Genetic Programming for Image Analysis

Riccardo Poli
School of Computer Science, The University of Birmingham
Birmingham B15 2TT, UK
R.Poli@cs.bham.ac.uk

ABSTRACT

This paper describes an approach to using GP for image analysis based on the idea that image enhancement, feature detection and image segmentation can be re-framed as filtering problems. GP can discover efficient optimal filters which solve such problems but in order to make the search feasible and effective, terminal sets, function sets and fitness functions have to meet some requirements. We describe these requirements and we propose terminals, functions and fitness functions that satisfy them. Experiments are reported in which GP is applied to the segmentation of the brain in medical images and is compared with artificial neural nets.

1 Introduction

Genetic Programming (GP) has been applied successfully to a large number of difficult problems like automatic design, pattern recognition, robotic control, synthesis on neural architectures, symbolic regression, music and picture generation, etc [Koza, 1992, Koza, 1994, K. E. Kinnear, Jr., 1994]. However, a relatively small number of applications of GP in the domain of image processing and computer vision have been reported in the literature [Tackett, 1993, Koza, 1994, Andre, 1994, Teller and Veloso, 1995, Breunig, 1995, Riolo and Line, 1995], most of which are concerned with the problem of recognizing/classifying the structure represented in an image.

In this paper, more than with the high-level task of recognizing or classifying the structure(s) present in the image, the focus is on lower-level image-analysis problems such as image enhancement, feature detection and image segmentation. Our approach to solve these problems is based on the idea of using GP to discover effective problem-specific filters capable of highly and selectively emphasizing some characteristics of the image. On the ground of this behavior, these filters, coupled with simple thresholding wrappers, can be used to detect features of interest in the image (e.g. edges or texture) or to build pixel-classification-based segmentation algorithms.

In the next section an approach to low-level image analysis based on properly designed filters is described. Then, in Section 3, the characteristics that terminal sets, function sets and fitness functions should posses in order to enable GP to discover such filters are discussed, and one possible solution is proposed. Section 4 reports on some preliminary experiments in which GP is applied to the segmentation of the brain in magnetic resonance images and compared with artificial neural nets. We draw some final comments in Section 5.

2 Low-level Image Analysis as Filtering

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

2.1 Image Enhancement

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 emphasize 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.

2.2 Feature Detection

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 prefiltering 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.

2.3 Image Segmentation

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 [Ballard and Brown, 1982]. 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 neighborhood 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 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 neighborhood 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 by giving the filter the information contained in large neighborhoods of pixels.

3 GP for Image Analysis

It should be noted that among the problems mentioned in the previous section, 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 neighborhoods 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 neighborhoods.

Although GP can be used to discover efficient optimal filters for image analysis, in order to make the search feasible and effective, terminal sets, function sets and fitness functions have to meet certain requirements.

3.1 Terminal set

A first idea to use GP for image filtering, and successively for pixel-classification, is to use a NN-like strategy, i.e. to use as many variables as the number of pixels in a neighborhood of fixed size, and then to apply the filter (i.e. run the program) to each pixel in the image (while appropriately shifting the neighborhood at each run). The variables would represent the gray level of the pixels in the neighborhood 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, this approach presents some problems. The reason is that even neighborhoods of modest size can lead to a very large terminal set including tens to hundreds of variables. This has the consequence that, if large-scale statistical information is necessary to determine the correct output of the filter, GP will have to build very large (sub-)programs in order to extract such information. This can have a very serious impact on the computation load and memory requirements imposed by GP and ultimately on the feasibility of the approach.[1]

Good sets of terminals for image analysis should meet the following *requirements*:

- They should include a limited number of variables to maintain the size of the search space amenable.
- They should be capable of capturing all the information present in the image at different scales (maybe in conjunction with the function set).
- During a run of GP, the process of binding the variables for each evaluation of a program should not require complex calculations as 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 can be performed once and for all in the initialization phase of GP and the results can be 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.

[1] Actually, in some experiments (not reported) in which various combinations of functions and terminals were tried we had some difficulties with the afore-mentioned terminal set.

While the first two requirements can be met by many different sets of terminals, the third and fourth constrain considerably the possible choice.

