

Understanding Evolved Genetic Programs for a Real World Object Detection Problem

Victor Ciesielski¹, Andrew Innes¹, Sabu John², and John Mamutil³

¹ School of Computer Science and Information Technology,
RMIT University, GPO Box 2476V, Melbourne, Vic Australia 3001
{vc, a_innes}@cs.rmit.edu.au

² School of Aerospace, Mechanical and Manufacturing Engineering,
RMIT University

³ Braces Pty Ltd, 404 Windsor Road, NSW 2153, Australia

Abstract. We describe an approach to understanding evolved programs for a real world object detection problem, that of finding orthodontic landmarks in cranio-facial X-Rays. The approach involves modifying the fitness function to encourage the evolution of small programs, limiting the function set to a minimal number of operators and limiting the number of terminals (features). When this was done for two landmarks, an easy one and a difficult one, the evolved programs implemented a linear function of the features. Analysis of these linear functions revealed that underlying regularities were being captured and that successful evolutionary runs usually terminated with the best programs implementing one of a small number of underlying algorithms. Analysis of these algorithms revealed that they are a realistic solution to the object detection problem, given the features and operators available.

1 Introduction

Genetic programming has been used with considerable success for a wide range of problems, both toy and real world. In the area of image processing there have been a number of successful applications to difficult problems. One of the drawbacks of genetic programming solutions is that the evolved programs are very hard to understand and can be very big. This makes it hard to sell genetic programming solutions as many engineers and other professionals, particularly in the medical area, are very unhappy with black box solutions. They want to understand the operation of any solution to better appreciate its limitations and generality.

There has been very little prior work on the problem of understanding evolved program trees. Most work is focused on finding acceptable solutions to problems without any major concern for their understandability. However, [1] describes ‘trait mining’, a method of finding sub-trees, or traits, in the evolved programs. Traits are associated with high fitness and provide some insight into the evolved solutions. The basic idea is that even if the whole program cannot be understood,

being able to understand some of the subtrees means that the evolved program is not a black box. [2] describes the evolution of classification rules with genetic programming. The rules are intended to be understandable to humans. This is achieved by evolving the rules in a predetermined if-then format, rather than as full genetic programs. [3] describes an approach to finding the underlying regularities in evolved programs which use pixel inputs for texture discrimination. A strategy of limiting the function set and penalising large programs was used. Analysis of the resulting programs revealed that a number of masks of pixel positions were being used to discriminate textures. Systematic regularities were consistently being discovered in the different runs, arbitrary positions of pixels were not being used.

We have recently had considerable success in using genetic programming to evolve object detection programs for finding orthodontic landmarks in cranio-facial X-Rays [4]. Orthodontists are not comfortable with genetic programming and there is a need to explain the evolved programs in ways that are credible to them. The aim of the work presented in this paper is to determine whether we can explain how evolved programs work for the object detection problem. In particular we are interested in:

1. Can we develop a methodology for analysing evolved programs?
2. Are regularities being discovered? If so what are they?
3. Can we express, in some reasonable way, the underlying algorithms in the evolved programs.

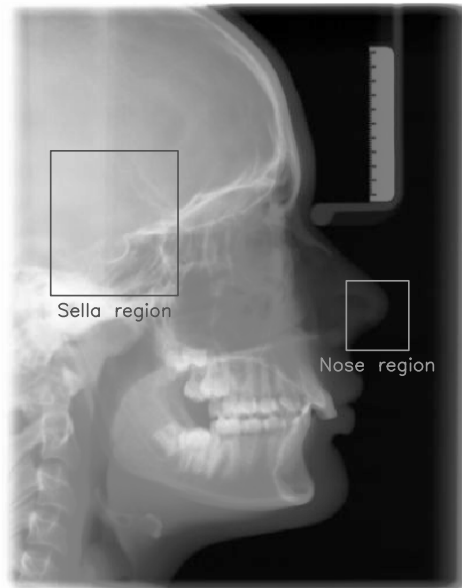


Fig. 1. A digital cephalogram depicting two regions containing the mid nose and sella landmarks

