Pixel Statistics and Program Size in Genetic Programming for Object Detection

Mengjie Zhang, Urvesh Bhowan

Technical Report CS-TR-04/3
Feb 2004

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

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

Pixel Statistics and Program Size in Genetic Programming for Object Detection

Mengjie Zhang, Urvesh Bhowan

Technical Report CS-TR-04/3
Feb 2004

Abstract

This paper describes an approach to the use of genetic programming for object detection problems. In this approach, domain independent, local region pixel statistics are used to form three terminal sets. The function set is constructed by the four standard arithmetic operators and a conditional operator. A multi-objective fitness function is constructed based on detection rate, false alarm rate, false alram position and program size. This approach is applied to three object detection problems of increasing difficulty. The results suggest that the concentric circular pixel statistics are more effective than the square features for the these object detection problems. The fitness function with program size is more effective and more efficient for these object detection problems and the evolved genetic programs using this fitness function are much shorter and easier to interpret.

Keywords Genetic programming, pixel statistics, false alarm position, program size, multiclass object detection.

Author Information

Mengjie Zhang is an academic staff member in computer science and Urvesh Bhowan was a postgraduate student in computer science, School of Mathematical and Computing Sciences, Victoria University of Wellington, New Zealand.

Pixel Statistics and Program Size in Genetic Programming for Object Detection

Mengjie Zhang and Urvesh Bhowan

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

Abstract. This paper describes an approach to the use of genetic programming for object detection problems. In this approach, domain independent, local region pixel statistics are used to form three terminal sets. The function set is constructed by the four standard arithmetic operators and a conditional operator. A multi-objective fitness function is constructed based on detection rate, false alarm rate, false alram position and program size. This approach is applied to three object detection problems of increasing difficulty. The results suggest that the concentric circular pixel statistics are more effective than the square features for these object detection problems. The fitness function with program size is more effective and more efficient for these object detection problems and the evolved genetic programs using this fitness function are much shorter and easier to interpret.

1 Introduction

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

There have been a number of reports on the use of genetic programming in object detection and classification [1–8]. Typically, simple image features or high level features based on the whole objects (or the whole sliding windows) are used to form terminals, and the four arithmetic operations form the function set. However, genetic programming with the global pixel statistics for object detection often results in many false alarms. The main reason is that these global features are position and rotation invariant and not effective for object localisation. In this paper, we investigates a number of domain independent terminal sets based on pixel statistics of local regions from the whole object or the whole sliding window.

The main objective of a detection system is to achieve a high detection rate and a low false alarm rate, so that fitness functions in genetic programming for object detection are often based on detection rate and false alarm rate or similar measures [9]. A problem with such fitness functions is that they may not reflect small improvements of evolved genetic programs. A further problem is that the sizes of evolved genetic programs are usually quite large and often have a large amount of redundancy. Both problems could make the genetic search very slow and very hard to find good solutions.

The overall goal of this paper is to develop a domain independent approach to the use of genetic programming for object detection problems. Specifically, we investigate two developments. The first is to explore the use of local region pixel statistics to form the terminal sets and investigate which set is the most effective. The second is to explore the effect on the evolutionary process of an additional measure, *program size*, in the fitness function.

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

2 The Approach

This approach 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, 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.

In this system, we used tree structures to represent genetic programs. The ramped half-and-half method [10] was used for generating the programs in the initial population and for the mutation operator. The proportional selection mechanism and the reproduction, crossover and mutation operators [10] were used in the learning process.

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

2.1 Terminal Sets

For object detection problems, terminals generally correspond to image features. To meet the requirement of domain independence, we use low level pixel statistics, mean and variance of the pixel values, as terminals. These features represent overall brightness/intensity and the contrast of a region of the image. Instead of using global pixel statistics of an entire input image window only, we consider a number of local regions from which pixel statistics will be computed. In this approach, we investigated three sets of terminals based on different local regions.

Terminal Set I — Whole Window and Central Pixel Statistics. This terminal set only consists of four pixel statistics: the mean (F1) and standard deviation (F2) of the whole window, and the mean (F3) and standard deviation (F4) of the central local region, as shown in figure 1 (a). Since the centres of the objects are used to represent the object themselves, we hypothesised the central region is important. The motivation of this set is to investigate whether a very simple set like this can do a good enough job or not.

