Genetic Programming applied to Morphological Image Processing

Marcos Iván Quintana Hernández

A thesis submitted

to The University of Birmingham

for the degree of Doctor of Philosophy

School of Computer Science

The University of Birmingham

Birmingham B15 2TT

United Kingdom

October 2004

Abstract

This thesis presents three approaches to the automatic design of algorithms for the processing of binary images based on the Genetic Programming (GP) paradigm. In the first approach the algorithms are designed using the basic Mathematical Morphology (MM) operators, i.e. erosion and dilation, with a variety of Structuring Elements (SEs). GP is used to design algorithms to convert a binary image into another containing just a particular characteristic of interest. In the study we have tested two similarity fitness functions, training sets with different numbers of elements and different sizes of the training images over three different objectives. The results of the first approach showed some success in the evolution of MM algorithms but also identified problems with the amount of computational resources the method required. The second approach uses Sub-Machine-Code GP (SMCGP) and bitwise operators as an attempt to speed-up the evolution of the algorithms and to make them both feasible and effective. The SMCGP approach was successful in the speeding up of the computation but it was not successful in improving the quality of the obtained algorithms. The third approach presents the combination of logical and morphological operators in an attempt to improve the quality of the automatically designed algorithms. The results obtained provide empirical evidence showing that the evolution of high quality MM algorithms using GP is possible and that this technique has a broad potential that should be explored further. This thesis includes an analysis of the potential of GP and other Machine Learning techniques for solving the general problem of Signal Understanding by means of exploring Mathematical Morphology.

Dedication

To my beloved son Andres, because there is no future without hope.

Evolution is not the survival of the fittest but the peaceful survival of everyone.

Acknowledgements

I would like to thank my external supervisor, Riccardo Poli, who accepted to guide my research from the beginning as an extension of his own discoveries. I am certainly impressed of his inexhaustible imagination and the amazing number of new ideas he have had whenever my research got stuck. It is not possible to express how much I value his efforts since he continued supervising my research even after he moved to the University of Essex and his commitments have raised exponentially month after month.

I am indebted to Ela Claridge, who kindly agreed to take me as her student when Riccardo had to leave. She has taught me how to always focus in the current stage of my research and to be more efficient as a research apprentice. Through her patience and kindness I have learnt to grow as a human being as well. Thank you very much Ela, you are an authentic example of a devoted scientist.

I appreciate very much the time that my examiners spent reading this thesis. I know both, Stefano Cagnoni and Xin Yao, are very busy people in the world of Evolutionary Computation; I am grateful to both of them.

I was fortunate to become a friend of Tim Kovacs before coming to Birmingham. He was my guardian angel for a long period of time. He taught me several things from improving my spelling and pronunciation to a better understanding of the world of Science. He was so generous to let me use the LaTeX files of his award-winning thesis [70] as a guide for this document.

I would like to thank the researchers who have been members of my thesis committee,

Julian Miller and Jon Rowe. I admire both of them and I am very grateful to have benefited from their comments and ideas to improve the quality of my research. My special gratitude to Aaron Sloman, Gustavo Olague, Manfred Kerber, Peter Hancox and Jeremy Wyatt for their interest on my thesis and their comments for sources to continue investigating. I am very grateful to the comments of the anonymous referees of my papers that help me improve my self-criticism skills. I have to point out the importance of the many little favours coming from Sammy Snow and Richard Pannel that made it possible for me to finish my studies.

Several PhD students contributed with ideas and suggestions to the improvement of this research. I won't mention all of them, but I must point out my recognition to the research carried out by Gavin Brown, John Woodward and Mark Roberts. I am particularly grateful to my Spanish speaking fellows Juan Carlos Cuevas, Hector Montes, Gregorio Sanchez, Felipe Orihuela, Raymundo Marcial and Max Salazar.

The biggest lessons in life are learnt when unexpected. I have to admit that the most insightful lesson I have learnt during these years abroad has been to forgive myself. I hope one day I can reach the forgiveness of those who I have hurt. Particularly I hope Maria Barilla knows that I am very sorry about her suffering.

