Genetic Programming for Feature Detection and Image Segmentation.

Riccardo Poli
School of Computer Science
The University of Birmingham
E-mail: R.Poli@cs.bham.ac.uk

Abstract

Genetic Programming is a method of program discovery/optimisation consisting of a special kind of genetic algorithm capable of operating on non-linear chromosomes (parse trees) representing programs and an interpreter which can run the programs being optimised. In this paper we describe a set of terminals and functions for the parse trees handled by genetic programming which enable it to develop effective image filters. These filters can either be used to highly enhance and detect features of interest or to build pixel-classification-based segmentation algorithms. Some experiments with medical images which show the efficacy of the approach are reported.

1 Introduction

Genetic Programming (GP) is the extension of Genetic Algorithms (GAs) in which the structures that make up the population under optimisation are not fixed-length character strings that encode possible solutions to a problem, but *programs* that, when executed, *are* the candidate solutions to the problem [1, 2].

Programs are expressed in GP as parse trees, rather than as lines of code. For example, the simple expression `max(x * x, x + 3 * y)` would be represented as shown in Figure 1. The set of possible internal (non-leaf) nodes used in GP parse trees is called *function set*, $\mathcal{F} = \{f_1, \cdots, f_{N_F}\}$. All functions of $\mathcal{F}$ have *arity* (the number of arguments) greater than one. The set of terminal (leaf) nodes in the parse trees representing programs in GP is called *terminal set* $\mathcal{T} = \{t_1, \cdots, t_{N_T}\}$. Table 1 shows some typical functions and terminals used in GP.

$\mathcal{F}$ and $\mathcal{T}$ can be merged into a uniform group $\mathcal{C} = \mathcal{F} \cup \mathcal{T}$ if terminals are all considered as 0-arity functions. The *search space* of GP is the set of all the possible (recursive) compositions of the functions in $\mathcal{C}$. The basic search algorithm used in GP is a classical GA with mutation and crossover specifically designed to handle parse trees.

GP has been applied successfully to a large number of difficult problems like automatic design, pattern recognition, robotic control, synthesis of neural networks, symbolic regression, music and picture generation, etc. However, a relatively small number of application of GP in the domain of image processing and computer vision have been reported in the literature. A short description of them follows.

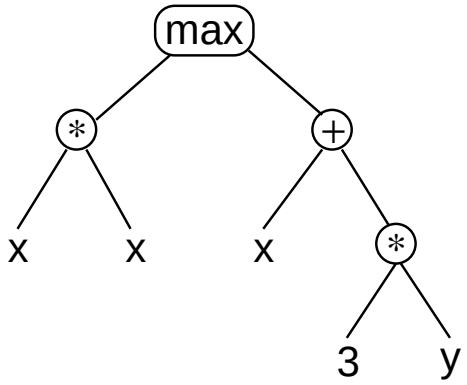


Figure 1: Parse-tree representation of the expression `max(x * x, x + 3 * y)`.

Functions	
Kind	*Examples*
Arithmetic	+, *, /
Mathematical	sin, cos, exp
Boolean	AND, OR, NOT
Conditional	IF-THEN-ELSE, IFLTE
Looping	FOR, REPEAT

Terminals	
Kind	*Examples*
Variables	x, y
Constant values	3, 0.45
0-arity functions	rand, go_left
Random constants	random

Table 1: Typical functions and terminals used in genetic programming.

Koza [2, Chapter 15] has applied GP to the problem of classifying 4×6 binary patterns representing the character "L", the character "I" or noise. The terminals used in the experiments were low-level operators moving a cursor within the bit map or returning the intensity of the pixels around the cursor. In his account Koza reported that, despite using population with 8,000 individuals, no perfect classifier was discovered using the basic GP paradigm. On the contrary, using GP with automatically defined functions (ADFs) playing the role of problem-dependent higher-level feature-detectors made the recognition task much easier.

A similar approach has been adopted by Andre [3] to solve the slightly more complicated task of classifying digits represented as 5×5 bit maps, with and without noise. Andre used a hybrid algorithm in which classification programs (one for each digit) were optimised by using GP, while automatically defined feature-detectors (represented as 2-D arrays of characters from a ternary alphabet) were optimised using a GA. Thanks to the exploitation of relatively large-scale 2-D features, Andre obtained good results with reasonable computational effort.

Johnson *et al.* [4] applied GP to a fiducial-point-localisation problem (a problem which does not involve recognising the presence or absence of a structure in the image, but the localisation of a structure which is known to be present). The problem was to find the left and right hands in the bitmap silhouette of a person, obtained from real images via segmentation. By using high-level functions and terminals incorporating knowledge about the task the authors have been able to obtain programs which correctly locate the left arm in 93% of the images, and the right arm in 70%.