An example of a simple terminal set which satisfies these requirements is shown in Table 1, 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 center of the neighborhood). 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 has the following features:

- It includes only a few terminals.
- It captures both fine scale and broader scale information. Also, in conjunction with the shifting macros described in the next section it provides full information (see below).
- The moving averages `I_3x3`, `I_7x7`, `I_15x15` etc. can be computed very efficiently, in the initialization phase, with an algorithm requiring only 4 additions and 1 division per pixel. Also, they can be stored in appropriate arrays, so that their evaluation at runtime requires only a memory access.

Terminals	
Name	*Definition*
`random`	Random constant generator
`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`	$\frac{1}{225}\sum_{d_x=-7}^{7}\sum_{d_y=-7}^{7} I(x_c+d_x, y_c+d_y)$
$\vdots$	Moving averages on larger neighborhoods

Table 1: A terminal set for image analysis.

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	Evaluates a with $x_c = x_c + 1$
`Xcm1`	1	Evaluates a with $x_c = x_c - 1$
`Ycp1`	1	Evaluates a with $y_c = y_c + 1$
`Ycm1`	1	Evaluates a with $y_c = y_c - 1$

Table 2: A function set for image analysis.

3.2 Function set

In addition to the usual closure and sufficiency properties, efficiency requirements similar to those of terminal sets apply to function sets for image analysis.

While closure is easy to achieve, sufficiency cannot be ensured, especially in problems for which no exact solution (i.e. filter) is known. A necessary step towards sufficiency is to use functions which can extract all the information contained in the image, when this is not already made available by the terminal set.

However, in order to meet the efficiency requirements, in most cases it is necessary not to include functions which perform real 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 programs, between different parts of a single program and more importantly from pixel to pixel. As a result it is impossible to pre-compute and store the values to be returned by such functions. Thus, the evaluation of the fitness of each individual may require a computation load orders of magnitude bigger than if image-processing functions are not used.

A function set that meets this requirements is shown in Table 2, 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.[2] 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 operations 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 hamper this approach. While the second problem can be reduced considerably by sampling the images of the training set at, say $10^3 - 10^4$, random points and running the program only in those locations, the first problem remains as hand retouching

[2] It is well known in image processing, that by combining properly shifted and scaled box operators, any filter (including Gaussian filters, derivatives of Gaussian, Laplacian filters, etc.) can be approximated to the desired degree of accuracy.

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 the class to which 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. The search that GP has to do can be made considerably easier by using a wrapper for the filtering program, 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 [Poli and Valli, 1996] 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[3] 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, optimizing their sum does not necessarily mean to achieve an optimum sensitivity/specificity trade-off. On the contrary, experiments with this fitness function have shown that, in most cases, GP tend to discover algorithms which are either very sensitive (and poorly specific) or vice versa.[4]

Unfortunately, optimising the sensitivity/specificity trade-off is quite difficult as, 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 it 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).

[3]For the sake of simplicity in the following we will use the term "fitness" to mean "cost", i.e. a quantity to be minimised. A simple change of sign can transform the former into the latter.

[4]The fitness function $f = FP^2 + FN^2$ tends to behave slightly better but still does not give acceptable results when the number of pixels to be detected is much smaller than the pixels in the image.

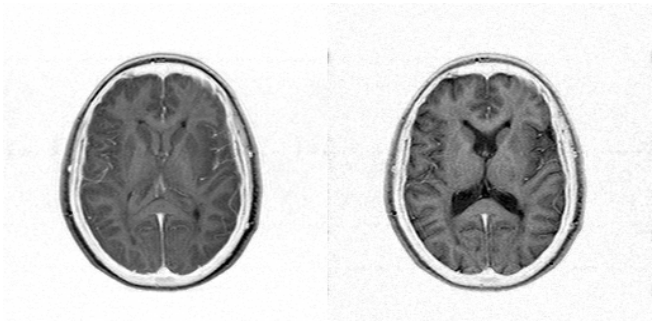


Figure 1: Two original MR images of the head.

In our experience, the fitness function that has given the best results is:

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

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 minimize the number of false detections. Instead, if FN/P is slightly bigger that α, $f \approx FN \exp(10\,(\frac{FN}{P} - \alpha))$, i.e. GP tries to minimize the number of misses.

4 Experimental Results

This section describes 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 small size images. On the contrary we decided to use real data from the medical domain and to face an extremely difficult problem: the segmentation of the brain in Magnetic Resonance (MR) images like the ones in Figure 1 (the images are maps of the intensity of two different physical properties of the same section of the head of a patient).