Several wonderful human beings have been part of my life during these four years. I won't feel complete if I do not mention my gratitude to Qulsom Fazil who has been always there even in the most difficult situations. Lupita Rodriguez who has been a source of valuable motivation when I really needed it. Rosa Maria Chavez who is taking care of my most valuable treasure. I have to mention the support from Lucia and Said, my closest family. My special gratitude to Maria Mato, the most special friend I have ever had. Maria, you know that I will always have a small corner in my heart for you and I will always remember with a nostalgic smile our talks about saving the world from injustice and TMN.

I cannot express how much I value the financial support of several sources that made

possible my studies in the UK. My generous sponsors include the University of Birmingham, Conacyt-SEP (Mexico) and ORS.

Last but not least, I would like to thank my girlfriend Susanne Jacke. I am a very lucky man to find such a wonderful woman during such a difficult time in my life. She has been the most supportive and understanding person even though I have not been a person easy to deal with. I apologise for the little time I spent with her and I hereby promise we will have more time for us in the future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer programming is the art of telling a computer, step by step, how to solve a problem. The programmer needs to have some expertise regarding the problem at hand in order to propose a solution, then convert the proposed solution to an algorithm and then to a program in some programming language.

The following scenario proposes an attractive alternative. A programmer has a completely new problem at hand. It is outside his current area of expertise and he has problems with proposing a solution. An expert who needs a solution to the problem decides not to use the programmer as an intermediary and to *explain* the problem directly to the computer instead. The expert presents *examples* of inputs and desired outputs to the computer, but not step by step *instructions* how to solve the problem. The computer produces a program which solves the given example problems and all similar problems. It is not difficult to imagine that a solution produced by the computer could be better than the solution the human programmer would propose. It is also possible to imagine that the computer is able to solve problems that a human programmer would not be able to solve at all.

One of the biggest challenges in Machine Learning (ML) [89] is a computer system able to program itself, so-called *Automatic Programming*. Perhaps the most popular technique for automatic programming nowadays is Genetic Programming (GP) [71, 12, 83], a branch

of Evolutionary Computation (EC) [122, 149, 150] and Artificial Intelligence (AI) [120] that, using principles from natural evolution, automatically generates computer programs. Since its invention almost two decades ago, GP has been used to solve problems in a wide variety of fields, in some cases so well that the results had been published as patents [76] (i.e. the results are better than any corresponding human creation).

GP allows the user to specify the inputs and the expected outputs for the problem at hand and then searches for a program that matches those inputs and outputs in an (approximately) optimal way. GP's potential is huge but it has been relatively explored little, particularly in its contribution to the field of Computer Vision. This thesis aims to explore the GP paradigm for Image Processing. The main contribution is made in the application of Mathematical Morphology (MM) [123, 124, 34, 35], one of the most popular techniques for non-linear image processing.

Section §1.1 presents a brief introduction to Computer Vision [10] and outlines some characteristics of Human Vision. In Section §1.2 a brief review of the different approaches to Image Processing [42] is presented. Section §1.3 explains the motivation for exploring the GP paradigm for Morphological Image Processing. It briefly introduces MM and explains the potential of exploring it by means of GP. Section §1.4 presents a list of the questions this thesis attempts to address and Section §1.5 presents an overview of the contents of this thesis.

1.1 Human Vision and Computer Vision

One of the most important aims of Artificial Intelligence is to create machines able to emulate the human perception and understanding of images. This field is known as Computer Vision.

To provide a computer with the ability to *see* is far from easy. Human vision is perhaps the most developed sense, since more than 80% of the brain activity is related to visual perception. Human vision is very complex and many characteristics are difficult to emulate.

Although there have been many successful approaches to vision problems concerned with shape, texture, colour, appearance, size, illumination, depth and movement, the computer understanding of images is still in its early stages. Some problems cannot be approached with the current technology such as high-level concepts, ambiguous pictures, non-canonical views, contextual images and camouflage. However, Computer Vision is a field that is growing very fast, in part because of the amazing growth of digital imaging technology and in part because of the developing of storage and processing equipment.

Practically any image may be digitalised nowadays and the scope of Computer Vision is growing every day. Some of the fields where Computer Vision has showed success are robot guidance, biomedical applications, industrial inspection, optical character recognition and satellite imaging.

Image Processing is often taken to be a subset of Computer Vision concerned with information processing where both the input and the output are images. However, this is a somewhat artificial boundary.

1.2 Approaches to Image Processing

The different stages of Image Processing include image acquisition, image enhancement, image restoration, colour image processing, compression, morphological processing, segmentation, recognition, representation and description.