Tackett [5] was the first to report on GP applications to non-binary images of cluttered environments. He used GP to determine whether a target (tank) was present in each of a set of overlapping large sub-windows of IR images. Two different terminal sets were used for the classification: one based on image features (e.g. average intensity, variance, etc.), the other on higher-level feature obtained from the bitmaps resulting from the segmentation of each window. Tackett showed how, on large database of training and testing samples, GP performed better than multi-layer perceptrons and binary classification trees.

In a system called PADO (Parallel Algorithm Discovery and Orchestration), Teller and Veloso [6] used a combination of GP and linear discrimination to obtain classification programs for signals and images. PADO programs are actually made up of several groups of programs (called systems) each group being used to recognise the instances of a given class. The system with the highest output determines the output of PADO. The output of a system is a linear combination of the output of the programs included in the system. The function set used in PADO is quite comprehensive including, for example, memory read/write primitives, large-scale image operators (e.g. mean and variance of intensity in a window, min and max pixels, etc.), ADFs and library functions. In experiments with images of seven different objects, PADO has shown a correct-classification rate of between 60 and 70%.

Most of the applications described above are concerned with the problem of recognising the structure represented in an image. A different objective is the one described recently by Harris and Buxton [7] who used GP to discover optimum linear filters for edge detection in signals. In this approach, GP has to evolve functions which are sampled to build masks (linear FIR filters). The masks are then convolved with a set of reference signals for which a desired output is known. The results obtained with this method are better than those obtained by Canny's edge detector, one of the best operators described in the literature. Extending the approach to image analysis is possible, but the computation involved in convolving large 2-D masks (a different one for each new individual) with a set of images can be extremely large.

In this paper, more than with the recognition of the structure(s) present in the image, we are concerned with the design of effective problem-specific filters capable of highly and selectively emphasise some characteristics of the image. On the ground of this behaviour, the filters, coupled with simple thresholding strategies, can either be used to detect features of interest like edges or texture or to build pixel-classification-based segmentation algorithms.

The paper is organised as follows. In the next section we describe an approach to low-level image analysis based on properly designed filters. In Section 3, a set of terminals and functions which enable GP to develop efficient linear or non-linear filters is presented. In Section 4 we report on some experiments with medical images which show the efficacy of our functions and terminals. Finally, in Section 5, we draw some conclusions.

2 Low-level Image Analysis as Filtering

Our approach to using GP for image analysis is based on the idea that most low-level image analysis tasks, namely image enhancement, feature detection and image segmentation, can be reframed as image filtering problems and that GP can be used to discover optimal filters which solve such problems.

In the case of image enhancement the objective can be to improve the actual quality of an image, for example by removing the noise present in it or by reducing the blurring due to the optics of the camera. Alternatively the objective can be to emphasise some structures of interest in the image, for example to help the visual perception of such structures. In both cases, image enhancement is normally performed with some kind of filter, which, applied to the image, transforms it into another image with the desired characteristics.

In feature detection the objective is usually to locate the position (and possibly the shape) of given kinds of (usually small) structures present in the image. Examples of such structures can be edges, ridges/lines, bars, corners, textural elements of various kinds, but also any other problem-specific item. In most cases feature detection involves two steps: a pre-filtering of the image which enhances the features of interest, followed by a decision phase in which thresholding or maxima-detection techniques are used to locate the features of interest in the enhanced image.

In general, image segmentation consists of labelling the pixels in the image with a small number of labels so that pixels belonging to regions with (more or less) homogeneous gray level have the same label and pixels belonging to regions with significantly different gray level have different labels. Many techniques are available in the literature to solve this problem. A number of them consider the problem of segmenting an image as a pixel classification problem where each pixel is assigned to a class considering local information only (usually the gray levels of the pixels in a neighbourhood of the pixel being classified or information derived from them).

When the number of classes in known in advance and hand-segmented images are available neural networks and other statistical classification techniques can be used for this purpose. In the case of neural networks, the approach is either to use a net with several outputs (one for each class) and assign each pixel to the class whose output neuron is maximally active, or to use as many different nets with a single output as the number of classes and select the class whose net provides the maximum output. In both cases there are as many input neurons as the pixels in the neighbourhood and the net is used as a kind of *non-linear filter* (possibly with multiple outputs).

Quite often in practical applications the objective of segmentation is more specific and more difficult to achieve: it is to label the pixels belonging to one or more structures of interest as foreground and all the rest as background, even if there are conspicuous intra-class inhomogeneities and inter-class similarities. In these cases segmentation-as-pixel-classification, when at all possible, requires the exploitation of both statistical information about the gray levels of the structures of interest and (possibly more importantly) of other large scale regularities such as the typical shape of the structures of interest. These regularities can only be captured giving the filter the information contained in large neighbourhoods of pixels.

It should be noted that among the problems mentioned above image segmentation is the most difficult one, and that standard filtering techniques (in particular linear FIR and IIR filters) usually provide insufficient discriminative power for a reliable classification. Only methods capable of considering all the available sources of information and of combining them in a strongly non-linear way, like artificial neural networks and some ad-hoc combinations of morphologic filters, seem to be able to give (relatively) good results.

