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

School of Mathematical and Computing Sciences
Computer Science

Pixel Statistics and False Alarm Area in Genetic Programming for Object Detection

Mengjie Zhang, Peter Andreae, Mark Pritchard

Technical Report CS-TR-02/16
November 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

Pixel Statistics and False Alarm Area in Genetic Programming for Object Detection

Mengjie Zhang, Peter Andreae, Mark Pritchard

Technical Report CS-TR-02/16
November 2002

Abstract

This paper describes a domain independent approach to the use of genetic programming for object detection problems. Rather than using raw pixels or high level domain specific features, this approach uses domain independent statistical features as terminals in genetic programming. Besides position invariant statistics such as *mean* and *standard deviation*, this approach also uses position dependent pixel statistics such as *moment*s and local region statistics as terminals. Based on an existing fitness function which uses linear combination of detection rate and false alarm rate, we introduce a new measure called "false alarm area" to the fitness function. In addition to the standard arithmetic operators, this approach also uses a conditional operator *if* in the function set. This approach is tested on two object detection problems. The experiments suggest that position dependent pixel statistics computed from local (central) regions and nonlinear condition functions are effective to object detection problems. Fitness functions with false alarm area can reflect the smoothness of evolved genetic programs. This approach works well for the detecting small regular multiple class objects on a relatively uncluttered background.

Keywords Evolutionary computation, object classification, target recognition, machine learning.

Author Information

Mengjie Zhang and Peter Andreae are members of the academic staff in computer science at Victoria University of Wellington. Mark Pritchard is a graduate student in computer science at Victoria University of Wellington.

Pixel Statistics and False Alarm Area in Genetic Programming for Object Detection

Mengjie Zhang, Peter Andreae, Mark Pritchard

School of Mathematical and Computing Sciences,
Victoria University of Wellington, Wellington, New Zealand
{mengjie,pondy,markp}@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/~mengjie,pondy

Abstract. This paper describes a domain independent approach to the use of genetic programming for object detection problems. Rather than using raw pixels or high level domain specific features, this approach uses domain independent statistical features as terminals in genetic programming. Besides position invariant statistics such as *mean* and *standard deviation*, this approach also uses position dependent pixel statistics such as *moments* and local region statistics as terminals. Based on an existing fitness function which uses linear combination of detection rate and false alarm rate, we introduce a new measure called "false alarm area" to the fitness function. In addition to the standard arithmetic operators, this approach also uses a conditional operator *if* in the function set. This approach is tested on two object detection problems. The experiments suggest that position dependent pixel statistics computed from local (central) regions and nonlinear condition functions are effective to object detection problems. Fitness functions with false alarm area can reflect the smoothness of evolved genetic programs. This approach works well for the detecting small regular multiple class objects on a relatively uncluttered background.

Keywords: Evolutionary computation, object classification, target recognition, machine learning.

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 objects of interest, find all images which contain this object and its location(s)". Figure 5 (page 8) shows examples of problems of this kind. In the problem illustrated by figure 5 (b), we want to distinguish and find the centers of all of the New Zealand 5 cent and 10 cent coins. Examples of other problems of this kind include target detection problems [2, 4, 8] 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 [2, 4], our objective is to detect objects from a number of classes.

Domain independence means that the same method will work unchanged on any problem, or at least on some range of problems. This is very difficult to achieve at the current state of the art in computer vision because most systems require careful analysis of the objects of interest and a determination of which features are likely to be useful for the detection task. Programs for extracting these features must then be coded or found in some feature library. Each new detection/vison system must be hand-crafted in this way. Our approach is to use easily computed pixel statistics such as mean and variance of the pixels in a sub-image and to evolve the programs needed for object detection.

There have been a number of reports on the use of genetic programming in object detection and classification [1, 3, 5–9]. Typically, simple features such mean and standard deviation or even high level features are used to form terminals, and the four arithmetic operations form the function set. However, genetic programming with pixel statistics of only mean and standard deviation for object detection often results in many false alarms. The main reason is that mean and standard deviation are position and rotation invariant and not effective for object localisation.

The function set consisting of the four standard arithmetic operations is simple, domain independent and easy to use. However, it is not effective at expressing conditional programs — programs with a different "shape" in different parts of feature space. The programs constructed with these functions are also often difficult to understand and interpret.

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 [10]. A problem with such fitness functions is that they may not reflect small improvements of evolved genetic programs, which makes the genetic search much harder.

1.1 Goals

This paper investigates three new developments in domain independent object detection using genetic programming.

- The first is to explore the use of position dependent pixel statistics to improve the object detection performance.
- The second is to investigate the effect of adding a conditional *if* operator to the function set on the on genetic search and on the comprehensibility of the constructed programs.
- The third is to explore the effect on the evolutionary process of an additional measure in the fitness function.