The fields where Image Processing is applied are so varied that categorising images according to their source is a difficult task. It is interesting to note that many applications require several steps to achieve a particular purpose. A very important step refers to the recognition or enhancement of features with a particular size or shape. It is common that in the process an image is first *cleaned* to remove noise and after that a particular feature is enhanced.

A popular enhancement technique is to binarise images. This means that a threshold is applied to the grey levels of the pixels in an input image and the output image's pixels

(a) (b)

Figure 1.1: Example of binary feature enhancement (face features). (a) Image in gray-level. (b) Binary Image. From [1].

(a) (b)

Figure 1.2: Example of binary feature enhancement (people tracking). (a) Image in colour. (b) Binary Image. From [131].

have only two values which represent black and white (usually zero and one). Figure 1.1 shows an example of the face feature detection [1] and Figure 1.2 shows an example of

people tracking [131].

GP has been little used for Computer Vision and practically unexplored for non-linear image processing. Mathematical Morphology (MM), is a set of non-linear techniques for dealing with shape and is particularly suitable for the treatment of binary images. MM is a fairly well known area of image processing and it has shown some success in solving problems difficult for linear image processing.

1.3 Why Genetic Programming and Mathematical Morphology?

The use of GP for Image Processing has a number of potential attractions. For example, the solutions that GP may propose are likely to be different to those proposed by a human programmer and thus innovative. GP also does not have pre-conceptions that a human programmer may have and thus is likely to be less biased towards certain kinds of solutions.

It is often difficult for an expert to convert visual expertise into a computer program, because even though he may understand the process required he does not know how he actually *sees* or how to separate the problem into stages to obtain the desired output. For this reason GP offers a good alternative where, by means of examples, an algorithm may be extracted. Different to other ML techniques, GP offers solutions that after simplification provide meaning to the user, this means that an actual algorithm may be obtained from it.

MM offers the advantage of a unified field based on set theory that allows the user to understand what *actually* is happening when modifying an image. By evolving MM programs using GP it might be possible to explore a space of solutions which are difficult to imagine for human programmers. In this way it might be possible to extend the scope of applications of MM and improve its achievements.

1.4 About the Thesis

This thesis explores the potential of GP as a tool for the automatic generation of image processing algorithms based on MM. To date, very few researchers have attempted to evolve MM algorithms using GP and other Evolutionary Computation techniques. GP has shown some success in image analysis but it has been mainly used for linear image processing and search for filters, leaving the important morphological side practically unexplored.

We consider that this research may start an interesting trend since mathematical morphology is a versatile non-linear technique that allows the combination of exploratory features in several ways.

There have been other attempts to achieve the automatic programming of MM algorithms. Some of the techniques explored include optimal operators based on Statistics and Probabilities, AI, Greedy Algorithms, PAC Learning, Iterative Design, Switching Algorithms, Artificial Neural Networks, Fuzzy Sets and Hybrid Human-Machine Design. This literature is reviewed in Section §2.3.2. However, we believe that the most recent theoretical discoveries in the foundations of GP may provide a better context for the automatic generation of MM algorithms.

It is interesting to note that one important limitation for human practitioners using MM is the understanding of the results obtained when using non-regular structuring elements (SEs also known as kernels, see Section §3.1 and Figure 3.1). This is not a problem when using an automatic technique such as GP (i.e. humans do not need to understand it since it is automated).

1.5 Structure of the Thesis

This thesis is divided into six chapters. This chapter has provided an introduction to the problem of automatic programming of image processing algorithms including a definition

of morphological image processing. Chapter §2 includes a literature survey that reviews the general GP approach to Image Processing and the previous approaches for morphological automatic programming. It also includes an introduction to the basic morphological operators. Chapter §3 presents exploratory study of GP applied to MM. It includes a set of hypotheses, a discussion on fitness functions, implementation of the proposed approach, results and analysis. Chapter §4 attempts to overcome one of the problems emerged in this exploration, namely the feasibility of implementation. This is done using Sub-Machine-Code GP (SMCGP), a technique to speed up GP by using the internal parallelism of CPUs. Chapter §5 presents an alternative solution to improve the results obtained in the exploratory approach using image storage and a combination of morphological and logical operators. Finally, Chapter §6 presents the conclusions of this research, including a general analysis of results, significance, limitations and ideas for future work.