However, both neural networks and morphologic operators offer only very peculiar kinds of non-linearities. If their relatively good performance derives from the use of large

neighbourhoods and the presence of non-linearities, it is arguable that a much larger space of methods exists with such features. Such a space can only be explored with an unbiased machine-learning technique, such genetic programming, which does not impose the use of a specific form of non-linearity, or a fixed shape for neighbourhoods.

3 GP for image segmentation and feature detection

3.1 Terminal set

A first idea to use GP for image filtering, and successively for pixel-classification, is to adopt a NN-like strategy, i.e. to use as many variables as the number of pixels in a neighbourhood of fixed size, and then to apply the filter (i.e. run the program) to each pixel in the image (while appropriately shifting the neighbourhood at each run). The variables would represent the gray level of the pixels in the neighbourhood of the current pixel. The output of the program (i.e. of the root node) would be taken to be the output of the filter.

However, in a large set of preliminary experiments in which various combinations of functions and terminals have been tried, we have reached the conclusion that this approach simply does not work. We believe that the reason is that even neighbourhoods of modest size can lead to a very large terminal set including tens to hundreds of variables. As a consequence, very large (sub-)programs are necessary in order to extract any form of large-scale statistical descriptor to be used to determine a correct output for the program. Although finding such big subtrees is not impossible, it was clearly beyond the computational power available.

As a consequence, we decided to try and use a more concise and efficient set of variables having the following features:

- The set must be capable of capturing the information present in the image at different scales.
- During a run of GP, the process of binding the variables for each evaluation of a program should not require complex calculations as the fitness functions for image analysis (see below) include thousands of fitness cases. The ideal set would require only memory accesses. This is possible, for example, if all the operations necessary to evaluate the terminals are performed once and for all in the initialisation phase of GP and the results are stored in separate arrays.
- If efficiency of the programs discovered via GP is sought, the computation load necessary to evaluate the aforementioned arrays should be as light as possible.

While the first requirement can be met by many different sets of terminals, the second and third constrain considerably the possible choice. In addition to the classic random constant generator, we selected the set of terminals shown in Table 2, where $I(x, y)$ is the gray level of the pixel at coordinates (x, y), and (x_c, y_c) are the coordinates of the current pixel (i.e. of the centre of the neighbourhood). The terminals are the value taken by the convolution of the image with "box" operators (i.e. constant masks) of different sizes.

This set of terminals meets all the requirements listed above:

Terminals	
Name	*Definition*
`I`	$I(x_c, y_c)$
`I_3x3`	$\frac{1}{9}\sum_{d_x=-1}^{1}\sum_{d_y=-1}^{1} I(x_c+d_x, y_c+d_y)$
`I_7x7`	$\frac{1}{49}\sum_{d_x=-3}^{3}\sum_{d_y=-3}^{3} I(x_c+d_x, y_c+d_y)$
`I_15x15`	$\frac{1}{225}\sum_{d_x=-7}^{7}\sum_{d_y=-7}^{7} I(x_c+d_x, y_c+d_y)$

Table 2: Terminal set used in our image processing experiments.

Functions		
Name	*Arity*	*Definition*
`Add`	2	$a_1 + a_2$
`Sub`	2	$a_1 - a_2$
`Mul`	2	$a_1 \times a_2$
`Div`	2	$\begin{cases} a_1/a_2 & \text{if } \lvert a_2\rvert \geq 0.00001 \\ a_1 & \text{otherwise} \end{cases}$
`Max`	2	$\max(a_1, a_2)$
`Min`	2	$\min(a_1, a_2)$

Macros		
Name	*Arity*	*Definition*
`Xcp1`	1	`Eval(`a`)` with $x_c = x_c + 1$
`Xcm1`	1	`Eval(`a`)` with $x_c = x_c - 1$
`Ycp1`	1	`Eval(`a`)` with $y_c = y_c + 1$
`Ycm1`	1	`Eval(`a`)` with $y_c = y_c - 1$

Table 3: Function set used in our image processing experiments.

- It captures both fine scale and broader scale information. In fact, it is well known in image processing, that by combining properly shifted and scaled box operators, any filter (including Gaussians, derivatives of Gaussian, Laplacians, Canny's, etc.) can be approximated to the desired degree of accuracy.
- The moving averages `I_3x3`, `I_7x7` and `I_15x15` can be computed very efficiently, in the initialisation phase, with an algorithm requiring only 4 additions per pixel instead of N^2 (with $N = 3, 7, 15$). These terminals can be then stored in appropriate arrays, so that their evaluation at runtime requires only a memory access.