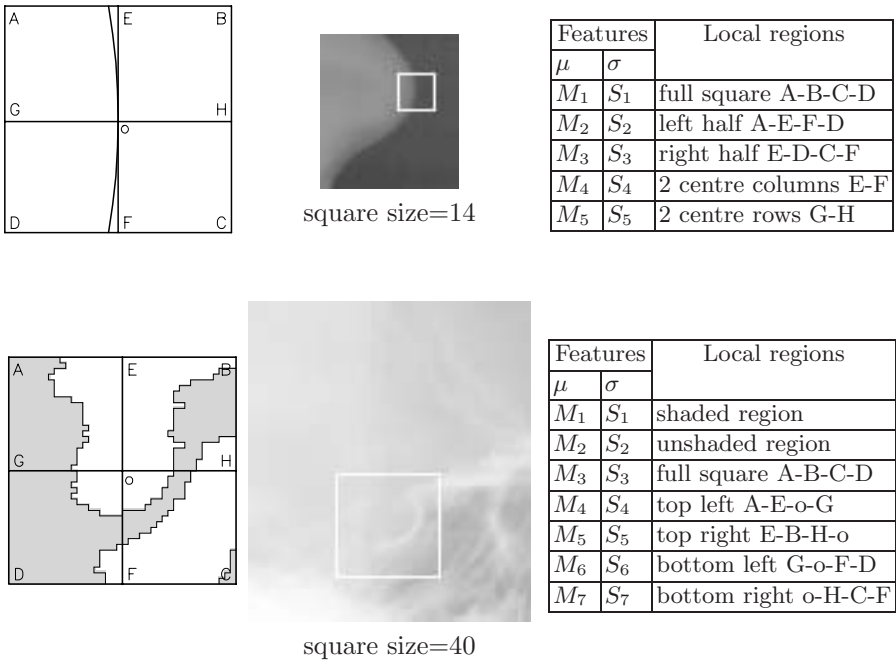


Fig. 2. The diagrams in the left column depict the feature maps used for the nose and sella landmarks. The features consist of the mean and standard deviation calculated for each shape from grey level intensities. The corresponding pictures in the middle column depict the size of the feature map (shown as the white square) relative to the image

2 The Object Detection Problem

The object detection problem is to locate, to within 2mm, a number of key landmarks used by orthodontists in treatment planning. Full details of this problem can be found in [4, 5, 6]. For this paper we focus on just two landmarks, an easy one, the tip of the nose and a very difficult one, the sella. These landmarks are shown in figures 1 and 2. We consider the finding of each landmark as an independent object detection problem. Using prior knowledge of facial geometry we identify regions in the picture where the landmark should be with respect to the edge of the ruler which is easy to find by classical image processing methods. Each of these individual problems is solved independently using the method described in the next section. It might appear that the problem of finding the nose tip would be very straight forward, but this is not the case due to the wide variation between humans in nose shapes and various noise effects on the X-Rays.

3 Finding Object Detection Programs by Genetic Programming

The basic method is to take an input window of pixels and slide the window over all positions in the region to find the location of the landmark [4]. This is similar to the template matching approach in classical vision except that the template is implemented as a genetic program. The output of the program is interpreted as the presence or absence of the landmark at the pixel position at the centre of the window. Since we know that there can be only one landmark in the region and that it must occur, we use the position with the largest output as the predicted position of the object.

The evolved programs use a feature set of pixel statistics of regions, or shapes, customised for each landmark. The features are obtained from a square input window centred on the landmark and large enough to contain key landmark characteristics, but not large enough to introduce unnecessary clutter, as shown in figure 2. For the nose point the input window has been partitioned manually into the shapes shown and the means and standard deviations of the pixels in each shape are the features. For the sella point the input window has been segmented into shapes by the use of pulse coded neural networks as described in [7]. The feature values are used as the terminals and the function set is $\{+, -, \times, \%\}$ (% denotes protected division).

A data base of images marked up by an orthodontist is used in training. To evaluate fitness during training, each program is applied, in moving window fashion, across each of the training images and the detection rate is computed. The detection rate is used as the fitness measure.

