

Morphological Algorithm Design for Binary Images using Genetic Programming

MARCOS I. QUINTANA

CIEP, Facultad de Ingeniería, Universidad Autónoma de San Luis Potosí, Av. Dr. Manuel Nava 8, C.P. 78290, San Luis Potosí, S.L.P., México

m.i.quintana@uaslp.mx

RICCARDO POLI

Department Computer Science, University of Essex, Colchester, CO4 3SQ, UK

rpoli@essex.ac.uk

ELA CLARIDGE

School of Computer Science, The University of Birmingham, Birmingham, B15 2TT, UK

e.claridge@cs.bham.ac.uk

Abstract. This paper presents a Genetic Programming (GP) approach to the design of Mathematical Morphology (MM) algorithms for binary images. The algorithms are constructed using logic operators and the basic MM operators, i.e. erosion and dilation, with a variety of structuring elements. GP is used to evolve MM algorithms that convert a binary image into another containing just a particular feature of interest. In the study we have tested three fitness functions, training sets with different numbers of elements, training images of different sizes, and 7 different features in two different kinds of applications. The results obtained show that it is possible to evolve good MM algorithms using GP.

Keywords: Genetic Programming, Mathematical Morphology, Image Analysis.

1. Introduction

Many image processing tasks can simply be seen as functional transformations T of an original image o (or a set of images) into an ideal goal image g (or set of images) [22]. Figure 1 exemplifies this for the task of extracting/detecting those image pixels of a binary image that represent a mountain. Naturally this transformation can easily be done by a human via manual editing. This suggests that there must exist an algorithm that implements T . A completely different question is, however, whether or not we can write down such an algorithm in a form that could be used by a computer.

Knowledge elicitation techniques, such as interviews, protocol analysis, etc., which have proven to be useful in other areas of artificial intelligence, are of a limited value in image analysis, in that vision is for humans a totally effortless automatic process involving the conscious mind only to limited extents [19, 5]. For this reason the design of image analysis algorithms is often a difficult task with no perfect solution, involving enormous amounts of trial and error. For example, despite all recent advances in image processing technology, it is effectively still impossible for a human to design algorithms capable of reliably solving problems such as the one sketched in Figure 1.

Mathematical Morphology (MM) is a powerful image processing technique based on set theory concepts. Serra introduced this technique to develop operators suitable for direct shape processing [34, 35]. Two main factors need to be considered when designing MM algorithms: the morphological filters (also called structuring elements or kernels) to be

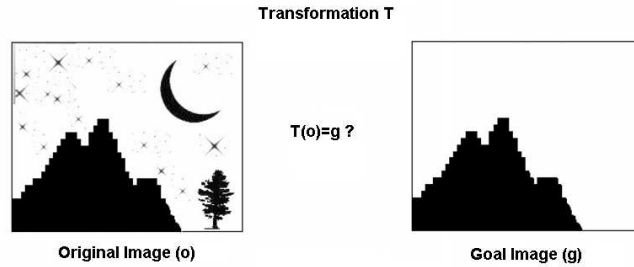


Figure 1. Example of image analysis transformation T from an original image (o) to a goal image (g).

used, and the morphological operators sequence. Morphological filters are an important class of non-linear digital signal processing filters, which have found a range of applications, giving excellent results in areas such as noise reduction, edge detection and object recognition [10, 8]. However, few design methods exist for morphological filters, the selection of the best type of filters to use and the order in which to apply them being very application-specific.

The number of possible structuring elements from which to choose is very large (for binary images there are 2^{n^2} structuring elements of size $n \times n$). The number of structuring elements combined with the possible choices of morphological operator sequences makes morphological filter design something of a *black art*. This is why a few researchers have attempted to (at least partially) automatise MM algorithm design [6, 33, 42]. Some of that work has used Genetic Algorithms (GAs) [15] for this purpose. However, as we will discuss in Section 3, to the best of our knowledge previous research on this topic has been limited to the optimisation of a single filter for gray-level images, or to filter sequences using only a reduced number of structuring elements or only a restricted set of morphologic operators.

The process of assembling structuring elements and morphological operators into a filter is effectively very reminiscent of the process of putting together instructions to form a computer program. The main difference here is that, in MM, instructions are drawn from a really large instruction set, and, unlike ordinary computer programming, it is really difficult to predict the contributions that each instruction will make towards the goal image.

The general difficulty of image analysis problems, the unavailability of explicit domain knowledge on how to go about solving them, the difficulty of understanding the effects of different structuring elements and operator sequences in MM, and the large size of the MM search space suggest that MM is an ideal application area for Genetic Programming (GP) [17, 3, 18]. Indeed, GP has provided excellent results in programming environments, such as parallel algorithms [32, 23] and quantum algorithm [36], where human programmers struggle. Furthermore, GP has been applied successfully to solve other difficult problems in the image analysis domain such as classification [40, 37, 46], detection [38, 2, 43, 16, 31], recognition [41, 1, 47], segmentation [29], analysis [22, 24], compression [21], feature construction [30] and stereo vision [11]. However, to date very limited

work has been done on using GP in the important area of mathematical morphology algorithm design.

In this paper we will start filling this gap and we propose to use GP to explore the space of possible MM algorithms and automatically induce programs that, via sequences of MM operations, can extract features of interest in binary digital images.¹ GP has the potential to outperform previous GA-based approaches because it explores a bigger space of variable length solutions. In addition, GP is not limited by human biases in its choice of operators and kernels (e.g., it can easily use both regular and irregular structuring elements of mixed sizes), and, so, in principle, it might eventually even do better than humans, at least in certain specific domains of image analysis applications.

The paper is organised as follows. In Section 2 a brief introduction to MM operators is provided. In Section 3 we review previous work on using evolutionary algorithms for morphological image analysis. We describe our GP approach to morphological algorithm design, including results and analysis, in Section 4. Finally conclusions and suggestions for further research are presented in Section 5.

2. Mathematical Morphology

MM is a relatively separate part of image analysis. Whereas many non-morphological approaches to image analysis are based on the concept of point-spread-function and on linear transformations such as convolution, MM is based on geometry and shape: morphological operations simplify images, and preserve the main shape characteristics of objects. MM is basically a tool for extracting image components, such as boundaries and skeletons, that are useful in the representation of image regions. Morphological techniques, such as morphological filtering, thinning and pruning, are also used for pre- or post-processing.

The language of MM is *set theory*. Thanks to this, morphology offers a unified and powerful approach to numerous image processing problems. Sets in MM represent the objects in an image. For example, in Figure 2 the set of all black pixels (which corresponds to the 1's in the binary matrix representation on the left) is a complete description of the image on the right.

In binary images, the elements of the sets in question are members of the 2-D integer space \mathbb{Z}^2 . Each element of a set is a tuple (x, y) representing the coordinates of a black pixel in the image. For example, the set of pairs

$$\{(5,5), (5,6), (5,7), (5,8), (5,9), (5,10), (6,5), (6,6), (6,7), (6,8), (6,9), (6,10), \\ (7,5), (7,6), (7,7), (7,8), (7,9), (7,10), (8,5), (8,6), (8,7), (8,8), (8,9), (8,10), \\ (9,5), (9,6), (9,7), (9,8), (9,9), (9,10), (10,5), (10,6), (10,7), (10,8), (10,9), (10,10)\}$$

represents the black square in Figure 2.

Gray-scale digital images can be represented as sets whose components are in \mathbb{Z}^3 . Each element of a set is a triplet (x, y, z) representing the coordinates of a pixel and its gray level. Sets in higher-dimensional spaces can represent other image attributes, such as colour and time varying components.