3.2 Function set

Requirements similar to those for the terminal set apply to the function set as well. In order to meet these requirements, in our experiments we preferred not to include in the function set functions which require image processing operations. The reason is that the computation performed by such functions would depend on the values of their arguments. Those values may vary considerably between different individuals, within a single individual and, more importantly, from pixel to pixel (they are functions of the terminals). As a result it is impossible pre-compute and store the values to be returned by such functions so that the evaluation of the fitness of each individual may require a computation load orders of magnitude bigger than if such functions are not used. In most of our experiments we used the functions set shown in Table 3, where a, a_1 and a_2 are dummy arguments.

It is worth noting that the macros `Xcp1`, `Xcm1`, `Ycp1` and `Ycm1` allow the construction of filters whose complexity is well beyond the one provided by the terminal set. In addition, the presence of `Min`, `Max`, `Div` and `Mul` allows the construction of highly non-linear filters (e.g. mathematical-morphology-like operators). These functions in conjunction with `Add` and `Sub` allow the creation of any necessary constants, too.

3.3 Fitness functions

If the objective is to develop filters for image enhancement, it is possible to use a symbolic-regression-like fitness function in which the output of the program at each location is compared with the desired output. The sum of the absolute errors made by the program for all the pixels of all the images of a training set can then be transformed into a fitness function with the usual scaling or mapping techniques. The availability of desired-output images and the huge number ($\approx 10^6 - 10^7$) of runs necessary to evaluate the fitness of each program are the problems hampering this approach. The second problem can be reduced considerably by sampling the images of the training set in, say $10^3 - 10^4$, random points and running the program only in those locations. However, the first problem remains as hand retouching seems unacceptable for real-world images. The only solution is to use other, much more expensive filtering techniques (say a Kalman filter) or to use different image acquisition technologies to provide the desired output.

If the objective is to develop filters for feature detection or image segmentation, then using a symbolic-regression-like fitness function that forces the output of the program to exactly match the binary values representing the structures to be segmented/detected in a set of training images seems inappropriate. There are two reasons for this.

The first reason is that the output of the filter will have to be passed to a decision algorithms which in most cases will simply apply a threshold to it in order to make a decision as to which class to assign the pixel in (x_c, y_c). So all the computational effort spent by GP to build programs with binary outputs is unnecessary in this case. In our work we have tried to make the search that GP has to do considerably easier by using a wrapper for the filtering program. In most cases a wrapper is sufficient which simply sets the output to 1 if the filter's output is positive, to 0 otherwise.

The second reason has to do with the sensitivity/specificity dilemma that any segmentation or detection algorithm has to face: with real-world images no algorithm can detect/segment the points belonging to the structures of interest (True Positives, TPs) without wrongly detecting also some points which belong to uninteresting structures (False Positives, FPs) and missing some points (False Negatives, FNs). Each algorithm will have both a *sensitivity* (number of TPs divided by the number of points to be detected) and a *specificity* (number of True Negatives, TNs, divided by the number of points not to be detected) smaller than 1. Changing the settings for the parameters of the algorithm can only increase the specificity and decrease the sensitivity or vice-versa. The optimum setting for an algorithm is usually one in which both the sensitivity and the specificity are "as big as possible".

The fitness function used in symbolic regression is essentially the sum of the number of FPs and FNs:

$$f = FP + FN.$$

For a fixed training set, FP and FN determine the sensitivity and specificity of each program. However, optimising their sum does not necessarily mean to achieve an optimum

sensitivity/specificity trade-off. On the contrary in most cases, GP tends to discover algorithms which are either very sensitive (and poorly specific) or vice versa.

In our experiments we have tried several different ways to optimise the sensitivity/specificity trade-off considering also that, depending on the particular detection task, FPs and FNs can have a completely different impact on the quality of the results. For example, in some tasks may be totally unacceptable to have false detections (FPs), while in others it is much better to have false detections than misses (FNs) (think for example of the detection of micro-calcifications in breast radiograms).

The fitness function that seems to be giving the best results is the following:

$$f = FP + FN \exp\left(10\left(\frac{FN}{P} - \alpha\right)\right)$$

where P is the number of pixels belonging to the structures to be detected and α is a domain dependent parameter which represents the maximum percentage of misses above which the number of misses is considered unacceptable (typically $\alpha = 0.6-0.9$). Basically, if FN/P is sufficiently smaller than α, $f \approx FP$, i.e. GP tries to minimise the number of false detections; if FN/P is slightly bigger that α, $f \approx f(FN)$, i.e. GP tries to minimise the number of misses.

4 Experimental Results

In this section we will describe a small set of experiments performed with the functions and terminal described in the previous section. As one of our objectives is to obtain filters that can perform optimal segmentation and detection in complex real-world gray scale images, we did not even try to apply GP to simplified problems involving the filtering of binary images, synthetic images or of small size images. On the contrary we decided to use real images used in one of the most difficult domains: medical imaging.

4.1 Segmentation of the brain in Magnetic Resonance Images

In this experiments we considered the problem of segmenting the brain in pairs of Magnetic Resonance (MR) images (each image representing a map of the intensity of a different physical property of the tissue being imaged).