Pixel classification strategies based on neural networks or other statistical approaches have been used in the past for this task with limited success. The difficulty resides in the fact that some kinds of tissue in the brain present the same physical properties as some tissues in the skin. As a consequence, brain segmentation based on small neighborhoods is basically impossible [Coppini et al., 1992]. However, this does not mean that using large neighborhoods, the task becomes easy. To show this and also to provide a reference for comparison with GP, some experiments using neural networks in

conjunction with medium- and large-size neighborhoods have been performed.

The architecture of the nets is shown in Figure 2. They are fully connected nets including: 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 neighborhood, b) ten hidden neurons, c) one output neuron whose activation determines whether the pixel at the center of the retinas belongs to the brain or not. The networks were trained with the backpropagation algorithm to segment the brain in the images in Figure 1.

Two different sizes for the retinas have been tried, 5×5 and 15×15, corresponding to medium and coarse scale (for 256×256 pixel wide images). Figures 3(a)–(b) show the output produced by the nets. Despite 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) and the presence of a *single image-pair in the training set*, the segmentation of the training set itself shown in Figures 3(c)–(d) looks very disappointing for both nets.[5] In particular, it can be observed that the small retinas of the first net provide insufficient information to avoid FPs in the skin region, while the large retinas of the second net lead to FNs due to the inability to correctly classify the brain edges. However, it should be noted that the poor quality of these results is subjective (i.e. it is the result of our demanding visual system) as the misclassification rates were relatively small: 3.90% for the 5×5 net and 1.84% for the 15×15 net. This is phenomenon is quite common in image analysis, where small variations in decimal figures can make big changes in the perceived quality of the results. For this reason image segmentation is in general such a difficult problem.

Figure 4 shows the results obtained in the same task by the following program with fitness $f = 16.6004$ discovered by GP after approximately 40,000 fitness evaluations:[6]

```
(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)
```

[5] Segmentation was obtained by thresholding the output of the nets. The threshold were chosen so as to minimise the difference between the segmented images and the training ones.

[6] The terminals with prefixes `MR0` and `MR1` represent the moving averages of the first and second image in Figure 1, respectively.

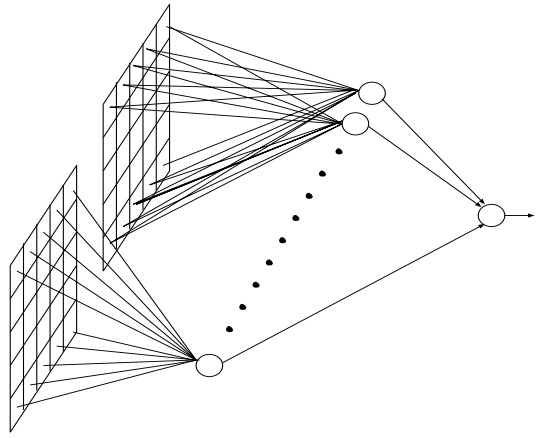


Figure 2: Architecture of NN used for MR image segmentation (for clarity some neurons and connections have not been drawn).

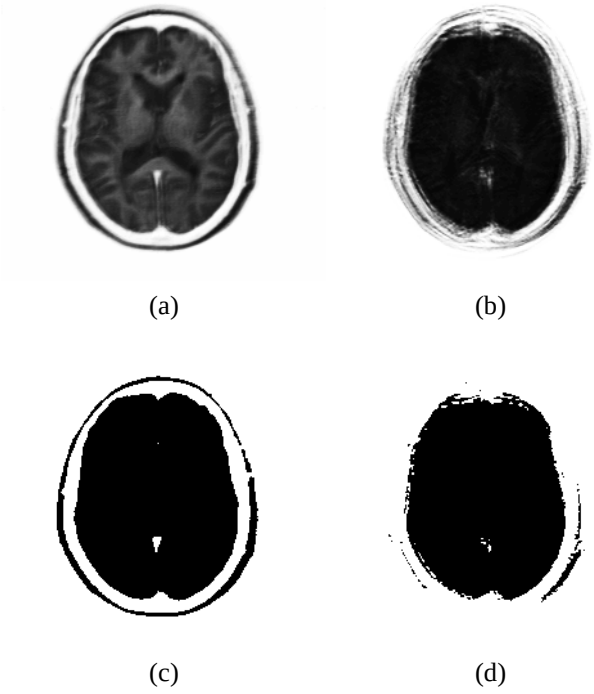
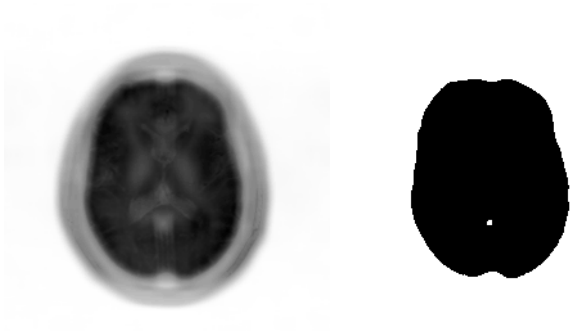