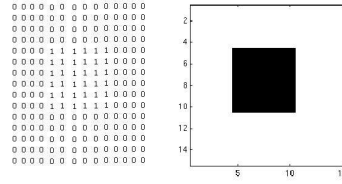


Figure 2. Binary image as the set of all black pixels.

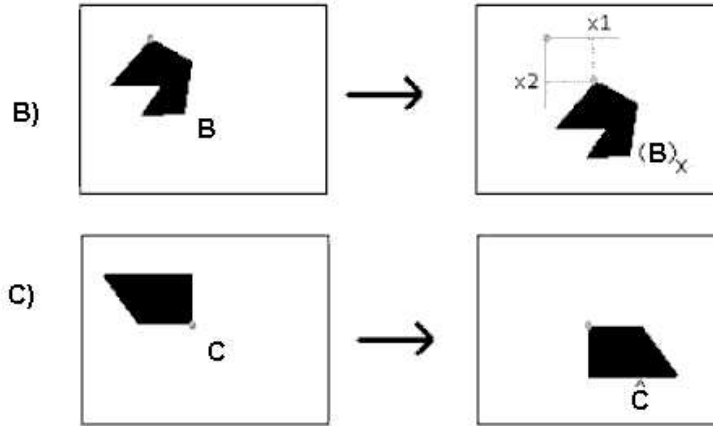


Figure 3. Translation (B) and Reflection (C).

2.1. Basic Morphological Operations

Dilation and erosion are the two fundamental morphological operations in MM. Most of the morphological algorithms developed by human experts make heavy use of these operations.

Dilation and erosion are based on two set operations, *translation* and *reflection*. The translation of set B by point $x = (x_1, x_2)$, denoted as $(B)_x$, is defined as

$$(B)_x = \{w | w = b + x, \text{ for } b \in B\}, \quad (1)$$

while the reflection of set C , denoted as \hat{C} , can be defined as

$$\hat{C} = \{w | w = -c, \text{ for } c \in C\}. \quad (2)$$

Figure 3 shows examples of translation and reflection.

If A and K are sets of points in \mathbb{Z}^2 and \emptyset denotes the empty set, the *dilation* of A by K , $A \oplus K$, requires performing the reflection of K about its origin, then shifting this reflection

\widehat{K} by $z \in \mathbb{Z}^2$ to obtain $(\widehat{K})_z$ and finally calculating the set of all displacements z such that $(\widehat{K})_z$ and A overlap by at least one element. That is:

$$A \oplus K = \{z | (\widehat{K})_z \cap A \neq \emptyset\}. \quad (3)$$

Usually the set K is called *structuring element* (or *kernel*). The structuring element is often small and its shape directly modifies the shape of set A . The origin of the structuring element is usually the centre. Dilation with different structuring elements produces different forms of thickening in the shapes in an image.

The *erosion* of A by K , $A \ominus K$, is defined as the set of all points z such that \widehat{K} translated by z is contained in A . That is:

$$A \ominus K = \{z | (\widehat{K})_z \subseteq A\}. \quad (4)$$

Erosion with different structuring elements produces different forms of thinning of the shapes in an image. Equations 3 and 4 are not the only definitions for dilation and erosion, but they are usually preferred in practical implementations because of their analogy with the operation of convolution in linear filtering.

Many algorithms may be constructed using these two basic operations. Gonzalez and Woods [10] and Dougherty and Latufo [8] present several examples where morphological operations are used for image pre-processing (noise filtering, shape simplification), enhancing object structure (skeletonising, thinning, thickening, extraction of convex hull, object making) and quantitative description of objects (area, perimeter, projections, Euler-Poincaré characteristics).

3. GAs for MM Filters and Algorithm Design

Morphological algorithm design requires that two sets of parameters are specified: the structuring elements and the sequence of morphological operations. As stated in the introduction, this is a difficult task. In this section we will review previous key work on the use of evolutionary algorithms applied to MM.

When using gray-level images, filter design (that is, structuring element design) is an optimisation problem in itself. Harvey and Marshall [12] demonstrated how simple GAs can be employed to search for (gray-level) morphological filters which provide good performance in a given image processing task. The authors did not limit themselves to evolving only the structuring element for one morphological operation. They also extended the search to sequences of four morphological operations. This work was a first step in the direction of using an evolutionary approach for morphological algorithm design. The search was limited to fixed-length sequences of morphological operators. However, a `do-nothing` operator was also available so as to effectively allow the evolution of variable-size (albeit, very short) sequences of operations. Interestingly, the `do-nothing` operator was reported to appear quite frequently in the chromosomes. Arguably, this suggests that the exploration of a variable length representations, such as those used in GP, would prove very beneficial.

In an attempt to automate morphological filter design for target detection in grey-scale images, Nong *et al.* [20] used chromosomes with a fixed number of components, and

the pixel-wise Square Error as their fitness function, proposing also new crossover and mutation operators to improve convergence.

Yu *et al.* [45] proposed a simple approach to image segmentation using GAs in conjunction with morphological operations. The images considered were composed of objects with a constant intensity embedded in a homogeneous background. The images were corrupted by additive white Gaussian noise. Segmentation was performed on 16×16 non-overlapping sub-images. The segmented sub-images were then combined to obtain the segmentation of the entire image. The fitness function measured the similarity between the output produced by an individual and the original noisy image. It included a penalty function which was introduced in order to reduce high frequency noise in segmented results. The authors concluded that without a priori knowledge of the object shape, it is difficult to determine the optimum size of the structuring element.

Yoda *et al.* [44] went into a very interesting research direction when they explored the possibility of obtaining MM algorithms (they called them *procedures*) for binary images by means of GAs. Even though they restricted the search to fixed-length chromosomes and limited the structuring elements to four different regular structures, they showed that the proposed method could obtain good solutions, although the computational effort required increased sharply with the complexity of the image-analysis task to perform.

Two GP approaches to image analysis are of relevance for the work reported in this paper. The first is the approach used in GENIE, an evolutionary system for finding features of interest in multi-spectral remotely-sensed images [13, 14]. In this system a set of morphological operators was provided as part of a larger set of image processing primitives. The sizes and shapes of the structuring elements used by these operators was restricted to a pre-determined set of classical primitive shapes. So, although in this work the resulting programs made some use of MM operators and were of variable size, the algorithms evolved used only a small set of regular kernels.

The second relevant GP approach is the work by Rosin and Hervás [31] who used GP to threshold the gray-level difference image of a landslide. Similarly to the work in [22], in this approach the original image was first preprocessed by using a variety of low-level image analysis operations, including Gaussian smoothing, morphological opening and closing with a small number of regular kernels, texture analysis, etc. GP was then used to evolve a function that, using arithmetic operators, maximum, minimum, absolute value and sigmoid could optimally combine the values stored in original image and the processed images. In order to process an entire image, the evolved program was applied to each pixel and its output was thresholded to obtain the pixel's classification. This work was limited to the processing of one single image of a particular landslide which was used both for training and for testing. In addition, the MM operators used, opening (an erosion followed by a dilation with the same kernel) and closing (a dilation followed by an erosion with the same kernel), are not atomic. So, most sequences of MM operations cannot be constructed using them. Finally, like the GENIE system, the algorithms evolved with this approach used only a small set of regular kernels, and there was no automatic kernel design. To summarise, although these two GP approaches to image analysis made some use of MM operations, they were not focused on MM algorithm design.

These ideas motivated our research and led us to explore the possibility of realising a GP system specialised to do MM algorithm design. We will describe our approach in the next section.