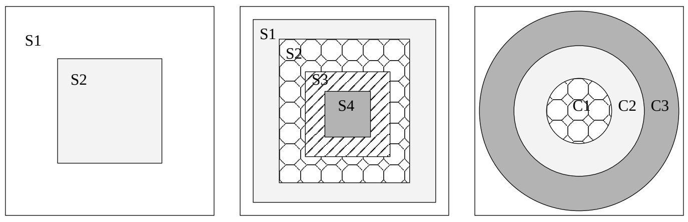


Fig. 1. Terminals. (a) Terminal set I; (b) Terminal set II; (c) Terminal set III.

Terminal Sets II — Local Square Region Pixel Statistics. This terminal set was constructed by the means and standard deviations computed from a series of concentric local square regions centred in the input image window, as shown in figure 1 (b). For each input image window, four such regions were formed and eight features were constructed. The motivation here is to investigate whether this group of local square region features can do a better job than the first terminal set.

Terminal Sets III — Local Circular Region Pixel Statistics. This terminal set was constructed by the means and standard deviations computed from a series of concentric circular regions centred in the input image window, as shown in figure 1 (c). For each input image window, three such regions were formed and six features were constructed. The motivation here is to investigate whether the circular local square region pixel statistics are more effective than the square region features.

2.2 Function Sets

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

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

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

2.3 Object Classification Strategy

The output of a genetic program 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 positive and negative numbers of a genetic program output corresponds to the separation of the objects of interest (of a single class) from the background (non-objects). However, for multiple class object detection problems described here, where two or more classes of objects of interest are involved, the standard genetic programming classification strategy mentioned above cannot be applied.

In this approach, we used a different strategy called *program classification map*, as shown in equation 1, for the multiple class object detection problems [9]. 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.

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

2.4 Fitness Function

The goal of object detection is to achieve a high detection rate and a low false alarm rate. In genetic programming, this typically needs a "multi-objective" fitness function. An example fitness function [9] is:

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

where DR is the Detection Rate (the number of small objects correctly reported by a detection system as a percentage of the total number of actual objects in the images) and FAR is the False Alarm Rate (also called *false alarms per object*, the number of non-objects incorrectly reported as objects by a detection system as a percentage of the total number of actual objects in the images). The parameters W_d, W_f reflect the relative importance between the detection rate and the false alarm rate.

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

A poor program may incorrectly identify a large cluster of locations as an object and a better program may identify a smaller cluster of locations (as shown in figures 2 (b) and (c)). Although the second program is better than the first, it has exactly the same FAR since both programs have two false positives. A fitness function based solely on DR and FAR cannot correctly rank these two programs, which means that the evolutionary process will have difficulty for selecting better programs. To deal with this problem, the False Alarm Position (FAP, the number of false alarm pixels which are not object centres but are incorrectly reported as object centres before clustering) was added to the fitness function [8].

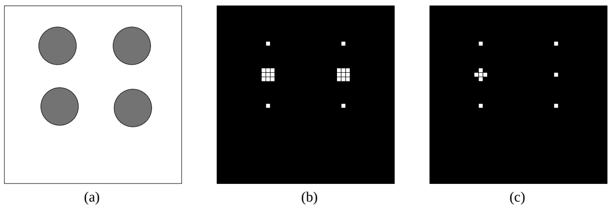


Fig. 2. Sample object detection maps. (a) Original image; (b) Detection map produced by a poor program; (c) Detection map produced by a better program.

Another problem of using this fitness function is that some genetic programs evolved are often very long. When a short program and a long program produce the same detection rate and the same false alarm rate, the GP system will randomly choose one for reproduction, mutation or crossover during the evolutionary process. If the long programs are selected, the evolution for the rest of the learning process will be slow. More importantly, the good building blocks in these long programs will have a much greater chance to be destroyed than in the short programs [10], which could lead to poor solutions by the evolutionary process. This is mainly because this fitness function does not include any hints about the size of programs.