The method was tested on a number of landmark points, ranging from relatively easy to very difficult. Detection performance on the easier points was excellent and the performance on the difficult points was quite good [4].

4 Methodology for Analysing Evolved Programs

The evolved programs from the system described in the previous section are large and difficult to understand. To evolve smaller programs which have a higher likelihood of being understood we repeat the evolutionary process with:

1. A size penalty for large programs. The fitness function is

$$fitness = (1 - Dr) \times 100 + \frac{Program\ Size}{511} \times \frac{1}{10}, \quad (1)$$

where Dr is the detection rate.

Program Size is the number of nodes in the program and there are 511 nodes in a full tree of depth 9, which is the max depth used in the runs. The size component represents constant parsimony pressure. The weightings of the two terms ensures that accuracy is the primary objective and size is secondary.

2. A reduced function set. For the nose point we use the two function sets $\{+\}$ and $\{+, -\}$. By keeping the operators to plus and minus we ensure that an evolved program will be a linear function of the inputs, possibly making it less accurate, but hopefully, easy to analyse.
3. A reduced terminal set. For the sella point we first use a terminal set which contains only the most obviously useful features together with the terminal set $\{+, -\}$. We then extend the analysis to the full terminal set.

5 Analysis of Evolved Programs

5.1 Easy Landmark: Nose

Function Set is $\{+\}$: The fittest individual, which was evolved in all 80 evolutionary runs, was:

$$\text{Output} = S_5 \quad (2)$$

Recalling that the position of the highest output is the predicted position of the object, and noting from figure 2 that S_5 is the standard deviation of the centre 2 rows of pixels, this program implements a reasonable algorithm for detecting the nose point. Inspection of figure 2 confirms that S_5 will be at a maximum when the input window is positioned over the nose tip. This program has a detection accuracy of 65.9%. It fails on a number of images, like the lower image in figure 3, in which there are some edge effects near the nose. However, it appears that the methodology is producing the best program possible given the restricted function set.

Function Set is $\{+, -\}$: In this situation all of the evolved programs can be simplified to the form given in equation 3. All α_i and β_i will be integers.

$$\text{Output} = \alpha_1 M_1 + \beta_1 S_1 + \alpha_2 M_2 + \beta_2 S_2 + \cdots + \alpha_5 M_5 + \beta_5 S_5 \quad (3)$$

Analysis of the best programs evolved in 80 runs revealed the following underlying algorithms:

$$\text{Output} = M_1 - S_2 - S_3 - 2M_4 + M_5 \quad (4)$$

$$\text{Output} = M_2 - 2S_2 - 2M_4 + M_5 \quad (5)$$

$$\text{Output} = 2M_1 - S_1 - 2M_4 - S_4 \quad (6)$$

$$\text{Output} = M_2 - 2S_2 - 2M_4 + M_5 \quad (7)$$

Analysis of an Individual Program: The following is an analysis of how the detection program described in equation 4 predicts the position of the landmark based on the values of features at six carefully chosen positions on an image. Each position used in figure 3 is indicative of regular patterns that occur in the training images. The window is located on soft tissue(1), the background(2), soft-tissue/background edge(3) and centred on the known position of the nose




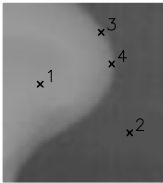




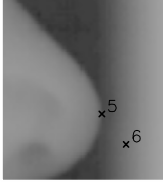


		Position		Output	
					
			$M_1 - S_2 - S_3 - 2M_4 + M_5$		
	1		$23.3-0.9-1.3-45.8+22.3=-2.4$		
	2		$4.5-0.5-0.5- 8.6+ 4.8=-0.3$		
	3		$11.6-5.1-3.3-22.4+12.0=-7.2$		
	4		$10.2-3.9-0.3-14.2+10.8= 2.6$		
		Highest $Err=(1,-1)$	$9.2-4.7-0.2-10.5+ 9.8= 3.6$		
	5		$10.1-3.1-1.4-15.5+10.6= 0.7$		
	6		$6.8-1.3-2.7-11.0+ 6.7=-1.5$		
	Highest $Err=(1,0)$	$9.5-3.9-1.8-10.2+10.0= 3.6$			

Fig. 3. Sample output using detection program, $M_1 - S_2 - S_3 - 2M_4 + M_5$, applied to six different positions. Output is based on features calculated using a greyscale image

landmark(4). Positions 5 and 6 are on an image which has some edge effects near the nose.