4. GP for Mathematical Morphology Algorithm Design

In our approach GP is used to find a sequence of morphological operators in the MM algorithm's search space capable of converting an input image into an output image containing only a particular feature of interest. The process includes several phases, some of which require human intervention or supervision. We describe them in the following subsections.

4.1. Training Sets

The first step is to build a set of examples (original/target image pairs) to provide guidance to GP as to what the desired feature extraction behaviour is. In order to test our approach to MM programming, we decided to use two datasets: one which is computer-generated and, therefore, fully under experimental control, and one more realistic dataset of musical sheets. These are described in the following two subsections.

4.1.1. Artificial datasets The use of computer-generated images makes it easy to create datasets of variable complexity without user intervention and time-consuming editing, which is very tedious for images containing a large number of features as our training images do.

Each of our artificial training sets is made up of two images, one source image and one target image both of size 640×480 . The images are synthetic and have been created using a purpose-built program. Each source image includes four different features, namely squares, circles, stars and rings. A target image is identical to the corresponding source image except that only one of the four features is present. In this way there are four types of target images. Each source image contains 50 features of each type to make a total of 200. The features are distributed randomly in the image and they may overlap, thereby making their detection more difficult. Target images contain 50 features each.

In order to test how well GP is able to cope with different resolutions, we created datasets with features of three different sizes. Figure 4 shows an example of a source image. An example of a corresponding target image, which contains only squares, is shown in Figure 5.

In addition to these training sets, we also generated 10 test image pairs for each resolution and feature. These test images were not part of the training sets.

4.1.2. Musical sheet datasets This second dataset includes images of musical sheets. The features we want to detect are noteheads, beams and staves. The dataset is rather difficult because the features are often overlapping and because of the similarity between beams and staves. Also, musical sheets contain other markings (e.g., clefs) that should not be picked up by the detectors for staves, noteheads and beams.

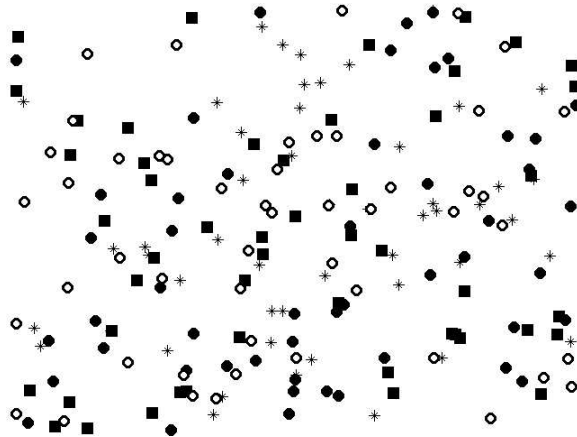


Figure 4. Original Image (size 3) containing four features: squares, circles, rings and stars.

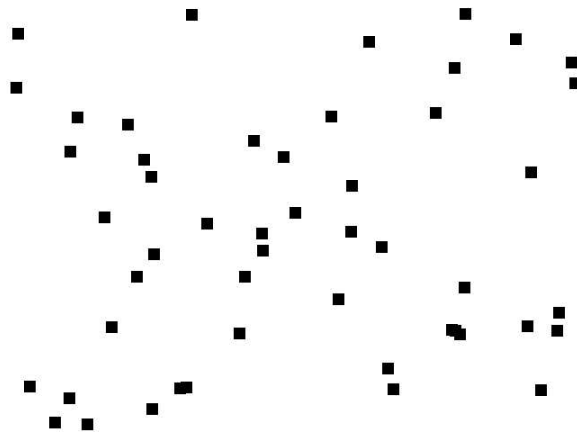


Figure 5. Expected squares from Figure 4

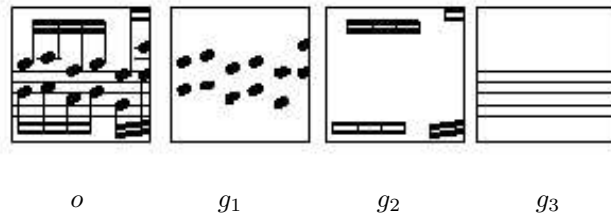


Figure 6. Examples of binary transformations. The original fragment of a music sheet (o) and three possible goals extracted manually: noteheads (g_1), beams (g_2) and staves (g_3).

To find out whether or not the size of the images in the training sets affects the quality of the results, we constructed (manually) training sets containing images of size 16×16 , 32×32 and 64×64 for each one of the different features to look for. An image belonging to the 64×64 training set with examples of target images is shown in Figure 6. To learn if the number of elements in the training set affected the quality of the results we used training sets of four different sizes: 1, 5, 15 and 25.

4.2. Fitness Functions

When applying a machine learning method or an optimisation algorithms to image analysis, typically one finds that the objective function can only provide a rough indication of the quality of solutions. This is a well known problem which we have encountered before both in GP-based [22] and neural-network-based [25] image analysis algorithms, and has been reported by others as well (e.g., [31]). No objective function and training set appear to distil either all the details of an image processing task or all perceptual qualities required in the output. In this, the human perceptual system is not only unbeatable but also a very severe critic.

To ensure that the results produced by GP are effective MM algorithms we need to attack the problem in two ways. Firstly, we need to test alternative fitness functions to identify which one provides the best indicator for a particular class of image analysis tasks. Secondly, we need to verify, a posteriori, with extensive testing and *visual* analysis, that the outcome (a MM algorithm) is indeed an acceptable solution to a given problem.

All our experiments were performed using three different fitness functions which evaluate the similarity between images. These were used to compare the outputs produced by each GP program with the set of desired outputs for all the images in the training set.

Our first fitness function is the one proposed by Yoda *et al.* [44], who defined the objective fitness function f_A known as *similarity* ($0 \leq f_A \leq 1$) as the normalised correlation coefficient between a goal and a processed image

$$f_A = \frac{(f \cdot g)}{\sqrt{(f \cdot f)}\sqrt{(g \cdot g)}} \quad (5)$$

where f and g are two digital images of size $N \times N$ and

$$(f \cdot g) = \frac{1}{N} \cdot \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^N f(i, j) \cdot g(i, j). \quad (6)$$

The function f_A intends to maximise the number of matching *black* pixels in the images f and g . Since in a binary image black pixels represent objects, this seems a reasonable choice of fitness function. The optimum is $f_A = 1$ when all the pixels match. The worst case is $f_A = 0$ when none of the pixels match.

Our second fitness function, f_B ($0 \leq f_B \leq 1$), is related to the *trade-off* between sensitivity and specificity needed in detection algorithms. Before we introduce it formally we need to define some more elementary notions.

In our experiments the goal is to convert the original image, o , into the goal image, g . Let (x_i, y_i) be the coordinates of a pixel. If $g(x_i, y_i) = 1$ then we say that we have a *true positive* if $o(x_i, y_i) = 1$. If, instead, $o(x_i, y_i) = 0$ we have a *false positive*. If $g(x_i, y_i) = 0$ then we have a *true negative* if $o(x_i, y_i) = 0$, and a *false negative* if $o(x_i, y_i) = 1$.

We can now define the notions of *sensitivity* (SV) and *specificity* (SP):

$$SV = \frac{TP}{TP + FN} \quad (7)$$

$$SP = \frac{TN}{TN + FP} \quad (8)$$

where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives and FN is the number of false negatives [9]. (These true/false positive/negative numbers are obtained by integrating over every pair of (x_i, y_i) coordinates in the original image.)

