{cases} 1 - \frac{\sqrt{x^2+y^2}}{r} & , \text{ if } \sqrt{x^2+y^2} \leq r \\ 0 & , \text{ otherwise} \end{cases} \tag{2}$$

where $\sqrt{x^2+y^2}$ is the distance of the localisation position (x, y) from target object centre, and r is called the "localisation fitness radius", defined by the user. In this system, r is set to a half of the square size of the input window, which is also the radius of the largest object.

In order to deal with all the situations in the five design requirements, we used the localisation fitness to construct our first new fitness function, as shown in Equations 4 to 5. The precision and recall are calculated by taking the localisation fitness for all the localisations of each object and dividing this by the total number of localisations or total number of target objects respectively.

$$\mathrm{WP} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{L_i} \mathrm{LF}(x_{ij}, y_{ij})}{\sum_{i=1}^{N} L_i} \tag{3}$$

$$\mathrm{WR} = \frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{L_i} \mathrm{LF}(x_{ij}, y_{ij})}{L_i}}{N} \tag{4}$$

$$\mathrm{fitness}_{RLWF} = \frac{2 \times \mathrm{WP} \times \mathrm{WR}}{\mathrm{WP} + \mathrm{WR}} \tag{5}$$

where N is the total number of target objects, (x_{ij}, y_{ij}) is the position of the j-th localisation of object i, L_i is number of localisations made to object i, WP and WR are the weighted precision and recall, and $\mathit{fitness}_{LFWF}$ is the localisation fitness weighted F-measure, which is used as the new fitness function.

4.2 Results

To give a fair comparison for the two fitness functions, the "localisation recall (LR) and precision (LP)" were used to measure the final object detection accuracy on the test set. LR is the number of objects with one or more correct localisations within the localisation fitness radius at the target object centres as a percentage of the total number of target objects, and LP is the number of correct localisations which fall within the localisation radius at the target object centres as a percentage of the total number of localisations made. In addition, we also check the "Extra Localisations" (ExtraLocs) for each system to measure how many extra localisations were made for each object. The training efficiency of the systems is measured with the number of training generations and the CPU (user) time in second.

Table 1: Results of the GP systems with the two fitness functions.

Dataset	Fitness function	Test Accuracy			Training Efficiency	
		LR (%)	LP (%)	ExtraLocs	Generations	time(sec)
Easy	CBF	99.99	98.26	324.09	13.69	178.99
	RLWF	99.99	99.36	98.35	36.44	111.33
Medium	CBF	99.60	83.19	804.88	36.90	431.94
	RLWF	99.90	94.42	95.69	34.35	105.56
Hard	CBF	98.22	75.54	1484.51	31.02	493.65
	RLWF	99.53	87.65	114.86	33.27	107.18

Table 1 shows the results of the GP systems with the two fitness functions. The results on the easy data set show that both the fitness functions achieved almost perfect test accuracy. Almost all the objects of interest in this data set were successfully localised with very few false alarms (both LR and LP are very close to 100%), reflecting the fact that the detection task in this data set is relatively easy. However, the extra locations and the training time resulted from the two approaches are quite different. The new fitness function (RLWF) produced a far fewer number of extra localisations per object than clustering based fitness function (CBF) and the gap between them is significant. Although the CBF approach used only 13.69 generations on average, which are considerably fewer than the new RLWF, it actually spent about 50% longer training time. This confirms our early hypothesis that the clustering process in the CBF approach is time consuming and the approach with the new fitness function is more efficient than that with CBF.

The results on the other two data sets show a similar pattern in terms of the number of extra localisations and training time. The systems with RLWF always produced a significantly fewer number of extra localisations and a much short training time than CBF. In addition, although almost all the objects of interest in the large images were successfully detected (LRs are almost 100%), the localisation precisions achieved by RLWF were significantly better than CBF, suggesting that the new fitness function outperforms the existing one in terms of reducing false alarms.

As expected, performance on the three data sets deteriorated as the degree of difficulty of the object detection problem was increased.

5 Optimising Training Data

We could train the detection system with a full set of cutouts taken from a window at all possible positions over the training images. However, for a set of large images, this can create a huge number of training examples making the training time unsuitably long. While we can reduce the total number of training examples using a combination of hand-chosen and randomly chosen examples [14], in this approach, we focus on investigating whether some examples are better than others and how we pick up better examples.