Pixel classification strategies based on neural networks or other statistical approaches have been used in the past for this task with limited success [8]. The difficulty resides in the fact that some kinds of the tissue in the brain present the same physical properties as some tissues in the skin (see Figure 2). As a consequence, brain segmentation based on small neighbourhoods is basically impossible. However, this does not mean that using large neighbourhoods, the task becomes easy. To show this and also to provide a reference for comparison with GP, we have performed some experiments using neural networks in conjunction with medium and large neighbourhoods.

Figure 3 shows the results obtained with neural nets trained with the backpropagation algorithm, having the following architecture: a) two squared "retinas" of $N \times N$ input neurons (one for each input image) fed with the gray levels of the pixels in the neighbourhood, b) ten hidden neurons, c) one output neuron whose activation determines whether the pixel at the centre of the retinas belongs to the brain or not. Two different sizes for the retinas have been tried, 5×5 and 15×15, corresponding to medium and coarse scale

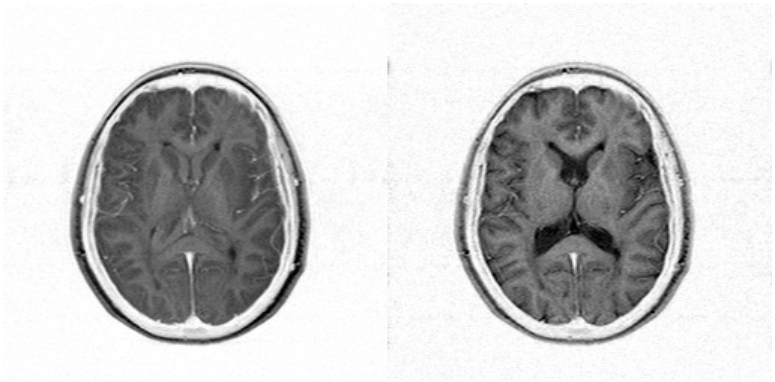


Figure 2: Two original MR images of the head.

(for 256×256 pixel wide images). Figures 3(a)–(b) show the output produced by the nets. Despite the optimum adjustment of the thresholds and the very large number of degrees of freedom in the networks (more than 500 for the smaller net, more than 4500 for the larger one), the segmentations shown in Figures 3 are disappointing. In particular, they seem to suggest that small retinas provide insufficient information to avoid FPs in the skin region, while large retinas lead to FNs due to the inability to correctly classify the brain edges. With feed-forward neural nets better results can only be obtained by exploiting also anatomical information [9].

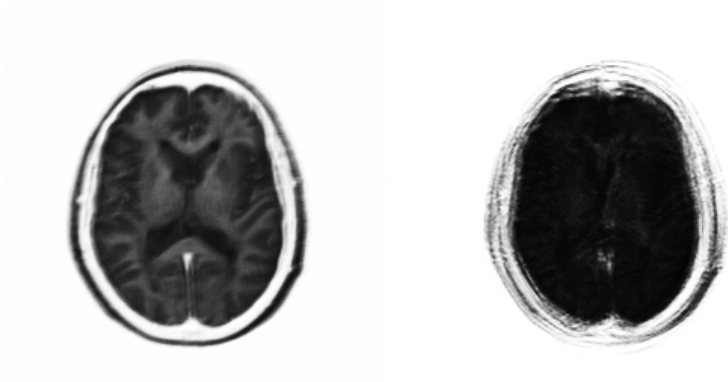
Figure 4 shows the results obtained in the same task by the following program with fitness $f = 16.6004$ discovered by GP after 40,000 fitness evaluations:

```
(Sub (Max MR0_15x15 (Max (Max (Mul (Max (Max (Mul (Max MR0_15x15 MR0_15x15)
MR0_15x15) (Sub MR1_15x15 MR0_15x15)) MR0_15x15) MR0_3x3) MR0_15x15) (Mul (Add
(Mul (Max MR0 (Max (Mul (Add (Mul (Div 0.55 (Max (Max (Mul MR0 MR0_7x7) (Max
(Max MR0_15x15 MR0_15x15) (Mul (Mul (Mul MR0_15x15 MR0_15x15) MR0_15x15)
MR0_15x15))) (Max (Mul (Add MR0_15x15 (Max (Max (Mul (Max (Mul -0.75 MR0_15x15)
(Max (Mul (Add MR0_15x15 (Mul (Mul (Add (Mul MR1_15x15 MR0_7x7) (Mul (Min
MR0_15x15 MR0_15x15) MR0_15x15)) MR1_15x15) MR0_15x15)) MR0_15x15) (Mul (Div
(Add (Mul (Mul (Add (Mul 0.55 MR1_3x3) (Mul (Min MR1_15x15 (Mul (Max (Mul -0.75
MR0_15x15) (Max (Mul (Sub MR0_15x15 MR0_15x15) MR0_15x15) (Mul (Div (Add (Mul
(Add (Mul 0.55 MR1_3x3) (Mul (Min MR1_15x15 MR0_15x15) MR0_15x15)) MR1)
MR0_15x15) 0.55) (Mul (Max (Max (Add MR1_3x3 MR0_15x15) (Sub MR0_15x15
MR0_15x15)) MR0_15x15) MR0_15x15)))) MR0_15x15)) MR0_15x15)) MR1) MR1)
MR0_15x15) 0.55) (Mul (Max (Max (Add MR1_3x3 MR0_15x15) (Sub MR0_15x15
MR0_15x15)) MR0_15x15) MR0_15x15)))) MR0_15x15) MR0_15x15) (Mul (Add (Mul 0.55
MR0_3x3) MR0_15x15) MR0_15x15))) MR0_15x15) MR0_15x15))) MR0_7x7) MR0_15x15)
0.98) MR0_15x15)) MR0_15x15) 0.55) MR0_15x15))) 0.55)
```