1.5.1 Publications

The research presented in this thesis has been partly published in peer-reviewed forums. An exploratory approach was presented in the Knowledge-Based Computer Systems Conference (KBCS) 2002 in Mumbai, India [110]. A second approach including SMCGP obtained the *best paper* award at the workshop of Evolutionary Image Analysis and Signal Processing (EvoIASP) in 2003 in Colchester, U.K. [111]. An extension of the research was accepted in the Journal *Genetic Programming and Evolvable Hardware* [109]. The discussion of Chapter §6 has been communicated to the International Symposium on Mathematical Morphology to be held in 2005 in Paris, France [108].

Chapter 2

Fundamentals of Genetic Programming and Mathematical Morphology

Chapter Outline

GP has a considerable potential to automatically create effective programs for Image Processing. This chapter introduces the fundamentals of GP and makes a brief survey of the main applications of GP to Image Processing. The survey shows that one of the many open lines of research is the GP approach to Morphological Image Processing, a branch of Computer Vision that has not been thoroughly explored using Machine Learning (ML) techniques. The basics of Mathematical Morphology (MM) are introduced with the general idea of the automatic generation of morphological image processing algorithms. We conclude with the motivation to explore the GP paradigm for morphological image processing.

The task of providing a machine with the ability to *see* is not easy; it is still an open scientific problem with some philosophical implications. The ultimate goal of Computer Vision [135, 10] is to emulate human vision, including learning, making inferences and taking actions based on visual inputs. The current technology is still far from providing a computer with the ability to understand images as well as a human being can. However, specific image processing and analysis applications have been used with success in both science and industry, and the number of applications is growing very fast. In addition, the availability of high performance hardware makes it possible to employ image processing in real-world applications.

In order to program a particular domain application it is usually necessary to have a group of programmers analysing the images (sometimes with the help of the domain experts) and proposing techniques to divide the problem into intermediate steps. Some of the intermediate steps may be solved by applying well known algorithms but most of the time the programmers need to use their intuition and expertise in order to solve the problem at hand.

One of the most exciting research threads in Image Processing is the automatic solution of these intermediate steps using Machine Learning techniques. In order to be implemented correctly by human programmers, a particular domain application should clearly state what is the final objective and what are the restrictions involved. However, it might be very difficult to express such an objective clearly and unambiguously. For example, a particular input image (e.g. a radiograph) simply may not make sense at all for non-specialists. Therefore, it is difficult to create programs to solve this kind of problem. This thesis supports the idea that this sort of problem could be solved by using EC [122, 149, 150] techniques and, more specifically, GP.

Genetic Programming [71] is a technique for getting computers to solve problems automatically without having to specify explicitly how to do it. GP is a very powerful optimisation scheme and it has been successfully used in a wide range of difficult applica-

tions, including automatic design [75], robotic control [13], data mining [148], synthesis of artificial neural networks [47, 46], bioinformatics [49], music [136], computer graphics [132] and picture generation [45]. The fundamentals of GP are introduced in §2.1 and there are numerous publications describing both theoretical studies (e.g., [83]) and applications (e.g., [71, 12, 78, 73, 74, 76, 77]).

GP is a descendant of Genetic Algorithms (GAs) [58, 41]. Whereas GAs have been widely applied to Image Processing [6, 96] and Pattern Recognition [97, 95], only a relatively small number of GP applications to Image Processing have been reported in the literature. This chapter presents a brief description of GP in §2.1, a survey of the GP approach to Image Processing in §2.2, the basics of Mathematical Morphology in §2.3, and the motivation to explore GP in the domain of Morphological Image Processing in §2.4.

2.1 Genetic Programming

One of the most succinct descriptions of GP can be found in Poli's paper [103]:

> GP is the extension of GAs [58, 41] 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.
>
> Programs are expressed in GP as syntax 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 2.1.
>
> The set of possible internal (non-leaf) nodes used in GP syntax trees is called the *function set*, $\mathcal{F} = \{f_1, \cdots, f_{N_F}\}$. The set of terminal (leaf) nodes in the syntax trees representing programs in GP is called the *terminal set* $\mathcal{T} = \{t_1, \cdots, t_{N_T}\}$. Table 2.1 shows some example functions and terminals used in GP.