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

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

1. Apply the program as a moving $n \times n$ window template (n is the size of the input image window) to each of the training images and obtain the output value of the program at each possible window position. Label each window position with the 'detected' object according to the object classification strategy. Call this data structure a detection map.
2. Find the centres of *objects of interest only* by the clustering algorithm:
   - 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. Skip pixels in $n/2 \times n/2$ square to right and below this point.
3. Match these detected objects with the known locations of each of the desired true objects and their classes.
4. Calculate the detection rate DR, the false alarm rate FAR, and the false alarm position FAP of the evolved program.
5. Count the size of the program by adding the number of terminals and the number of functions in the program.
6. Compute the fitness of the program according to equation 3.

$$fitness = K1 \cdot (1 - DR) + K2 \cdot FAR + K3 \cdot FAP + K4 \cdot ProgSize \quad (3)$$

where $K1, K2, K3$, and $K4$ are constant weights which reflect the relative importance between detection rate, false alarm rate, false alarm area, and program size.

2.5 Parameters and Termination Criteria

The important parameter values used in this approach are shown in table 1.

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

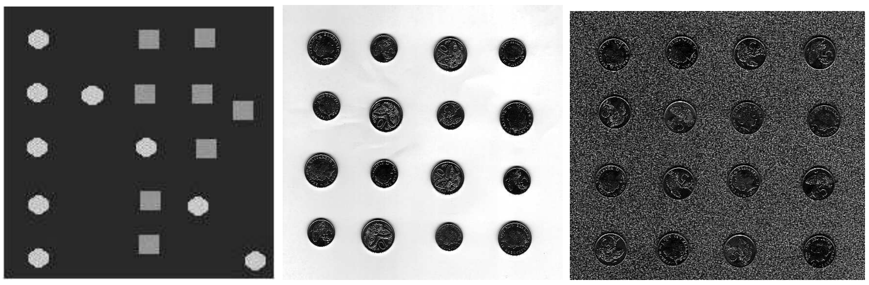
- The detection problem has been solved on the training set, that is, all objects in each class of interest in the training set have been correctly detected with no false alarms.
- The number of generations reaches the pre-defined number, *max-generations*.

3 Image Data Sets

We used three different data sets in the experiments. Example images are given in figure 3. These data sets provide object detection problems of increasing difficulty. Data set 1 (Shape) was generated to give well defined objects against a uniform background. The pixels of the objects were generated using a Gaussian

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

Parameter Kind	Parameter Name	Shape	Coin1	Coin2
	population-size	800	1000	1600
Search	initial-max-depth	2	2	5
	max-depth	6	7	8
Parameters	max-generations	50	150	200
	input-size	20×20	72×72	62×62
	reproduction-rate	2%	2%	2%
Genetic	cross-rate	72%	70%	70%
Parameters	mutation-rate	28%	28%	28%
	T	100	80	80
Fitness	K1	5000	5000	5000
	K2	100	100	100
Parameters	K3	10	10	10
	K4	1	1	1



No. of images: 10
Object size:18×18

(Easy)

No. of images: 20
Object size:70× 70

(Coin1)

No. of images: 20
Object size:60×60

(Coin2)

Fig. 3. Object Detection Problems

generator with different means and variances for different classes. There are two classes of small objects of interest in this database: circles and squares. Data set 2 (Coin1) was intended to be somewhat harder and consists of scanned images of New Zealand coins. There are two object classes of interest: the 5 cent coins and 10 cent coins. Each class has either tail side up or head side up and accordingly has a greater variance than data set 1. The objects in each class have a similar size but are located at arbitrary positions and with different rotations. Data set 3 (Coin2) also contains two object classes of interest, but the detection task is significantly more difficult. The task is detecting the head side and the tail side of New Zealand 5 cents from a highly cluttered background. This detection task is very difficult and even human eyes could not detect those objects perfectly.

In the experiments, we used one, three, and five images as the training set and used five, ten and ten images as the test set for the *easy, coin1*, and *coin2* data sets, respectively.

4 Results

This section describes the detection results. For all cases, the experiments were repeated 10 times and the average results on the *test set* were presented.

4.1 Data Set I — Shape

The detection results using the three terminal sets and the two fitness functions are shown in table 2. The GP system using each of the three terminal sets and each of the two fitness function produced ideal result, that is, all the objects of interest were correctly detected from the large images in the test set without producing any false alarms. This reflects the fact that the detection problem in this data set is relatively easy. However, the genetic program detectors evolved by the evolutionary learning process were quite different if different fitness functions were used. We now analyse the evolved genetic programs.

Table 2. Object detection results for the shape data set.

Shape Data Set		Object Classes	
		circles	squares
Best Detection Rate(%)		100	100
False Alarm Rate (%)	Term set I	0	0
	Term set II	0	0
	Term set III	0	0

Genetic Program vs Fitness Function. To analyse the effect of program size in fitness function, we use the second terminal set as an example for this discussion. Other two terminal sets have a similar pattern.