2 The Approach

2.1 Overview of the Approach

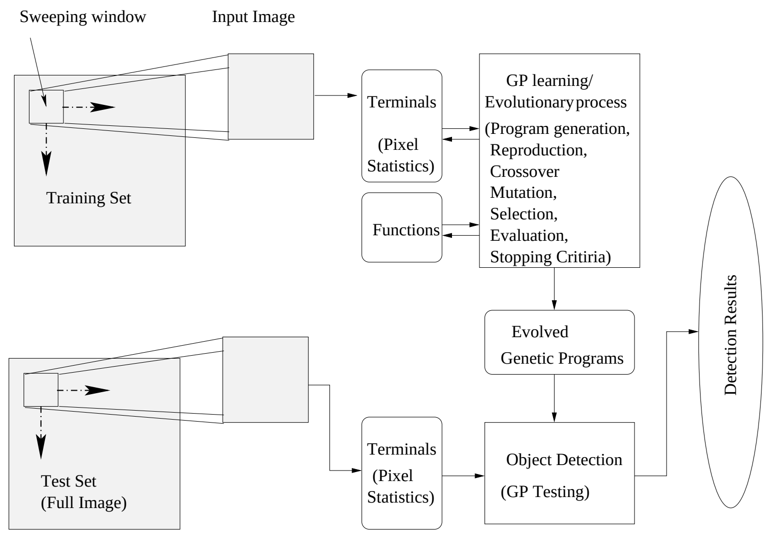


Fig. 1. An overview of the GP approach for multiple class object detection.

Figure 1 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, 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 use tree-structured programs 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. The proportional selection mechanism and the reproduction, crossover and mutation operators 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.

2.2 Terminals

For object detection problems, terminals generally correspond to image features. The simplest statistics (mean and variance of the pixel values) are position and rotation invariant statistics, representing overall brightness/intensity and the contrast of a region of the image. We considered two approaches to introducing position dependence. The first was to use statistics that depended on the positions as well as the values of the pixels. The second was to use position invariant pixel statistics on smaller, local regions of the sliding window. We investigated three terminal sets.

Terminal Set I — Whole Window Pixel Statistics. The first terminal set consisted of the *mean, variance, first order moment, second order moment* of the whole window, and a constant threshold value T. The threshold T is set by the user and is specified in section 2.4. The moments are position dependent but rotation invariant statistics that measure how the high valued pixel are distributed away from the center of the window. The first order moment was the sum of the pixel values weighted by their (Euclidean) distances from the center of the sliding window, and the second order moment was the variance of these values:

$$moment_1 = \frac{1}{n}\sum_{i=1}^{n} f(x_i, y_i) \cdot d(x_i, y_i) \tag{1}$$

$$moment_2 = \frac{1}{n}\sum_{i=1}^{n} (f(x_i, y_i) \cdot d(x_i, y_i) - moment1)^2 \tag{2}$$

Terminal Sets II and III — Local Region Pixel Statistics. For terminal sets II and III, we broke the sliding window into small regions and computed just the mean of the pixel values in each region. We also included the mean of the whole window and the threshold T in each set. For terminal set II, the regions consisted of the four quadrants of the sliding window; for terminal set III, we also included the central square, as shown in figure 2.

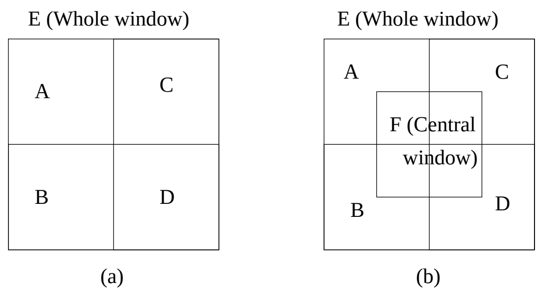


Fig. 2. Local region terminals. (a) Terminal set II; (b) Terminal set III.

2.3 Functions

In the function set, the four standard arithmetic and a conditional operation was 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.

2.4 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 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 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 used a different strategy called *program classification map*, as shown in figure 3, for the multiple class object detection problems [10]. Based on the output of an evolved genetic program, this map can identify which class of the object located in the current input field belongs to. In this map, m refers to the number of object classes of interest, v is the output value of the evolved program and T is a constant defined by the user, which plays a role of a threshold.

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

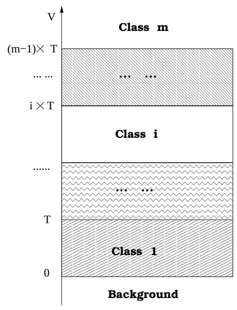


Fig. 3. Mapping of program output to an object classification.