We are now in a position to define our second objective function:

$$f_B = 1 - \frac{\sqrt{(1 - SP)^2 + (1 - SV)^2}}{\sqrt{2}}. \quad (9)$$

Unlike f_A , the function f_B intends to maximise both specificity (the ability to detect true negatives accurately) and sensitivity (the ability to detect true positives accurately). The optimal value is $f_B = 1$ when $SP = SV = 1$; the closer both SP and SV are to 1 the closer f_B will be to 1; the worst case is $f_B = 0$ when $SP = SV = 0$.

Taking inspiration from the fitness function in [22], which was specifically developed for segmentation tasks rather than detection tasks, we also define a third fitness function. This is effectively a modification of function f_B where a variable α (0..1) allows tuning the bias of the search depending on whether we are looking for a program with high sensitivity (high α value) or with high specificity (low α value). The function is defined as:

$$f_C = c_1 - c_2 \times \frac{\sqrt{(1 - (SP \times (1 - \alpha)))^2 + (1 - (SV \times \alpha))^2}}{\sqrt{2}}, \quad (10)$$

where c_1 and c_2 are appropriate normalisation constants.

As we mentioned above, the best evolved sequences need also be analysed visually to decide whether or not their high numerical fitness value really corresponds to high quality of results. If it is, the problem is solved. Otherwise one needs to run the algorithm again.

4.3. Primitive Set

4.3.1. Structuring Elements In order to use GP to evolve MM algorithms, it is necessary to define the type, size and number of structuring elements to be made available to GP.

When an MM algorithm is designed manually, the programmer usually chooses a small number of regular structuring elements to be used in MM operations. There is no reason for choosing a *regular* structuring element except that humans are more likely to understand regularities such as squares, lines, triangles, etc. rather than irregular structures. However, the MM algorithm’s search space does not have to be limited to regular structuring elements. There are many *irregular* structuring elements which could provide significant benefits for specific applications. We decided, therefore, to include also irregular structuring elements in the GP search space.

In order to explore the full space of morphologic operators of a given size, in the experiments with the artificial dataset we provided GP with all structuring elements of sizes 3×3 . There are $2^{3^2} = 512$ of them, many of which are of irregular shape. We identified these by a number in the range 0..511. This is the decimal representation of the binary number obtained by reading out the bits in the kernel row by row and column by column.

For example kernel 8 (binary 000001000) corresponds to the structuring element $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$.

In the experiments with musical sheets, we anticipated the need to use structuring elements of a variety of sizes and used structuring elements of sizes 3×3 , 5×5 and 7×7 , examples of which are shown in Figure 7. Since the resulting set includes approximately 5.6×10^{14} different structuring elements, we could not allow them all to be used in the primitive set. Therefore, we provided a biased subset to GP by including 11 regular and 11 irregular (and randomly chosen) structuring elements of each size, for a total of 66 structuring elements. We identified these using the following code:

$$yz[w] \tag{11}$$

where $y \in \{R, I\}$ represents the type of structuring element selected (regular and irregular), $z \in \{3, 5, 7\}$ represents the size of structuring element and $w \in \{1..11\}$ represents the structuring element index. For example, the primitive $R3[7]$ represents the seventh regular structuring element of size 3×3 .

4.3.2. Terminal Set Most of the “action” in our approach takes place in the terminal set, in that, unlike most other GP approaches, our terminals are not input variables, but primitives with side effects. These primitives act on a memory structure that initially contains the original image. At the end of program execution, the (transformed) image in the memory structure is read out and is returned as the output of the program.

The most important operators are erosion and dilation, which we represent as $e(k)$ or $d(k)$, respectively, where k is a kernel. Depending on dataset, k is either an integer in 0..511 or one of the more complex operands in Equation 11. For example, the terminal $e(R3[7])$ represents erosion using the seventh regular structuring element of size 3×3 , while $d(7)$ represents dilation with a 3×3 structuring element containing a horizontal line.

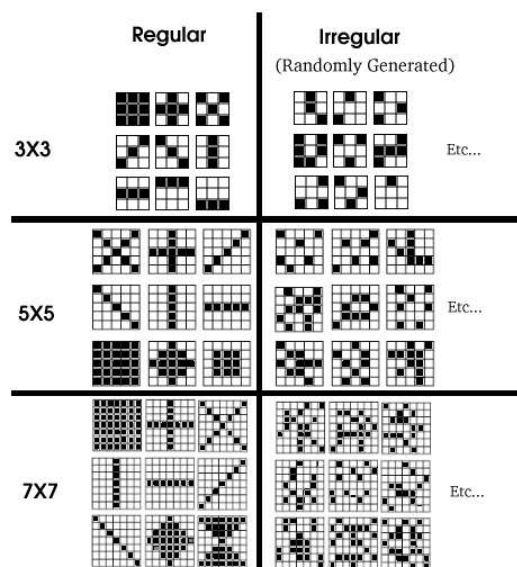


Figure 7. Examples of regular and irregular structuring elements of various sizes.

Table 1. Terminals and function used in the approach.

Name	Arity	Description
&	0	\cup , Union, AND
	0	\cap , Intersection, OR
\wedge	0	Exclusive-or, XOR
\sim	0	Negation, NOT
s	0	Store operator
$e(k)$	0	Erosion using kernel k
$d(k)$	0	Dilation using kernel k
EVAL1	1	Tree generator
EVAL2	2	Tree generator

In addition to these two MM operators, we also included logic operators in the terminal set (see Table 1). The reason for this is that when processing any realistically complicated images it is very rare to use MM operators alone. Most logic operations require two arguments. In our implementation the first argument is always the image currently being processed and the second input is provided by recalling the contents of a memory store, where the algorithm can save previous intermediate results via a *store* operator s . When this is executed, the current intermediate result of the computation overwrites the image currently saved in memory.

4.3.3. Function Set We use a function set including only two functions, EVAL1 and EVAL2 of arity 1 and 2, respectively, which simply execute their arguments. At fitness evaluation a GP tree is converted into a sequence of morphological/logic operators by reading out the leaves encountered in depth first traversals, one by one. The sequence of operators is executed from left to right. This MM algorithm is applied to the training set in order to evaluate the fitness of the corresponding tree. Before the sequence starts, we perform a store operation to save the original image into the memory store, thereby providing a meaningful second argument also for binary logic operations which are not explicitly preceded by a *store* operation.

For example, the program tree (EVAL2 \sim (EVAL2 (EVAL2 $d(238)$ (EVAL1 $\&$)) (EVAL2 (EVAL2 $d(462)$ s) (EVAL2 $e(185)$ \wedge))) is first converted into the sequence $\sim d(238)\&d(462)se(185)\wedge$. This is effectively our MM algorithm. When executed from left to right it leads to the following sequence of operations: the storage of the input image in the memory (default initial operation), followed by a logic NOT, followed by a dilation using kernel 238, followed by a logic AND, followed by a dilation using kernel 426, followed by the storage of the current image in memory, followed by an erosion using kernel 185 and finally a logic XOR of the current image with the image in memory.

Note that thanks to the use of a GP tree representation, the resulting MM algorithms are sequences of variable size.