5.1 Four Training Data Types

The traditional approaches usually use *positive* and *negative* examples. The former refers to the exact object examples and the latter refers to those for the background [9, 10, 13]. However, this did not consider those with some piece of object and some piece of background. In this approach, we identified four basic types of training examples, as shown in figure 5. The

exact centre type (figure 5a) refers to the positive object examples which sit exactly the centre of the sweeping window. This type of examples has only a very small number. For example, in each of our training images, we have only 16 such examples out of approximately half a million pixel positions. The *background* type (figure 5d) refers to the positions (x) which do not contain any piece of objects. This type typically has a huge number of examples. The *close to center* type refers to the examples that have the centre of the sweeping window falling down within the bounds of an object (figure 5b). The *include objects* type refers to the examples that contain some pixels of an object but are not considered as the *close to centre* type.

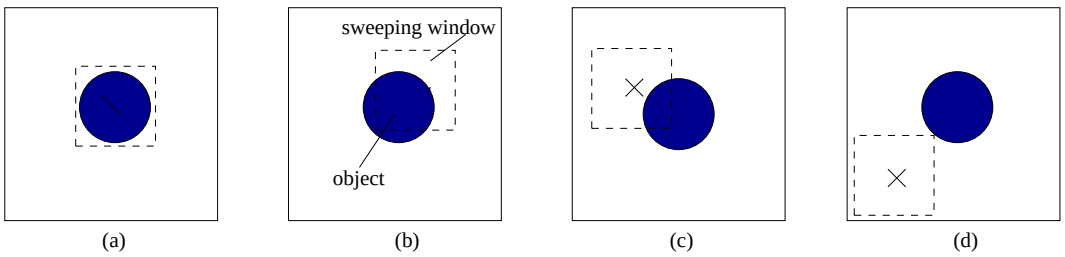


Figure 5: Examples of training data types caused by different input window positions.

5.2 Optimisation of Training Data

For a problem domain, we assume that there is some proportion of these four types which is optimal (or close to optimal) for object detection. From the previous research, we found that the exact centre type is always important for object detection. As the number of examples of this type is very small, we will always use this type of examples in the experiments and assume that the best results can only be achieved using them. In the remainder of investigation, we will vary the proportions among the rest three types to find the optimal combinations.

Based on this idea, if we use C, I and B to refer to percentage of the examples for the three types *close to centre, include objects* and *background*, then we have:

$$C + I + B = 100\%$$

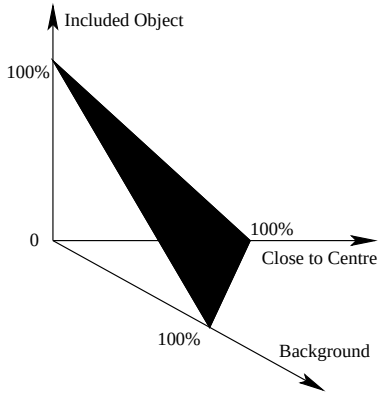
This has the nice feature that it represents only a plane effectively reducing the parameter search space from 3D to 2D, as shown in figure 6 (left). We experimented with 28 separate proportions sampled from the plane in figure 6 (left), as shown in figure 6 (right), where each entry represents value for I for a given C and B. For example, the first two entries in the first row show that, using no background ($B = 0$), we will examine 100% C with 0% I, and 83% C with 17% I type objects, etc.

5.3 Results

For each experiment with a sampled proportion, we did 100 runs. These were made up of 10 different random seeds when extracting the training data from source images, by 10 different random seeds for the GP system. Other parameters are the same as before.

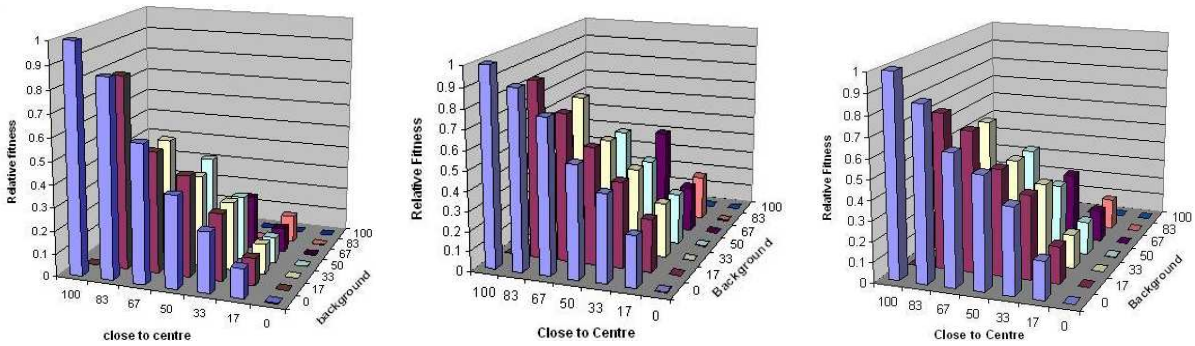
The average results on the *test set* are shown in figure 7. In the figure, the x and y axises are the C and B, and the z is the *relative* fitness for the these problems (1.0 or 100% means the ideal case).