To make a fair comparison, we did 20 experimental runs by using each of the two fitness functions and chose the best 40 evolved genetic program detectors trained using each fitness function. The average program size and a typical evolved genetic program trained with each fitness function is shown in figure 4.

```
                             Fitness function I

Average program size: 62
Sample Program:
(/ (+ (* (/ (/F8 T) (+ T F1)) (+ (+ T F2) (if F7 T F8)))
      (if (- (* T F4) F5)  (- (+ F6 T) (* T F7)) (/ (- T F8) (- F1 T)))
   )
   (+ (if (/ (/ F2 T) (* T F3))  (+ (if T F4 F7) (- T F5))
          (* (/ F1 F6) (+ F7 T))) (* (/ T (+ T T)) (/ T (/ T F4))))
)
                             Fitness function II

Average program size: 24
Sample program:
  (/ (if (/ (- F7 T) F7) F8 (* (- F7 F3) F2)) (/ F7 F7))
```

Fig. 4. Program size, sample programs and fitness function

As can be seen from figure 4, the size of evolved genetic program detectors obtained using the second fitness function is much shorter than using the first. This not only had the training time greatly reduced to find a good program detector, but also could make the program detector much easier to interpret. For example, the program detector obtained using the second fitness function in figure 4 can be simplified as follows:

```
(if (- F7 T)  F5  (* (- F7 F3) F2))
```

where F3, F5 and F7 are the means of the central regions S2, S3, and S4 (see figure 1b) of a detected object window, respectively, F2 is the standard deviation of the largest region, and T is a predefend threshold. This program can be translated as the following rule:

```
if (F7 > T) then
  ProgOut = F5;
else
  ProgOut = (F7 - F3) * F2;
```

If the sweeping window is over the background only, F7 would be smaller than the threshold (100 here), the program would execute the "else" part. Since F7 is equal to F3 in this case, the program output will be zero. According to the classification strategy — object classification map, this case would be correctly classified as *background*. If the input window contains a portion of an object of interest and some background, F7 would be smaller than F3, which resulted in a negative program output, corresponding to class *background*. If F7 is greater than the threshold T, then the input window must contain an object of interest, either for *class1* or for *class2*, depending the value of F5.

While this program detector can be relatively easily interpreted and understood, the programs obtained using the first fitness function are generally hard to interpret due to the length of the program and lack of good building blocks.

4.2 Data Set II — Coin1

Since the second fitness function was proved to be better than the first fitness function, we only applied the second fitness function to this and the next data sets. The results for detecting 5 cent coins and 10 cent coins from a relatively uniform background are shown in table 3.

Table 3. Object detection results for the second data set.

Coin1 Data Set		Object Classes		
		coin10	coin05	overall
Best Detection Rate(%)		100	100	100
False Alarm Rate (%)	Term set I	0	20.83	10.42
	Term set II	0	12.5	6.25
	Term set III	0	0	0

According to table 3, all terminal sets correctly detected all the objects of interest for each class. The false alarm rates produced by the three sets were, however, quite different. While terminal set III gave the ideal performance, terminal set II produced fewer false alarms than set I, indicating only four pixel statistics from the whole window and a single central region were not sufficient for the detection problems in this data set. In addition, terminal sets I and II took much longer training time than set III. This suggests that the detection problems in this data set are more difficult than the first data set, and the terminal set III is the best suited to these detection problems.

4.3 Data Set III — Coin2

As can be seen from table 4, none of the three terminal sets gave the ideal results, reflecting the difficulty of the object detection problems in this data set. While all the three terminal sets correctly detected all objects of interest in all images, they produced different false alarm rates. The first terminal set resulted in 68.5% false alarm rate. In particular, it produced 100% false alarm rate for detecting the tails. This indicates that four features only, two from the whole window and two from the local central region of the window, are not sufficient at all for this difficult detection problem. The result was improved to 50% by using more local square region features in terminal set II. The third terminal set, with the local circular region features, produced the best results. The overall false alarm rate was decreased to 39.28%, which suggests that concentric local circular region pixel statistics perform better than local square region features. This is probably because the local circular regions captured more knowledge from the coins.

Table 4. Object detection results for the third data set.