4.4. GP Setup

All the experiments were implemented using *Lilgp* [48]. In most of our experiments we used a Linux cluster with one master node (CPU dual Intel Xeon 2GHz, 2GB Memory) and 22 client nodes (CPU Dual Athlon MP 1900+, 1.6GHz, 1GB Memory). Despite providing these significant resources to GP, because of the heavy computational load involved in fitness evaluation in image analysis applications, we had to make some compromises in our experimentation, performing either large sets of runs but with small populations of 50 individuals run for 100 generations, or fewer bigger runs with populations of 500 individuals run for 500 generations.²

The GP parameters used in our experiments are similar to those suggested by Koza [17]. Specifically, a 0.9 crossover rate and 0.1–0.2 mutation rate were used. Mutation was based on the ramped-half-and-half initialisation method, which was also used to initialise the population. The method was slightly modified so as to allow biasing the choice of terminals. In particular, a parameter, β , was introduced which allows to change the balance between logic and morphological terminals. If $\beta = 0.5$ then logic and morphological operators are selected with equal probability. A higher value of β biases the search toward logic operators and vice versa. The reason for including this parameter is that logic and morphological operators modify the image in a very different way. Whilst logic operators act equally on every pixel of an image, morphological operators perform local modifications to the shapes in the image, having an effect particularly at their edges. These two different types of behaviours may be appropriate for different applications. By tuning β we are able to match the primitive set to the application.

Table 2. Performance of the GP MM approach on the artificial dataset.

Feature Type	Statistics	Feature Size		
		1	2	3
Squares	Mean	0.870	0.864	0.843
	StdDev	0.087	0.079	0.076
	Minimum	0.761	0.781	0.748
	Maximum	0.944	0.933	0.930
Circles	Mean	0.868	0.807	0.870
	StdDev	0.074	0.118	0.059
	Minimum	0.746	0.642	0.811
	Maximum	0.947	0.935	0.935
Rings	Mean	0.906	0.900	0.896
	StdDev	0.043	0.047	0.048
	Minimum	0.862	0.850	0.845
	Maximum	0.948	0.944	0.939
Stars	Mean	0.922	0.921	0.857
	StdDev	0.028	0.025	0.105
	Minimum	0.893	0.895	0.641
	Maximum	0.951	0.948	0.943

4.5. Results and Analysis: Artificial Dataset

After a set of preliminary tests (not reported) aimed at determining which of the three different fitness functions described in Section 4.2 was best suited for the artificial dataset (see Section 4.1.1), we settled for function f_C (Equation 10) with parameter $\alpha = 0.9$ (setting α to this high value consistently led to better results than when α was low). Furthermore, we decided to allow the use of logic operators and memory by setting $\beta = 0.7$, although performance was not significantly affected by the precise value of this parameter.

Subsequently, we performed a total of 108 GP “production” runs using the artificial dataset, one for each of the 3 different feature sizes, 4 different features and 9 different random seeds. Each run took between 4 and 7 hours to complete. The remaining parameters of the runs were: a crossover rate of 0.9, a mutation rate of 0.2, a population size of 500 individuals run for 500 generations.

Table 2 provides the statistics for the best fitness values recorded in the runs. As can be seen, for all features, average fitness is relatively high, with at least one program at the end of 9 runs achieving a fitness value over 0.93. There seems to be no significant influence of feature type and sizes, which encouragingly suggests a generality of the approach.

Let us try to understand the meaning of the high fitness values reached in the experiments. In particular, to focus our discussion let us see what a fitness of 0.94 means in terms of sensitivity and specificity for a detector. By setting $f_C = 0.94$ in Equation 10 and solving for SV , it is easy to show that sensitivity and specificity are related by the following parabolic equation

$$SV \approx \frac{1 - \sqrt{0.876 - (1 - SP \times 0.1)^2}}{0.9}, \quad (12)$$

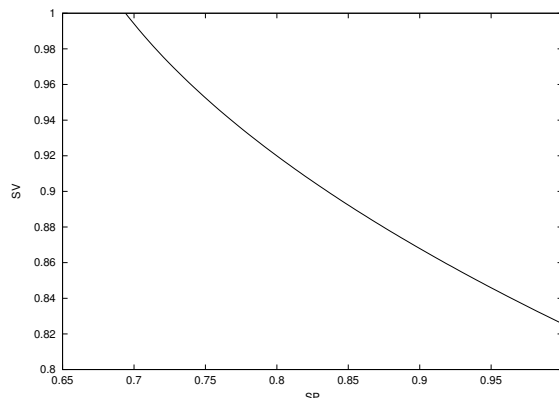


Figure 8. Sensitivity/specificity tradeoff for the MM algorithms with $f_C = 0.94$.

which is plotted in Figure 8 (similar relationships are obtained for different values of f_C). As illustrated in the figure, a GP program can achieve the fairly high fitness value of 0.94 with a wide range of sensitivities and specificities, some of which may be visually unacceptable. For example, in Figure 9 we can see the output produced by an evolved ring detector, which was trained to detect rings in the image in Figure 4 and attained a high fitness. Visual analysis reveals that while evolving this detector, GP generated an algorithm of high sensitivity, as specified by the parameter α (almost all pixels belonging to actual rings have been correctly detected). However, it is clear that this algorithm had too low a specificity.

These experiments suggest that the fitness value alone is not a very good indicator of *visual* quality of the pictorial result. Higher fitness is likely to correspond to better detectors, but to judge the true quality of the end-of-run detectors evolved it is essential to visually analyse their output.

Fundamentally, the reason for this is that in image analysis it is very difficult to clearly formalise the objective of an image processing task in terms of an objective function. It is then not surprising that, given such poor guidance, a machine learning method will have problems achieving exactly what was needed. This is one of the reasons why researchers have started exploring the possibility of using *interactive* evolutionary systems in image analysis with good success [7, 24, 39].

4.6. Results and Analysis: Musical Sheet Dataset

Due to the heavy computation and memory loads in the set of experiments using musical sheets (see Section 4.1.2) we used smaller populations (50 individuals) and shorter runs (100 generations) than for the artificial datasets. In this way we were able to do many more runs.

As before, we did preliminary tests aimed at determining the suitability of the three fitness functions described in Section 4.2. We excluded function f_C , and used instead f_A

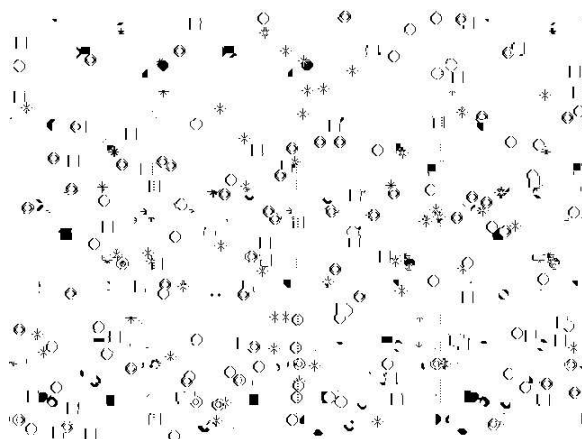


Figure 9. Ring detector. This is the final output provided by the ring-detecting program $\sim d(238)\&d(462)e(185) \wedge se(508)e(246)\wedge$ evolved by GP when applied to the image in Figure 4

and f_B (Equations 5 and 9, respectively). Furthermore, we decided to not allow the use of logic operators and memory by setting $\beta = 0$.

We did a total of 1512 GP runs combining the detection of 3 features (noteheads, beams and staves), 4 training set sizes (1, 5, 15 and 25), 2 fitness functions, 3 training-set image-sizes (16×16 , 32×32 and 64×64), 3 different combinations of structuring element types (Regular, Irregular and Mixed Regular and Irregular) and 7 different combinations of structuring elements sizes (3; 5; 7; 3 and 5; 3 and 7; 5 and 7; 3, 5 and 7).