Although the understandability of this program is minimum, i.e. comparable with a neural net, its performances are significantly closer to the desired ones.

4.2 Detection of Blood vessels in X-ray coronarograms

In X-ray coronarograms blood vessels appear as elongated relatively thin structures connected to form a tree (see Figure 5). The detection of coronary arteries would, therefore, seem to be a relatively simple line-detection task. On the contrary, the presence of noise,



(a) (b)

(c) (d)

Figure 3: Segmentation of the images in Figure 2 produced by neural nets: (a) output of a net with 5×5 input retinas, (b) output of a net with 15×15 input retinas, (c) segmentation (via optimum thresholding) of the image in (a), (d) segmentation (via optimum thresholding) of the image in (b).

(a) (b)

Figure 4: Segmentation of the images in Figure 2 produced with GP: (a) output of a filtering program, (b) segmentation produced by the wrapper applied to the image in (a).

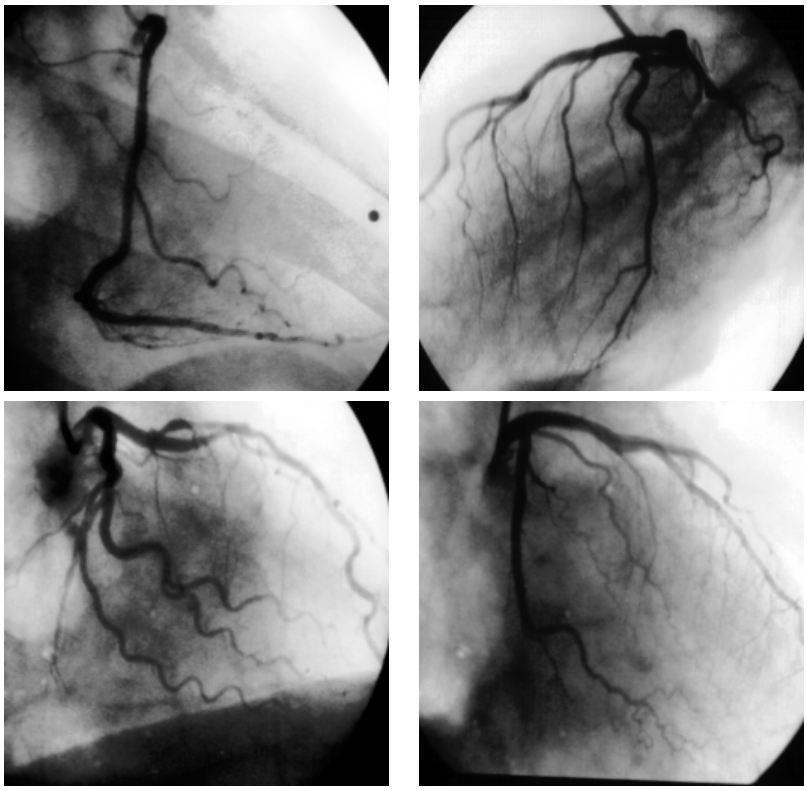


Figure 5: Four original 256×256 X-ray coronarograms.

the variability of the background and the low and varying contrast of vessels make the reliable automatic or even semi-automatic vessel detection a formidably difficult task [10].

Again, in order to provide a comparison for GP we tried to develop and train a set of neural networks to solve the problem. In particular we have used networks with a structure similar to the one described in the previous section but with a single "retina" of $N \times N$ input neurons. Four different retina-sizes have been tried: 5×5, 11×11, 15×15 and 21×21. Figure 6 shows the output obtained by the 15×15 neural network (the best of the four – see below) applied to the images in Figure 5.