Coin2 Data Set		Object Classes		
		tails	heads	overall
Best Detection Rate(%)		100	100	100
False Alarm Rate (%)	Term set I	100	30.7	68.5
	Term set II	87.5	12.5	50
	Term set III	55.35	23.21	39.28

It is interesting to note that in each case, the false alarm rate for the tails was always higher than for heads. After a careful check and analysis from the object centres detected by the system, it was observed that all false alarms for the tails were generated along the borders of some objects of the heads. None of the centres of class *heads* were reported as false alarms for class *tails* — it was only the input windows with half of some objects in class heads and a bit more than half window of background. Figure 5 (b) shows such an example, while (a) and (c) are correctly detected as class *background* and class *heads*.

We found three major possible reasons for producing those false alarms. The first is that the classification strategy used in this approach was not powerful enough. The second is that the fitness parameters particularly K2 and K3 were

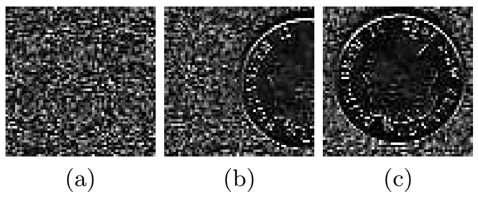


Fig. 5. Example detected objects. (a) class *background*; (b) false alarm for *tails*; (c) class *heads*.

too small compared with K1. The third is that the terminals and functions were not sufficient and powerful. In the future, we will further investigate these factors.

5 Conclusions

The goal of this paper was to investigate the effectiveness of different local region pixel statistics and effect of the program size in fitness functions for object detection using genetic programming. The approach was tested on three object detection problems of increasing difficulty and achieved good results.

We developed three different terminal sets based on domain independent, statistical image features. Our results suggest that, in genetic programming for object detection problems, input terminals should include sufficient local region pixel statistics, and that the pixel statistics computed from local circular regions are more effective than square regions for these object detection problems.

We modified the fitness function by including two measures called false alarm position and program size. The inclusion of false alarm position made the fitness function smoother to reflect both large and small improvements of genetic programs. The inclusion of program size in the fitness function resulted in shorter and better genetic program detectors. This not only reduced the search space and accordingly saved computation time, but also made the genetic program much easier to interpret and accordingly improved the comprehensibility of the evolved programs.

For future work, we will investigate whether the performance on the highly cluttered coin data sets can be improved if more features are added to the terminal set, if the fitness parameters K2 and K3 are increased, and if a better classification strategy is applied. We will also investigate whether the approach can perform well on more difficult object detection problems and compare this approach with other long term established methods such as neural networks and decision trees.

References

1. 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.

2. 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.
3. Cristopher T. M. Graae, Peter Nordin, and Mats Nordahl. Stereoscopic vision for a humanoid robot using genetic programming. In Stefano Cagnoni, Riccardo Poli, George D. Smith, David Corne, Martin Oates, Emma Hart, Pier Luca Lanzi, Egbert Jan Willem, Yun Li, Ben Paechter, and Terence C. Fogarty, editors, *Real-World Applications of Evolutionary Computing*, volume 1803 of *LNCS*, pages 12–21, Edinburgh, 17 April 2000. Springer-Verlag.
4. 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.
5. F. Lindblad, P. Nordin, and K. Wolff. Evolving 3d model interpretation of images using graphics hardware. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation, CEC2002*, Honolulu, Hawaii, 2002.
6. 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.
7. Mengjie Zhang and Victor Ciesielski. Genetic programming for multiple class object detection. In Norman Foo, editor, *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence (AI'99)*, pages 180–192, Sydney, Australia, December 1999. Springer-Verlag Berlin Heidelberg. Lecture Notes in Artificial Intelligence (LNAI Volume 1747).
8. Mengjie Zhang, Peter Andreae, and Mark Pritchard. Pixel statistics and false alarm area in genetic programming for object detection. In Stefano Cagnoni, editor, *Applications of Evolutionary Computing, Lecture Notes in Computer Science, LNCS Vol. 2611*, pages 455–466. Springer-Verlag, 2003.
9. Mengjie Zhang, Victor Ciesielski, and Peter Andreae. A domain independent window-approach to multiclass object detection using genetic programming. *EURIASP Journal on Signal Processing, Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis*, 2003(8):841–859, 2003.
10. 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.