The best and mean fitness values of f_A and f_B are shown in Tables 3–6. Analysing Table 3 we may observe that, in general, superior results are obtained when using larger structuring elements. The use of irregular kernels provides better performance than the use of regular kernels only (see Table 4). Also, on the basis of the results shown in Table 5, it appears to be preferable to use bigger training images. For instance, the 64×64 examples obtained better results than the others. The number of images used in the training set did not affect the final results much, with the exception, perhaps, of the training set of size 15, which was slightly better according to the results in Table 6.

We analysed visually the behaviour of the evolved MM algorithms. Although visual analysis cannot be exhaustive (we focused on those MM algorithms that had a high fitness value and those with interesting features) we observed that fitness f_B produced programs with consistently balanced behaviour, while f_A often had a tendency to be either overly sensitive or overly specific. Visual analysis also confirmed the aforementioned observation regarding Table 6.

Tables 7, 8 and 9 show the results of a detailed visual analysis of the best evolved algorithms over 10 test images not included in the training set. Unlike the small images used in the training set, the images in the testing set are of size 512×512 to verify the effectiveness of the programs when applied to bigger images including the same features. The first column in each table represents the image ID, the second column the number of features actually present in the image, the third column the number of features correctly detected

Table 3. Best and average numerical fitness values for Structuring Elements size

Structuring element size(s)	Noteheads				Beams				Staffs			
	f_A		f_B		f_A		f_B		f_A		f_B	
	best	mean	best	mean	best	mean	best	mean	best	mean	best	mean
3	0.701	0.612	0.643	0.559	0.809	0.535	0.644	0.454	0.618	0.541	0.604	0.521
5	0.735	0.646	0.653	0.580	0.825	0.547	0.655	0.501	0.660	0.584	0.625	0.551
7	0.748	0.649	0.672	0.579	0.833	0.566	0.682	0.512	0.683	0.599	0.911	0.589
3,5	0.735	0.651	0.656	0.580	0.648	0.473	0.825	0.552	0.660	0.582	0.625	0.547
3,7	0.747	0.657	0.662	0.561	0.833	0.563	0.658	0.504	0.683	0.596	0.636	0.510
5,7	0.750	0.653	0.672	0.570	0.833	0.560	0.682	0.495	0.683	0.601	0.911	0.567
3,5,7	0.701	0.614	0.637	0.542	0.809	0.506	0.644	0.469	0.618	0.540	0.593	0.492

Table 4. Best and average numerical fitness values for Structuring Elements type.

Structuring element type(s)	Noteheads				Beams				Staffs			
	f_A		f_B		f_A		f_B		f_A		f_B	
	best	mean	best	mean	best	mean	best	mean	best	mean	best	mean
Regular	0.747	0.623	0.655	0.528	0.832	0.526	0.641	0.448	0.678	0.573	0.623	0.508
Irregular	0.750	0.660	0.672	0.620	0.833	0.570	0.681	0.540	0.683	0.581	0.911	0.593
Both	0.747	0.638	0.661	0.555	0.832	0.545	0.645	0.473	0.678	0.578	0.625	0.517

Table 5. Best and average numerical fitness values for Image Size

Image size	Noteheads				Beams				Staffs			
	f_A		f_B		f_A		f_B		f_A		f_B	
	best	mean	best	mean	best	mean	best	mean	best	mean	best	mean
16 × 16	0.624	0.537	0.643	0.499	0.391	0.309	0.474	0.335	0.567	0.530	0.911	0.494
32 × 32	0.738	0.675	0.672	0.581	0.833	0.676	0.681	0.543	0.625	0.585	0.624	0.533
64 × 64	0.750	0.708	0.662	0.622	0.703	0.656	0.641	0.582	0.683	0.617	0.636	0.592

Table 6. Best and average numerical fitness values for Training Set size

Training set size	Noteheads				Beams				Staffs			
	f_A		f_B		f_A		f_B		f_A		f_B	
	best	mean	best	mean	best	mean	best	mean	best	mean	best	mean
1	0.730	0.593	0.658	0.594	0.665	0.543	0.675	0.430	0.683	0.566	0.775	0.584
5	0.750	0.648	0.662	0.576	0.712	0.554	0.641	0.508	0.637	0.576	0.842	0.521
15	0.738	0.664	0.672	0.603	0.833	0.604	0.681	0.537	0.648	0.583	0.731	0.574
25	0.732	0.656	0.658	0.594	0.798	0.600	0.665	0.543	0.643	0.585	0.775	0.580



Figure 10. Testing image not included in the training set

(true positives), the fourth column the number of features miss-detected (false negatives) and the fifth column the number of other regions present in the image (false positives). Note that we are evaluating regions not pixels.

GP easily found different algorithms to solve the *noteheads* extraction problem. Figure 10 shows an image not included in the training set and Figure 11 shows the result of an MM notehead-detector algorithm generated by GP. In Table 7 we may observe that the same algorithm applied to the test images obtained 100% sensitivity (having detected all true positives without false negatives) and high specificity (a small number of false positives).

The *beams* extraction problem is difficult. Beams can be mismatched with staves, noteheads or other features present in a musical sheet. In spite of that, some GP algorithms generated good approximations to the desired outcome. Figure 12 shows a GP-generated program applied to the image Figure 10. In table 8 we may observe a considerable number of miss-detections and a large number of other regions present. Most of these regions are very small (many including just 3 pixels or less), and the visual quality of the results is only weakly affected by these misclassifications.

Contrary to expectation, the *staves* extraction problem presented many difficulties for the GP approach proposed. The programs evolved tended to either produce completely white images or to mismatch the staves with other features. Table 9 shows a GP staffs-detector with 100% sensitivity but a high number of false positives (mostly due to beam-like structures). This is illustrated in Figure 13 where GP accurately finds the staves, but also includes most of the beams present in the test image.

Table 7. Visual analysis of the notehead detector $e(R3[3])e(R3[9])d(R3[10])e(I3[9])$ generated using GP

Test Image	Noteheads in image	True Positives	False Negatives	False Positives
1	190	190	0	7
2	131	131	0	7
3	166	166	0	12
4	160	160	0	15
5	198	198	0	15
6	87	87	0	3
7	99	99	0	7
8	128	128	0	12
9	161	161	0	17
10	171	171	0	15

Table 8. Visual analysis of the beam detector $e(R3[0])d(R3[3])e(R3[0])d(R3[8])$ generated using GP

Test Image	Beams in image	True Positives	False Negatives	False Positives
1	44	31	13	90+
2	28	18	10	80+
3	48	32	12	90+
4	39	25	14	70+
5	49	41	8	40+
6	24	21	3	30+
7	24	23	1	50+
8	36	28	8	50+
9	41	34	7	70+
10	39	32	7	70+

Table 9. Visual analysis of the staff detector $e(R5[9])e(R7[10])d(R7[10])$ generated using GP

Test Image	Staffs in image	True Positives	False Negatives	False Positives
1	40	40	0	50+
2	40	40	0	40+
3	35	35	0	40+
4	35	35	0	40+
5	35	35	0	40+
6	30	30	0	20+
7	34	34	0	20+
8	32	32	0	30+
9	35	35	0	40+
10	35	35	0	40+

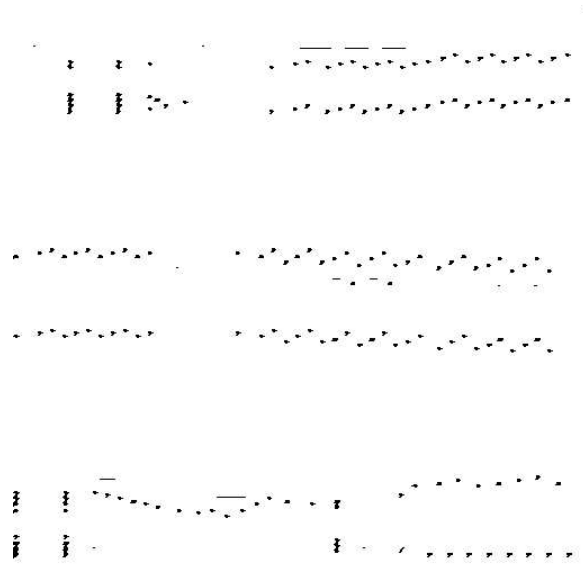


Figure 11. Output provided by the notehead-detecting MM program $e(R3[3])e(R3[9])d(R3[10])e(I3[9])$ evolved by GP when applied to the image in Figure 10.