If the program is evaluated when the input window is located on an area of constant brightness, such as positions 1 and 2, the output of the program is ≈ 0 . The mean pixel intensities of each of the shapes, M_i , will be approximately the same and the standard deviations, S_i , will be ≈ 0 . On edge position (3) the program output will be negative because S_1 and S_2 will be large. When the input window is centred on the true position (4), the program produces a high output in comparison to the previous positions. If we compare the outputs at positions 3 and 4 we observe that when the input window is located on a diagonal edge the most significant component for varying the output is terminal M_4 . If the input window moves either side of the soft-tissue/boundary, the value of either S_2 , the standard deviation of the left half, or S_3 , the standard deviation of the right half, will decrease the output because of the negative coefficients. Figure 3 also shows the value of the highest output of the evolved program and the error in pixels in the x and y directions from the true position.

The analysis is consistent with the graph shown in Figure 4 which is a visualisation of the output of the program superimposed on a grey level image. From this analysis we can conclude that the program implements a reasonable algorithm for finding the nose point. A similar analysis performed for equations 5-7

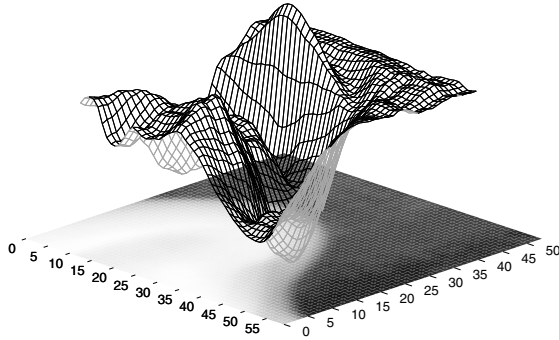


Fig. 4. Output from individual, $M_1 - S_2 - S_3 - 2M_4 + M_5$

reveals that they also implement reasonable algorithms. The accuracy achieved by these programs is no different from the accuracy achieved by programs using the full function set $\{+, -, \times, \%\}$ which indicates a linear problem.

Analysis of regularly occurring patterns across programs: It is clear that the function set used contains some redundancy, M_1 , M_2 and M_3 , for example, are obviously related. If we remove some of the redundancy by using the relationships $M_1 = \frac{1}{2}(M_2 + M_3)$ and $M_1 \approx M_5$, we obtain the programs shown in the central column of table 1. The program of equation 4 reduces to the fourth one in this table. Simplifying the best programs from 80 runs in the same way revealed that a number of underlying algorithms were repeatedly being evolved. For example, 13 runs produced the program shown in the first line of the table. Also, in all evolved programs that achieved a 100% detection rate, the sum of the coefficients of the M_i features used was 0 and the coefficients of the S_i features were always negative.

Figure 5 shows that there is a very large variation at the LISP level in programs that implement the same underlying algorithm.

5.2 Difficult Landmark: Sella

Due to the complexity of this landmark and the large number of features, we have carried out the analysis of the evolved programs in two stages. In the first stage we analyse runs using a reduced terminal set and in the second stage we

Table 1. Frequency of occurrence of detection program in 80 runs

Frequency	Program	Detection Rate
13/80	$1.5M_2 - 2S_2 + .5M_3 - 2M_4$	100% (82/82)
13/80	$1.5M_2 - S_2 + .5M_3 - 2M_4 - S_4$	100% (82/82)
7/80	$0.5M_2 - S_2 - .5M_3 - S_3 + S_5$	98.7% (81/82)
6/80	$M_2 - S_2 + M_3 - S_3 - 2M_4$	100% (82/82)
4/80	$-S_1 + M_2 + M_3 - 2M_4 - S_4$	100% (82/82)

```
(- (- M5 S2) (+ M4 (+ S2 (- M4 M2))))
(- (+ (- M5 (+ M4 (+ M4 (- S2 M2)))) S5) (+ M1 S2))
(+ (+ (+ (- (- S5 M4) S2) (- (- (- S2 (+ S2 M3)) M4) (+ S1 S5)))
(+ M4 M3))(- (- (+ (+ M4 M5) (+ (+ M5 S1) (+ M3 M1))) (+ M3 M1))
(+ (+ S5 (+ (+ M4 M3)M4)) (- M2 (+ (- M2 S2) (+ M1 M2))))))
```