In order to obtain better results from these kind of images, we decided to apply a more sophisticated detection algorithms known as hysteresis thresholding [11] which can exploit the connectedness of vessels. Despite the optimum adjustment of the algorithm and the very large number of degrees of freedom in the networks, the segmentations results, summarised in Table 4, were very disappointing for all networks.[1] The problem is that the images contain many thin vessels that need a small 5×5 retina to be properly detected. Larger vessels can only be partially detected with such a retina. The 15×15 retina detects optimally large vessels, but cannot properly detect the thin ones. In all cases the number of false detections is comparable with the number of correct ones. Figure 7 shows the "vessels" detected from the images in Figure 6.

The results obtained by applying GP to this problem are quite different. GP rapidly

[1]The relatively high values of specificity derives from the large number of non-vessel-representing pixels in the image. A value of 90% means that 10% of such pixels have been taken as representing vessels: a very poor result.

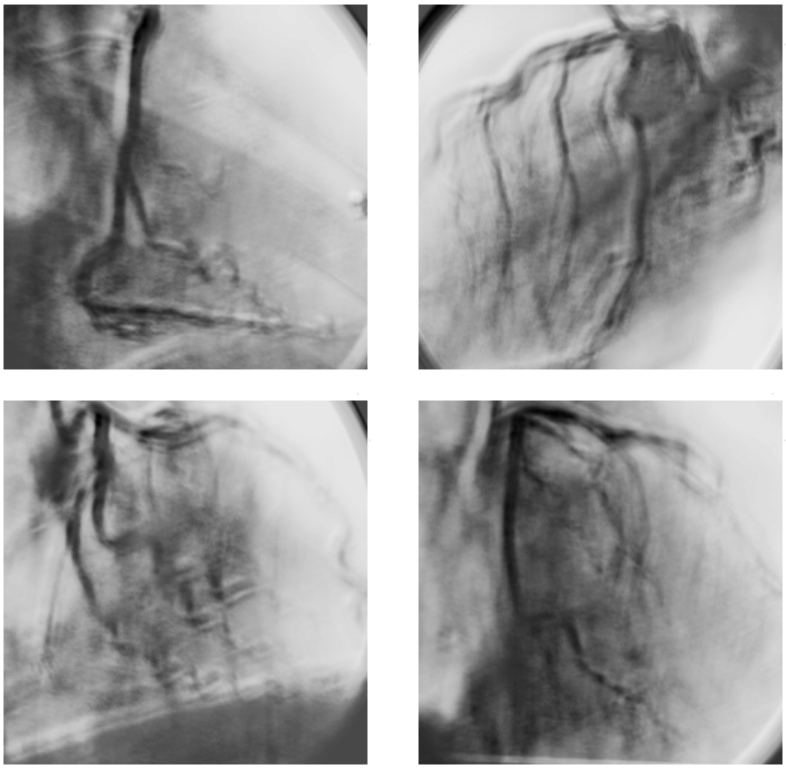


Figure 6: Output of the neural network with 15×15 retina.

Figure 7: Hysteresis thresholding applied to the images in Figure 6.

Network	*Sensitivity*	*Specificity*
5×5	27.3%	91.4%
11×11	23.3%	90.5%
15×15	31.7 %	92.2%
21×21	24.4%	91.3%

Table 4: Results of neural-network-based vessel detection.

discovers how to eliminate most of the background variations and how to obtain the maximum from its wrapper, the hysteresis thresholding algorithm. For example, in a run the program `(Sub I I_15x15)` with fitness $f = 5629.36$, sensitivity 58.7% and specificity 98.6% has been discovered after 1,200 fitness evaluations only. In the same run, after approximately 20,000 fitness evaluation the GP has discovered the filter

```
(Mul (Sub I_3x3 I_15x15) (Add (Add 1.01 I_15x15) (Sub (Mul (Sub I (Sub
(Add  1.01 1.01) (Add (Add 1.01   I) (Sub (Mul  (Sub  I (Sub (Add 1.01
1.01)  (Add (Add 1.01 I)  (Sub (Mul (Sub I  (Sub (Add  1.01 1.01) (Sub
I_15x15 (Sub I_3x3 I)))) (Sub  I_15x15 (Sub I_3x3 I_3x3))) I_15x15))))
I_3x3) I_7x7)))) I_3x3) I_7x7)))
```

with fitness $f = 4684.09$, sensitivity 61.5% and specificity 99.2%. This program outperforms the neural networks described above. The results it produces are shown in Figure 8 which after hysteresis thresholding give the vessels shown in Figure 9.

5 Conclusions

In this paper we have presented an approach to image analysis based on the idea of using GP to discover optimal image filters. Although GP could be applied in a naive way to such a problem, we have outlined some criteria that terminal sets, function sets and fitness functions should satisfy in order to make the search feasible and produce efficient filters. On the ground of this criteria we have proposed some simple terminals, functions and fitness functions that in experiments with medical images have enabled GP to outperform neural nets and, we believe, many other techniques reported in the image-analysis literature, too.