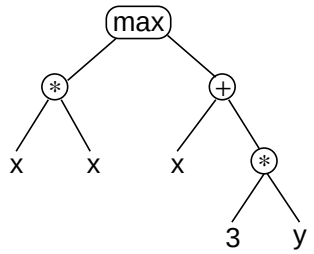


Figure 2.1: Syntax-tree representation of the expression `max(x*x,x+3*y)`.

The *search space* of GP is the set of all the possible (recursive) compositions of the functions in $\mathcal{F} \cup \mathcal{T}$. The basic search algorithm used in GP is a classical GA with mutation and crossover specifically designed to handle syntax trees. GP starts with an initial population of randomly generated computer programs composed of functions and terminals appropriate to the problem domain. The algorithm is controlled by a fitness function which evaluates the quality of the programs stochastically produced by GP. This typically requires running such programs (within the GP environment) on a number of test cases. The best individuals are selected and modified to produce a new population which is evaluated in the next generation. This process is repeated until a stopping criterion is reached.

Like other ML techniques, GP solves problems without being explicitly programmed. Because it directly searches the space of computer programs, its outputs (computer programs) can be immediately interpreted and used. Thanks to these two properties GP is an excellent tool to provide optimal or near optimal algorithms to solve problems which are difficult for humans to understand and solve, like some problems of Image Processing.

2.2 GP approach to Image Processing

Although there is no general agreement among authors regarding where Image Processing stops and Computer Vision starts [42, 135], it is generally agreed that Image Processing encompasses operations which transform one image into another. This includes the basic pixel representation of digital data and various processes applied to image data, such

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 2.1: Example functions and terminals used in GP.

as transformations, filtering, segmentation, enhancement, noise reduction, mathematical morphology, image compression and encoding, image restoration and others.

Although the basic Image Processing operations are based on sound principles of mathematics, signal processing, physics, etc., more complex algorithms for solving specific Image Processing problems are built up from the basic operations using heuristics and experience of a designer. An Image Processing expert who designs an algorithm to solve a new problem is very likely to use methods and algorithms shown to be successful in the past for a similar range of problems. Over the years this has lead to the development of a particular *library* of solutions, which can be considered to be a subset of a larger class of solutions, some yet undiscovered.

This thesis explores the idea that GP may offer an interesting alternative to expert-designed image processing algorithms. As GP does not have to be constrained by past experience or by perceptual or aesthetic qualities, solutions evolved in this way are likely to explore a broader range of the solution landscape than those designed by experts.

GP has been relatively little explored for Image Processing. The following Subsections review briefly the application of GP to Image Analysis(§2.2.1), Feature Detection(§2.2.1), Classification(§2.2.1) and other image related applications(§2.2.2).

2.2.1 GP for Image Analysis

Many Image Processing tasks involve the use of low-level algorithms to perform enhancement, segmentation or noise reduction. These processes can be implemented using filters which are optimised for the particular task.

Poli [101, 100, 102] describes a technique to find efficient filters using GP. He introduced a set of guidelines to make the search feasible and effective. The guidelines suggest a set of variables capable of capturing the information present in the image at different scales, the requirement of non-complex calculations (ideally only memory accesses) and a computation load as light as possible. The implementation showed very good results on several examples in the medical imaging domain.

Poli and Cagnoni [106] use GP for image enhancement in a user driven approach, where the presence of the user transforms GP into a very efficient search procedure without fitness function. To overcome the user bottleneck caused by the limited speed at which complex comparisons can proceed, they proposed the idea of automatically inducing programs to model the user's behaviour.

Depending on the image nature and the particular application it could be necessary to make some intermediate low-level steps before it is possible to obtain useful information. Daida *et al.* [31, 24] developed a GP system for Image Analysis and Scientific Inquiry in Geoscience and Remote Sensing. The system is a powerful tool for the processing of images in problems considered untenable for standard image processing algorithms due to low signal-to-noise ratios, arbitrariness of feature shapes and orientation problems.

Feature Detection

Feature detection is an image analysis operation which attaches a semantic label to a group of pixels. Features may vary from simple geometric entities such as lines, blobs or geometric forms to complex entities such as $2D$ and $3D$ objects.