Fig. 5. Programs equivalent to $1.5M_2 - 2S_2 + 0.5M_3 - 2M_4$

use the full terminal set. In both stages we use the function set $\{+, -\}$. Runs with just $\{+\}$ for the function set did not result in programs that were accurate enough to be worth analysing.

Reduced Terminal Set. The full feature set used for this landmark are shown in the lower half of figure 2. Features M_1, M_2, S_1 and S_2 were obtained from a segmentation of training images using pulse coupled neural networks [7] and are clearly the most discriminating features. Due to the large number of features we first analyse runs which use just these four features. After performing 80 runs and simplifying the evolved programs, we found that the best programs were all variants of two underlying algorithms:

$$\begin{aligned} Output &= 5M_1 - 5S_1 - 5M_2 - 2S_2 & (8) \\ &= M_1 - S_1 - M_2 - 0.4S_2 \end{aligned}$$

$$\begin{aligned} Output &= 3M_1 - 3S_1 - 3M_2 - S_2 & (9) \\ &\approx M_1 - S_1 - M_2 - 0.3S_2 \end{aligned}$$

Programs implementing equation 8 achieved a test accuracy of 70.7% while programs implementing equation 9 achieved a test accuracy of 69.5%.

Analysis of an Individual Program: We performed similar analyses to that shown in figure 3 for equations 8 and 9. As before, the analyses revealed that the algorithms are reasonable ones in the context of the window sweeping through the image with the position of highest output chosen as the predicted position of the sella landmark. In summary, features M_1 and M_2 assist with differentiating between the true position and a scene of constant brightness by the positive output of $M_1 - M_2$, while S_1 and S_2 are used for differentiating between cluttered scenes and the known position.

Analysis of Regularly Occurring Patterns Across Programs: As for the nose point, in all of the best evolved programs, the sum of the coefficients of the M_i features used was 0 and the coefficients of the S_i features were always negative.

Full Terminal Set. Eighty evolutionary runs using the full set of fourteen features shown in figure 2 were carried out. The linear functions shown in Equations 10-13 are the fittest programs from four randomly chosen evolutionary runs.

$$Output = 5M_1 - 2M_2 - S_2 - 2M_3 - 3S_3 - 2S_4 + S_5 - M_6 + S_7 \quad (10)$$

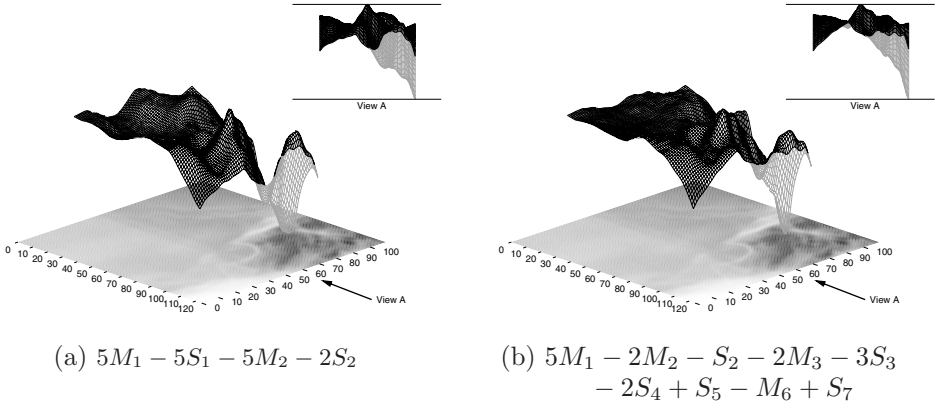


Fig. 6. Visualisation of output landscape, sella point, reduced function set (a), full function set (b)