The research on genetic programming for image analysis is hampered by the tremendous demand of computational resources involved in fitness evaluation. Despite our efforts towards efficiency, this has prevented us and, we believe, anybody else from being able to produce the good result-documentation (e.g. with plots of the probability of reaching a certain fitness value vs. generation number, or the minimum computation effort required, etc.) typical of genetic programming literature.

However, the impressive performances shown (after learning) by GP compared with NNs in the experiments reported in this paper seem to suggest that there is a huge space of image analysis tools much more powerful than those used in image processing nowadays and that GP can be used to start exploring this space. Whether GP will be the ultimate tool to this is certainly an open question. Our results along with those obtained by other researchers certainly seem to suggest that putting research efforts in this area can be a very good investment.

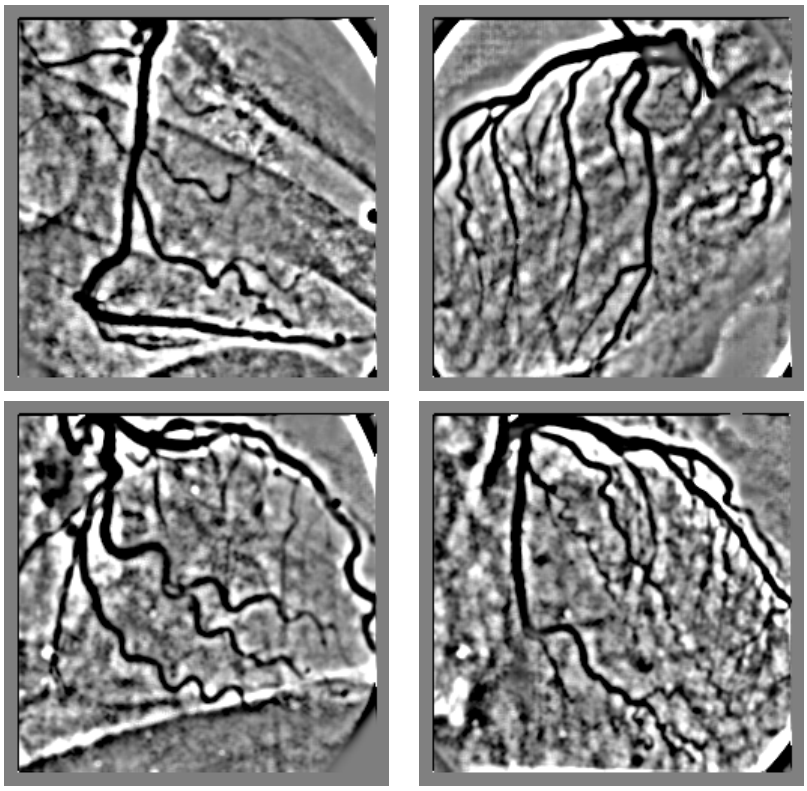


Figure 8: Output of a filter discovered via GP.

Figure 9: Hysteresis thresholding applied to the images in Figure 8.

Acknowledgements

The author wishes to thank Aaron Sloman and all the members of the EEBIC (Evolutionary and Emergent Behaviour Intelligence and Computation) group for useful discussions and comments. This research is partially supported by a grant under the British Council-MURST/CRUI agreement.

References

[1] John R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[2] John R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Pres, Cambridge, Massachusetts, 1994.

[3] David Andre, "Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them", in *Advances in Genetic Programming*, Kenneth E. Kinnear, Jr., Ed., chapter 23. MIT Press, 1994.

[4] Michael Patrick Johnson, Pattie Maes, and Trevor Darrell, "Evolving visual routines", in *ARTIFICIAL LIFE IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems*, Rodney A. Brooks and Pattie Maes, Eds., MIT, Cambridge, MA, USA, 6-8 July 1994, pp. 198–209, MIT Press.

[5] Walter Alden Tackett, "Genetic programming for feature discovery and image discrimination", in *International Conference on Genetic Algorithms*, 1993.

[6] Astro Teller and Manuela Veloso, "PADO: Learning tree structured algorithms for orchestration into an object recognition system", Tech. Rep. CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.

[7] Christopher Harris and Bernard Buxton, "Evolving edge detectors", Research Note RN/96/3, UCL, Gower Street, London, WC1E 6BT, UK, Jan. 1996.

[8] Lawrence O. Hall, Amine M. Bensaid, Laurence P. Clarke, Robert P. Velthuizen, Martin S. Silbuger, and James C. Bezdek, "A comparison on neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain", *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 672–682, 1992.

[9] G. Coppini, R. Poli, M. Rucci, and G. Valli, "A neural network architecture for understanding 3D scenes in medical imaging", *Computer and Biomedical Research*, vol. 25, pp. 569–585, 1992.

[10] G. Coppini, M. Demi, R. Poli, and G. Valli, "An artificial vision system for X-ray images of human coronary trees", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 2, pp. 156–162, 1993.

[11] John Francis Canny, "Finding edges and lines in images", Tech. Rep. 720, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, June 1983.