As shown in the figure, for all the three data sets, the value of C, or the percentage of the objects for the *Close to Centre* type played an important role using our new fitness function. The best detection results were achieved with 100% examples for the *close to centre* type and



B\C	I for each C and B (%)						
	100	83	67	50	33	17	0
0	0	17	33	50	67	83	100
17		0	17	33	50	67	83
33			0	17	33	50	67
50				0	17	33	50
67					0	17	33
83						0	17
100							0

Figure 6: Training data proportions set.



(a) (b) (c)

Figure 7: Results of optimisation. (a) Easy; (b) Medium difficulty; (c) Hard.

the worst results were produced when we do not use any example in this type at all. The more object examples used in this type, the best results achieved. However, the *Background* type objects were not critical for these data sets. These examples did not seem to have clear bad or good influence.

These results suggest that, when using the new NLWF fitness function for these object detection, good fitness results can be achieved with only the two types, *Exact Centre* and *Close to Centre*, and most if not all object examples for the other two types *Include Object* and *Background* can be taken out from the training set.

Inspection of this reveals that, this is not only because the first two types of objects might contain the most useful information for object detection, but more importantly, because the new RLWF fitness function is capable of learning well from these two types of examples and can cope well with the goal of finding object centres from large images. This is mainly due to the fact that the RLWF fitness function consider the relative effect of the detected "objects" in different locations.

A further inspection of the use of the old fitness function reveals that the old fitness function must use object examples from all the four types. This is because the old fitness function cannot capture the relative effect information from the objects of the first two types only. This also suggests that the new fitness function is more effective than the old one for object detection.

6 Conclusions

The goal of this paper was to develop a new fitness function for object detection and investigate its influence on optimising the training data. Rather than using a clustering process to determine the number of objects detected by the GP systems, the new fitness function introduced a weight called localisation fitness to represent the goodness of the detected objects and used weighted F-measures. To invetigate the training data with this fitness function, we categorise the training data into four types. This approach is examined and compared to that with the old clustering based fitness function on three coin detection problems of increasing difficulty.

The results suggest that the new fitness function outperforms the old one by producing far fewer false alarms and spending much less training time. Further investigation on the four types of the training object examples suggests that the first two types of objects can be used to produce good detection results and that the new fitness function is effective in optimising the training data for object detection.

In the future, we will apply the new approach to other object detection problems particularly with non-circular objects.

Acknowledgement

We would like to thank Prof Yuejin Ma at Artificial Research Centre and College of Mechanical and Electrical Engineering, Agricultural University of Hebei, China for his providing work environment support and a number of useful discussions.

This work was supported in part by the Marsden Fund at Royal Society of New Zealand under grant No. 05-VUW-017 and University Research Fund 6/9 at Victoria University of Wellington.

References

[1] Gader, P.D., Miramonti, J.R., Won, Y., Coffield, P.: Segmentation free shared weight neural networks for automatic vehicle detection. Neural Networks **8** (1995) 1457–1473

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

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

[4] Waxman, A.M., Seibert, M.C., Gove, A., Fay, D.A., Bernandon, A.M., Lazott, C., Steele, W.R., Cunningham, R.K.: Neural processing of targets in visible, multispectral ir and sar imagery. Neural Networks **8** (1995) 1029–1051

[5] Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications. Morgan Kaufmann Publishers; Heidelburg (1998)

[6] Koza, J.R.: Genetic programming : on the programming of computers by means of natural selection. Cambridge, Mass. : MIT Press, London, England (1992)

[7] Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge, Mass. : MIT Press, London, England (1994)

[8] Song, A., Ciesielski, V., Williams, H.: Texture classifiers generated by genetic programming. In Proceedings of the 2002 Congress on Evolutionary Computation, IEEE Press (2002) 243–248

[9] Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann (1993) 303–309

[10] Zhang, M., Andreae, P., Pritchard, M.: Pixel statistics and false alarm area in genetic programming for object detection. In Applications of Evolutionary Computing, LNCS Vol. 2611, Springer-Verlag (2003) 455–466

[11] Zhang, M., Ciesielski, V., Andreae, P.: A domain independent window-approach to multiclass object detection using genetic programming. EURASIP Journal on Signal Processing, **2003** (2003) 841–859

[12] Smart, W., Zhang, M.: Classification strategies for image classification in genetic programming. In Proceeding of Image and Vision Computing Conference, Palmerston North, New Zealand (2003) 402–407

[13] Howard, D., Roberts, S.C., Brankin, R.: Target detection in SAR imagery by genetic programming. Advances in Engineering Software **30** (1999) 303–311

[14] Bhowan, U.: A domain independent approach to multi-class object detection using genetic programming. BSc Honours research project, School of Mathematical and Computing Sciences, Victoria University of Wellington (2003)