$$Output = 6M_1 - 2S_1 - 5M_2 - 3S_2 - M_3 - 3S_4 + M_5 + S_5 \quad (11)$$

$$Output = 3M_1 - 3S_1 - M_2 - 4S_2 - M_3 + 4S_3 - S_4 - M_6 + S_7 \quad (12)$$

$$Output = 6M_1 - S_1 - 5M_3 - 3S_3 - 2S_4 + S_5 - 2M_6 + M_7 + S_7 \quad (13)$$

Analysis of an Individual Program: Comparison of these functions with equations 8 and 9 reveals that equation 11 could be considered as a refinement of equation 8. However, other similarities are difficult to find. Placing the input window in a number of positions and computing the outputs as before, shows that the output is highest around the true position of the landmark, but, unlike the previous two cases, does not reveal any intuition about why the program works.

Analysis of regularly occurring patterns across programs: After performing similar simplifications and substitutions as for the nose point, we found that the sum of the coefficients of the M_i terms in an equation was 0, as in the previous two situations. We have not been able to determine the significance of this, however the regularity is striking. However, the coefficients of the S_i were no longer all negative, as before. We have examined three dimensional views, such as figure 4, of program output. Many of these views are minor variations of the landscapes shown in figure 6. This provides additional evidence that the same underlying algorithms are being evolved.

6 Conclusions

Our aim was to determine whether we could develop explanations of how the evolved programs work for an object detection problem. We have succeeded in this to a large extent. The programs no longer need to be presented as some

kind of black box that works by magic. We have shown that a methodology of simplifying function and terminal sets and simplifying the resulting programs to linear functions yields insight into the underlying algorithms implemented in the evolved programs. We found that underlying regularities were being consistently discovered in many repeated runs and that the regularities could be expressed as linear functions which were realistic in the context of the input window sweeping across the image to find the point of interest.

In the context of this particular object detection problem, the methodology works well for up to 4 terminals. For this number of terminals it is possible to comprehend the inter-relationships between the components of the linear function in the context of the sweeping process. For more than 4 terminals this is very difficult. However, the fact we have been able to identify the underlying algorithms for simplified problem instances gives us confidence that, in situations where there are too many terminals and functions to permit understanding, that the evolved programs are still capturing regularities of the domain.

We were surprised that the accuracy of the best programs did not drop with the reduced function set. It appears that genetic programming is very good at finding linear models for these kinds of problems.

References

1. Walter Alden Tackett. Mining the genetic program. *IEEE Expert*, 10(3):28–38, June 1995.
2. I. De Falco, A. Della Cioppa, and E. Tarantino. Discovering interesting classification rules with genetic programming. *Applied Soft Computing*, 1(4):257–269, May 2001.
3. Andy Song and Vic Ciesielski. Texture analysis by genetic programming. In Garrison Greenwood, editor, *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, volume 2, pages 2092–2099. IEEE, June 2004.
4. Vic Ciesielski, Andrew Innes, John Mamutil, and Sabu John. Landmark detection for cephalometric radiology images using genetic programming. *International Journal of Knowledge Based Intelligent Engineering Systems*, 7(3):164–171, July 2003.
5. J. Cardillo and M.A. Sid-Ahmed. An image processing system for locating craniofacial landmarks. *IEEE Transactions on Medical Imaging*, 13(2):275–289, Jun 1994.
6. Y. Chen, K. Cheng, and J. Liu. Improving cephalogram analysis through feature subimage extraction. *IEEE, Engineering in Medicine and Biology Magazine*, 18(1):25–31, Jan–Feb 1999.
7. Andrew Innes, Vic Ciesielski, John Mamutil, and Sabu John. Landmark detection for cephalometric radiology images using pulse coupled neural networks. In Hamid Arabnia and Youngsong Mun, editors, *Proceedings of the International Conference on Artificial Intelligence (IC-AI'02)*, volume 2, pages 511–517, Las Vegas, June 2002. CSREA Press.