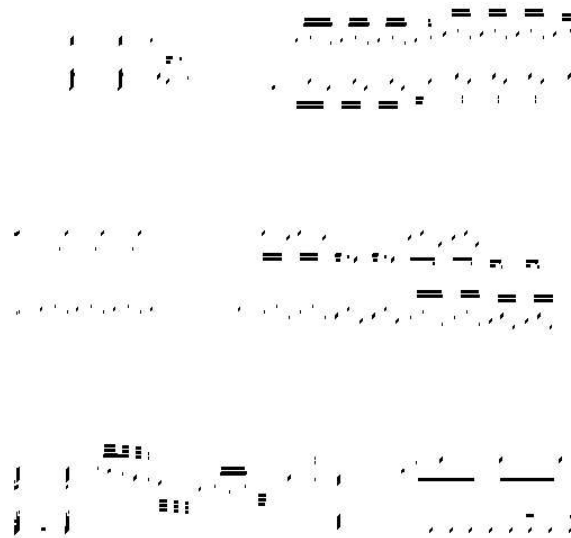


Figure 12. Output provided by the beam-detecting MM program $e(R3[0])d(R3[3])e(R3[0])d(R3[8])$ evolved by GP when applied to the image in Figure 10.

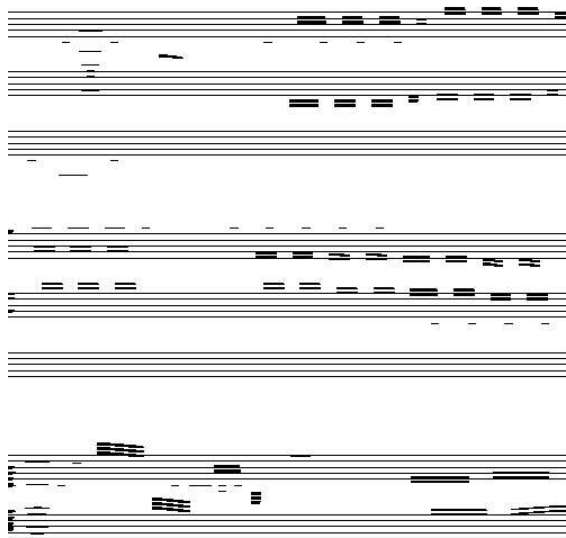


Figure 13. Output provided by the staff-detecting MM program $e(R5[9])e(R7[10])d(R7[10])$ evolved by GP when applied to the image in Figure 10.

5. Conclusions

In this paper we have presented an approach to morphological binary image analysis based on GP. We have applied this approach to the extraction of geometrical features in synthetic images as well as beams, staves and noteheads in musical score sheets obtaining promising results. In our approach, there are two phases where human intervention is required: the creation of a training set and the post-run evaluation of the results produced by GP. Apart from these two phases, the different MM feature extraction algorithms were obtained using GP in exactly the same way.

Since very little work has previously been done on the use GP in MM, the main objective of this paper was to charter this unexplored terrain rather than to develop any human-competitive MM algorithm. So, we have investigated the behaviour of different setups, studying three different fitness functions, a variety of regular and irregular structuring elements, three feature sizes, various image resolutions, several training set cardinalities, and different primitive sets.

As a result of this work we have identified a set of good GP setups for future MM applications and some promising directions for further research. For example, the use of irregular structuring elements has proved to be useful and so this should be explored in more depth, e.g. using larger kernel sizes. Bigger image sizes and larger training set sizes must be preferred, and we will therefore use these in future research. We have also seen that fitness function f_A has a tendency to maximise either sensitivity or specificity, which is undesirable. However, the other two fitness functions, f_B and f_C , hold much more

promise. Although in the present study the results of evolution had to be tested thoroughly and evaluated visually to determine if any of the best-of-run individuals provided results of sufficiently high quality for the tasks at hand, there is hope that further improvements to these two fitness functions will eventually lead to reliable, fully automated GP systems for MM and, possibly, other image analysis applications.

Acknowledgements

The authors would like to thank the anonymous referees, the associate editor in charge of this submission, the editor-in-chief, Tim Kovacs, Juan C. Cuevas and Hector Montes for useful comments on this paper. Marcos Quintana would like to thank the School of Computer Science of the University of Birmingham and SEP-Conacyt (Mexico) for financial support. Riccardo Poli would like to thank the members of the NEC (Natural and Evolutionary Computation) group at Essex for helpful comments and discussion.

Notes

1. We have presented some preliminary work on GP for MM in [26, 27, 28].
2. These types of compromises have been reported for a variety of different approaches in evolutionary image analysis. In [22], for example, *each run* took between one and two weeks of CPU time to complete on a fast 64-bit DEC Alpha, in [4] runs took around one week on a powerful 12 processor server, while in [31] runs took 4 days on Sun Ultra-5 workstation.

References

1. G. Adorni and S. Cagnoni. Design of explicitly or implicitly parallel low-resolution character recognition algorithms by means of genetic programming. In R. Roy, M. Köppen, S. Ovaska, T. Furuhashi, and F. Hoffmann, editors, *Soft Computing and Industry Recent Applications*, pages 387–398. Springer-Verlag, 10–24 Sept. 2001. Published 2002.
2. D. Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In K. E. Kinneer, editor, *Advances in Genetic Programming*, Complex Adaptive Systems, pages 477–494, Cambridge, 1994. MIT Press.
3. W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag, 1998.
4. S. Cagnoni, A. B. Dobrzeniecki, R. Poli, and J. C. Yanch. Genetic algorithm-based interactive segmentation of 3D medical images. *Image and Vision Computing*, 17:881–895, 1999.
5. C. Cinel, G. W. Humphreys, and R. Poli. Cross-modal illusory conjunctions between vision and touch. *Journal of Experimental Psychology: Human Perception and Performance*, 28(5):1243–1266, 2002.
6. T. Crimmins and W. Brown. Image algebra and automatic set recognition algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 21:60–69, 1985.
7. J. M. Daida, T. F. Bersano-Begey, S. J. Ross, and J. F. Vesecky. Computer-assisted design of image classification algorithms: Dynamic and static fitness evaluations in a scaffolded genetic programming environment. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 279–284, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
8. E. R. Dougherty and R. A. Latufo. *Hands-on Morphological Image Processing*, volume TT59. Spie Press, 2003.
9. J. P. Egan. *Signal Detection Theory and R.O.C. Analysis*. Academic Press, 1975.
10. R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, USA, 3rd edition, 1992.