Figure 3: Segmentation of the images in Figure 1 produced by neural nets: (a) output of a net with 5×5 input retinas, (b) output of a net with 15×15 input retinas, (c) segmentation (via optimum thresholding) of the image in (a), (d) segmentation (via optimum thresholding) of the image in (b).



(a) (b)

Figure 4: Segmentation of the images in Figure 1 produced with GP: (a) output of the filtering program, (b) segmentation produced by the wrapper applied to the image in (a).

Although the understandability of this program is minimum (comparable with that of a neural net), the segmentation it produces looks considerably better. GP performances are also objectively better as the misclassification rate is only 1.56%. To our knowledge this result is very difficult to achieve with other techniques (even on a single image) unless anatomical knowledge is explicitly used.

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 some preliminary experiments with medical images have enabled GP to outperform neural nets.

The research on genetic programming for image analysis is hampered by the tremendous demand of computational resources involved in fitness evaluation. This makes it very difficult if not impossible to produce the good result-documentation typical of genetic programming literature (e.g. plots of the probability of reaching a certain fitness value vs. generation number, the minimum computation effort required, generalisation capabilities of the filters discovered, etc.). However, the impressive performance shown by GP compared with NNs in the experimental results reported in this paper seems to suggest that there is a huge space of image analysis tools much more powerful than the ones used in image processing nowadays waiting to be discovered/designed. Whether genetic programming will be the ultimate tool to do that is certainly an open question to be addressed with future research.

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

[Andre, 1994] Andre, D. (1994). Automatically defined features: the simulataneous evolution of 2-dimensional feature detectors and an algorithm for using them. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 23, pages 477–494. MIT Press.

[Ballard and Brown, 1982] Ballard, D. and Brown, C. (1982). *Computer Vision*. Prentice-Hall, Englewood Cliff, NJ.

[Breunig, 1995] Breunig, M. M. (1995). Location independent pattern recognition using genetic programming. In Koza, J. R., editor, *Genetic Algorithms and Genetic Programming at Stanford 1995*, pages 29–38. Stanford Bookstore, Stanford University.

[Coppini et al., 1992] Coppini, G., Poli, R., Rucci, M., and Valli, G. (1992). A neural network architecture for understanding 3D scenes in medical imaging. *Computer and Biomedical Research*, 25:569–585.

[K. E. Kinnear, Jr., 1994] K. E. Kinnear, Jr., editor (1994). *Advances in Genetic Programming*. MIT Press.

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

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

[Poli and Valli, 1996] Poli, R. and Valli, G. (1996). Hopfield neural nets for the optimum segmentation of medical images. In Fiesler, E. and Beale, R., editors, *Handbook of Neural Computation*, chapter G.5.5. Oxford University Press. (in press).

[Riolo and Line, 1995] Riolo, R. L. and Line, M. P. (1995). Automatic discovery of classification and estimation algorithms for earth-observation satellite imagery. In Siegel, E. S. and Koza, J. R., editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 73–77, MIT, Cambridge, MA, USA. AAAI.

[Tackett, 1993] Tackett, W. A. (1993). Genetic programming for feature discovery and image discrimination. In *International Conference on Genetic Algorithms*.

[Teller and Veloso, 1995] Teller, A. and Veloso, M. (1995). A controlled experiment: Evolution for learning difficult image classification. In *Seventh Portuguese Conference On Artificial Intelligence*. Springer-Verlag.