2.5 Fitness Function

Fitness functions used for object detection. 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 [10] is:

$$fitness(DR, FAR) = W_d * (1 - DR) + W_f * FAR \tag{3}$$

where DR is the Detection Rate (fraction of objects correctly identified and localised) and FAR is the False Alarm Rate (the number of incorrectly identified or localised objects as a fraction of the number of true objects). The parameters W_d, W_f reflect the relative importance between detection rate and false alarm rate. Note that the fitness is zero for the ideal case.

Although such a fitness function accurately reflects the performance measure of an object detection system, it is not very smooth. In particular, small improvements in an evolved genetic program performance 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 a 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. A modified program may identify a smaller cluster of locations (as shown in figures 4 (b) and (c)). Although the second program is better than the first, and is closer to the ideal program, it has exactly the same FAR since both programs have one false positive. A fitness function based solely on DR and FAR cannot correctly rank these two programs, which means that the evolutionary process will have difficulty selecting better programs.

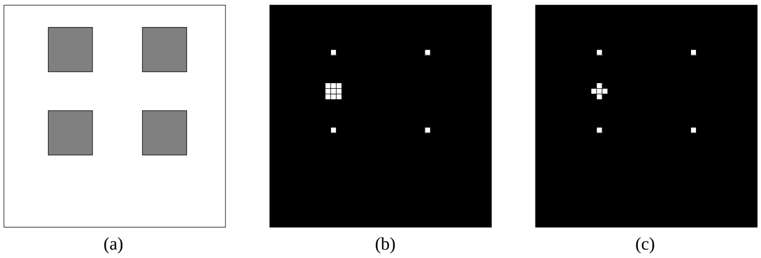


Fig. 4. Sample object detection maps. (a) Original image; (b) Object detection map produced by program 1; (c) Object detection map produced by program 2.

Fitness Function in This Approach To smooth the fitness function so that small improvement in genetic programs could reflect small improvement in fitness measure while large improvement in genetic programs could reflect large improvement in fitness, we introduced a new measure, *false alarm area*, into the above fitness function. The new 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 described in figure 3. 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 $n/2$ 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, the false alarm rate FAR, and the false alarm area FAA of the evolved program. The 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 FAR, also called false alarms per object, 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). The FAA is the number of false alarm pixels which are not object centres but are incorrectly reported as object centres before (without) clustering.
5. Compute the fitness of the program according to equation 4.

$$fitness(FAR, DR) = A \cdot (1 - DR) + B \cdot FAA + C \cdot FAR \tag{4}$$

   where A, B, C are constant weights which reflect the relative importance of detection rate versus false alarm rate versus false alarm area.

2.6 Parameters and Termination Criteria

The 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:

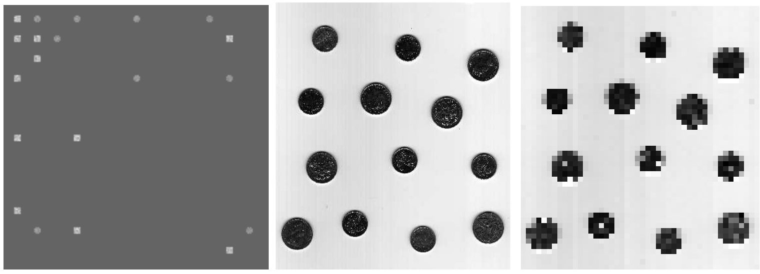
- The detection problem has been solved on the training set, that is, all objects in each class of interest in the training set have been correctly detected with no false alarms. In this case, the fitness of the best individual program is zero.
- The number of generations reaches the pre-defined number,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

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

Parameter Kinds	Parameter Names	Easy Images	Coin Images
Search Parameters	population-size	300	500
	initial-max-depth	4	5
	max-depth	8	12
	max-generations	150	200
	input-size	16×16	20×20
Genetic Parameters	reproduction-rate	10%	5%
	cross-rate	65%	70%
	mutation-rate	25%	25%
	cross-term	15%	15%
	cross-func	85%	85%
Fitness Parameters	T	80	100
	A	50	50
	B	1000	1000
	B	100	200
	tolerance (pixels)	2	2

3 Image Databases

We used two different databases in the experiments. Example images are given in figure 5. 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 two classes of small objects of interest in this database: circles and squares. Database 2 (Coins) was intended to be somewhat harder and consists of scanned images of New Zealand coins. There are two object classes of interest: the 5 cent coins and 10 cent coins. The objects in each class have a similar size but are located at arbitrary positions and with different rotations. Although the task is simple for humans with the original scanned image (Fig 5 (b)), we used a low resolution version (Fig 5 (c))that reduced the computation time, but resulted in a significantly more difficult task.



Easy (a) | Coin (b) | Coin (Low Resolution) (c)

Fig. 5. Object Detection Problems

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 Easy Images

Table 2. Object detection results for the easy images.