11. C. T. M. Graae, P. Nordin, and M. Nordahl. Stereoscopic vision for a humanoid robot using genetic programming. In S. Cagnoni, R. Poli, G. D. Smith, D. Corne, M. Oates, E. Hart, P. L. Lanzi, E. J. Willem, Y. Li, B. Paechter, and T. C. Fogarty, editors, *Real-World Applications of Evolutionary Computing*, volume 1803 of *LNCS*, pages 12–21, Edinburgh, 17 Apr. 2000. Springer-Verlag.
12. N. Harvey and S. Marshall. The use of genetic algorithms in morphological filter design. *Signal Processing: Image Communication*, 8(1):55–72, 1996.
13. N. R. Harvey, S. Perkins, S. P. Brumby, J. Theiler, R. B. Porter, A. C. Young, A. K. Varghese, J. J. Szymanski, and J. J. Bloch. Finding golf courses: The ultra high tech approach. In S. C. et al., editor, *Real World Applications of Evolutionary Computing*, volume 1803 of *Lecture Notes in Computer Science*. Springer, 2000.
14. N. R. Harvey, J. Theiler, S. P. Brumby, S. Perkins, J. J. Szymanski, J. J. Bloch, R. B. Porter, M. Galassi, and A. C. Young. Comparison of GENIE and conventional supervised classifiers for multispectral image feature extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 40(2):393–404, Feb. 2002.
15. J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, USA, 1975.
16. D. Howard, S. C. Roberts, and R. Brankin. Target detection in imagery by genetic programming. *Advances in Engineering Software*, 30(5):303–311, 1999.
17. J. R. Koza. *Genetic programming: On the programming of computers by natural selection*. MIT Press, Cambridge, Mass., 1992.
18. W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.
19. D. Marr. *Vision*. W.H. Freeman & Co., New York, 1982.
20. Y. Nong, L. Yushu, and X. Qifu. The use of genetic algorithms in morphological filter design. In *5th International Conference on Signal Processing Proceedings WCCC-ICSP 2000*, volume 1, pages 476–479, 2000.
21. P. Nordin and W. Banzhaf. Programmatic compression of images and sound. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 345–350, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
22. R. Poli. Genetic programming for image analysis. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 363–368, Stanford University, CA, USA, 1996. MIT Press.
23. R. Poli. Parallel distributed genetic programming. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, Advanced Topics in Computer Science, chapter 27, pages 403–431. McGraw-Hill, Maidenhead, Berkshire, England, 1999.
24. R. Poli and S. Cagnoni. Genetic programming with user-driven selection: Experiments on the evolution of algorithms for image enhancement. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 269–277, Stanford University, CA, USA, 1997. Morgan Kaufmann.
25. R. Poli and G. Valli. Hopfield neural nets for the optimum segmentation of medical images. In E. Fiesler and R. Beale, editors, *Handbook of Neural Computation*, chapter G.5.5. Oxford University Press, 1997.
26. M. I. Quintana. *Genetic Programming applied to Morphological Image Processing*. PhD thesis, School of Computer Science, University of Birmingham, Birmingham, UK, 2005.
27. M. I. Quintana, R. Poli, and E. Claridge. Genetic programming for mathematical morphology algorithm design on binary images. In M. Sasikumar, J. Hedge, and M. Khavita, editors, *Artificial Intelligence, Proceedings of the International Conference KBCS-2002*, pages 161–171. Vikas, 2002.
28. M. I. Quintana, R. Poli, and E. Claridge. On two approaches to image processing algorithm design for binary images using GP. In G. R. Raidl, S. Cagnoni, J. J. R. Cardalda, D. W. Corne, J. Gottlieb, A. Guillot, E. Hart, C. G. Johnson, E. Marchiori, J.-A. Meyer, and M. Middendorf, editors, *Applications of Evolutionary Computing, EvoWorkshops2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, EvoSTIM*, volume 2611 of *LNCS*, pages 426–435, University of Essex, England, UK, 14–16 Apr. 2003. Springer-Verlag.
29. M. E. Roberts and E. Claridge. An artificially evolved vision system for segmenting skin lesion images. In R. E. Ellis and T. M. Peters, editors, *Proceedings of the 6th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 2878 of *LNCS*, pages 655–662, Montreal, Canada, Nov. 2003. Springer-Verlag.
30. M. E. Roberts and E. Claridge. A multistage approach to cooperatively coevolving feature construction and object detection. In F. Rothlauf, J. Branke, S. Cagnoni, D. W. Corne, R. Drechsler, Y. Jin, P. Machado,

- E. Marchiori, J. Romero, G. D. Smith, and G. Squillero, editors, *Applications of Evolutionary Computing, EvoWorkshops2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*, volume 3449 of LNCS, pages 396–406, Lausanne, Switzerland, 30 Mar.-1 Apr. 2005. Springer Verlag.
31. P. Rosin and J. Hervás. Image thresholding for landslide detection by genetic programming. In L. Bruzzone and P. Smiths, editors, *Analysis of multi-temporal remote sensing images*, pages 65–72. World Scientific, 2002.
 32. C. Ryan. *Automatic Re-engineering of Software Using Genetic Programming*, volume 2 of *Genetic Programming*. Kluwer Academic Publishers, 1 Nov. 1999.
 33. M. Schmitt. Mathematical morphology and artificial intelligence: An automatic programming system. *Signal Processing*, 16:389–401, 1989.
 34. J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.
 35. J. Serra. *Image Analysis and Mathematical Morphology: Volume 2*. Academic Press, 1989.
 36. L. Spector. *Automatic Quantum Computer Programming: A Genetic Programming Approach*, volume 7 of *Genetic Programming*. Kluwer Academic Publishers, Boston/Dordrecht/New York/London, June 2004.
 37. S. A. Stanhope and J. M. Daida. Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming*, volume 1447 of LNCS, pages 735–744, Mission Valley Marriott, San Diego, California, USA, 25-27 Mar. 1998. Springer-Verlag.
 38. W. A. Tackett. Genetic programming for feature discovery and image discrimination. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 303–309, University of Illinois at Urbana-Champaign, 17-21 1993. Morgan Kaufmann.
 39. H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, Sept. 2001. Invited Paper.
 40. A. Teller and M. Veloso. Algorithm evolution for face recognition: What makes a picture difficult. In *International Conference on Evolutionary Computation*, pages 608–613, Perth, Australia, 1995. IEEE Press.
 41. A. Teller and M. Veloso. PADO: Learning tree structured algorithms for orchestration into an object recognition system. Technical Report CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.
 42. R. C. Vogt. *Automatic Generation of Morphological Set Recognition Algorithms*. Springer-Verlag, New York, 1989.
 43. J. F. Winkeler and B. S. Manjunath. Genetic programming for object detection. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 330–335, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.
 44. I. Yoda, K. Yamamoto, and H. Yamada. Automatic acquisition of hierarchical mathematical morphology procedures by genetic algorithms. *Image and Vision Computing*, 17(10):749–760, 1999.
 45. M. Yu, N. Eua-anant, A. Saudagar, and L. Udpa. Genetic algorithm approach to image segmentation using morphological operations. In *International Conference on Image Processing*, volume 3, pages 775–779, 1998.
 46. M. Zhang, V. B. Ciesielski, and P. Andreae. A domain-independent window approach to multiclass object detection using genetic programming. *EURASIP Journal on Applied Signal Processing*, 2003(8):841–859, July 2003. Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis.
 47. M. Zhang and W. Smart. Learning weights in genetic programs using gradient descent for object recognition. In F. Rothlauf, J. Branke, S. Cagnoni, D. W. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. D. Smith, and G. Squillero, editors, *Applications of Evolutionary Computing, EvoWorkshops2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*, volume 3449 of LNCS, pages 408–417, Lausanne, Switzerland, 30 Mar.-1 Apr. 2005. Springer Verlag.
 48. D. Zongker, B. Punch, and B. Rand. Lilgp 1.01. user manual. Technical report, Michigan State University, 1996.