Easy Images			Object Classes	
			circles	squares
Best Detection Rate(%)			100	100
False Alarm Rate (%)	GP1	Term set I	240	0
	GP2	Term set II	0	0
	GP3	Term set III	0	0

Table 2 shows the best results of the GP approach with the three different terminal sets for the easy images. Detecting squares was relatively straight forward, where the GP with all the terminal sets achieved a 100% detection rate with no false alarms. For circles, GP2 and GP3 achieved ideal results, but GP1 resulted in 240% false alarm rate at a detection rate of 100%. GP2 took 147 generations to achieve the ideal result but GP3 only used 87 generations. This suggests that the position invariant statistics on local regions are more effective than the position dependent statistics on the whole window. The addition of the central local region is also helpful for object localisation.

Sample object detection maps which gives an intuitive view of the detection results using GP2 is shown in figure 6. In figures 6 (b) and (c), a white point (pixel) represents a match between a detected object centre and a desired object centre for circles and squares. This figure clearly shows that GP2 achieved ideal performances for detecting objects of interest in these large images.

Original (a) — Class circle (b) — class square (c)

Fig. 6. Sample Object detection map for the easy images.

4.2 Coin Images

Table 3. Object detection results for the coin images.

Coin Images			Object Classes	
			coin005	coin010
Best Detection Rate(%)			100	100
False Alarm Rate (%)	GP1	Term set I	440	210
	GP2	Term set II	34	0
	GP3	Term set III	0	0

Experiments with the coin images gave similar results to the easy images, as shown in table 3. GP1 resulted in a large number of false alarms for both classes. While GP2 greatly improved the performance, it still produced some false alarms for detecting the 5 cent coins. GP3 achieved the best (ideal) performance. These results confirmed that the moment statistics on the whole input window did not provide sufficient position dependence for this object detection problem. Statistics on local regions, especially if the central region was included, could do a better job. The results also demonstrated that the augmented fitness function worked well for the two object detection problems.

4.3 Comprehensibility

To check the role of the conditional *if* operator played in the evolutionary process, an evolved genetic program produced by GP2 for the easy images is shown in figure 7.[1]

```
(if (if (- (+ T T) F4) (+ (- F3 F5) (- (+ T T) F4)) (+ F1 F5))
    (+ (/ F2 (- F1 (+ F4 F5))) F3)
    (- (* F3 F4) T)
)
```

Fig. 7. A sample genetic program.

This program is considerably simpler than the programs generated in previous experiments that did not include an *if* operator. The program contains of two parts — two different expressions that will be used in different parts of the feature space. The first line represents the (fairly complex) condition that segments the two parts of the space. Notice the part of the condition that is equivalent to `(F4 < 2T)`. Although it is still hard to interpret the exact meaning of this program, it does show that this program could approximate the classification rule — we have two classes of interest and the thresholds T and 2T occurred in this program which made the program have three ranges (for two classes of interest plus the background).

[1] `F1, F2, F3, F4, F5` represent the mean of local regions `A, B, C, D` and the whole window (`E`) on figure 2, respectively.Note that this program has had some manual simplification to remove the large amount of redundancy in the original program.

5 Conclusions

The goal of this paper was to investigate the three new developments in multi-class, domain independent object detection using genetic programming. Three terminal sets based on domain independent, statistical image features were developed. The function set was augmented by the use of a newly defined conditional function. The fitness function was improved by introducing a new measure called false alarm area. This approach was tested on two object detection problems an achieved good results.

During the development and experiments, the following characteristics can be concluded:

- The position dependent pixel statistics on the four quadrants of a sliding window could do a better job than those only on the whole window. In particular, statistics containing a central local region seem more effective for these detection problems than without. This suggests that, in genetic programming for object detection problems, input terminals should include local region pixel statistics not just the whole input sliding window.
- The evolved genetic programs with an *if* function are more comprehensible and more understandable than without. In addition, this kinds of evolved genetic programs could simulate a different "shape" in different parts of feature space. Accordingly, for multiclass object detection problems, a function set needs to include an *if* function.
- For these object detection problems, it is good to use the false alarm area measure in the fitness function, which could effectively reflect both large and small improvements of genetic programs. The fitness function should be an accurate, smooth measure of the evolved genetic programs.

This approach has a number of limitations, which need to be considered and improved in the future:

- The programs generated by the evolutionary process contain a large amount of redundancy. Simplification of these programs should be considered during the evolutionary process so that the search space can be greatly reduced.
- The training process is relatively long due to a large number of input images (examples) applied in the window moving procedure. A better method of producing representative training examples should be considered.
- While this approach did very well on the two detection problems here, it should be tested on more difficult object detection problems in the future.

References

1. 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.

2. 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.
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 Richard Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311, 1999.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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).
10. Mengjie Zhang, Victor Ciesielski, and Peter Andreae. A domain independent approach to multiclass object detection using genetic programming. Technical Report CS-TR-02-4, Victoria University of Wellington, April 2002.