There are many examples of the successful use of GP for the operations in this class, including the detection of a variety of objects [72, 147, 154, 126, 127] and targets [23, 20, 62], visual features [19, 100], features in SAR [22, 21] and infrared imagery [116], generic [7] and specific $2D$ features including the presence of edges [51, 37, 38, 50, 119], and instances of a specific orientation [117]. Further examples include the extraction of facial features [64], face recognition [143] and multi-class object detection [156, 155].

Classification

Classification denotes a process in which a decision rule is applied to categorise a set of image data. Usually this step transforms image data into symbolic information and follows earlier steps, namely enhancement, segmentation and feature detection. A good classification system should be able to correctly separate sets of images with features previously defined.

There are numerous examples of GP-based classification. Relatively simple examples include classification applied to printed text [28, 87] and car number plates [2]. Most typical is the classification of specific objects or features (for example roads [125]) in complex images such as satellite images [32], remotely sensed images [112], Synthetic Aperture Radar data [30, 137], sonar [88] imagery and multi-spectral [113], IR image sets [140, 141] and visual routines [66].

GP has been also explored to improve the classification performance using pre-classification [138, 139], supervised classification [128, 129], supervised learning classifiers [5], multi-class classification [44, 157, 158], multi-category pattern classification [67, 68] and general purpose classification [144, 145].

2.2.2 Other Image Related Applications

As the use of GP becomes generally popular, so are its applications in the image-related fields. Examples of these varied applications range from those operating purely on the image data, such as texture generation [146], restoration [59], equalisation [39], colour [118], pseudo-colour [105], compression [94, 40, 61, 65, 90] and video compression [26], to complex computer vision operations, such as automatic generation of programs for low-level vision [159, 4], model-based vision [92], stereo vision [43, 82], shape modelling [98, 99], tracking [98], and motion analysis [45, 114, 133, 134].

2.3 Mathematical Morphology

Mathematical Morphology (MM) [123, 124, 34, 35] is a relatively separate part of image processing. The non-morphological approach to image processing is close to calculus, being largely based on the point-spread-function concept and linear transformations such as convolution. Instead, 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 the different regions of an image. 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 2.2 the set of all black pixels (which corresponds to the 1s 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 $2D$ integer space Z^2. Each element of a set is a tuple (x_1, x_2) representing the coordinates of a black pixel in the image. For example the set of pairs represents the black square in Figure 2.2

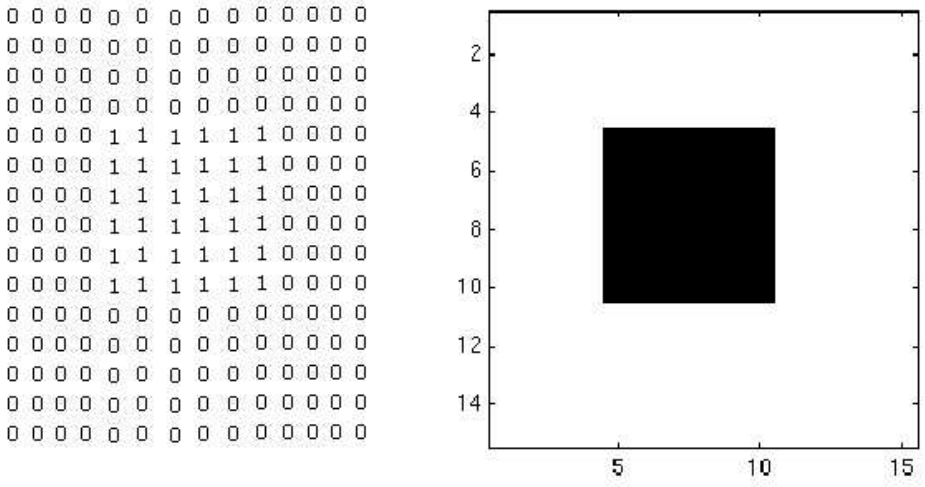


Figure 2.2: Binary image as the set of all black pixels.

as follows

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

Grey-scale digital images can be represented as sets whose components are in Z^3. Each element of a set is a triplet (x_1, x_2, x_3) representing the coordinates of a pixel and its grey level. Figure 2.3 shows a simple example of a grey-scale image with three grey levels. The image corresponds to the set

$$\{(1,1,2), (1,2,2), (1,3,2), (1,4,2), (1,5,2)$$
$$(2,1,2), (2,2,2), (2,3,2), (2,4,2), (2,5,2)$$

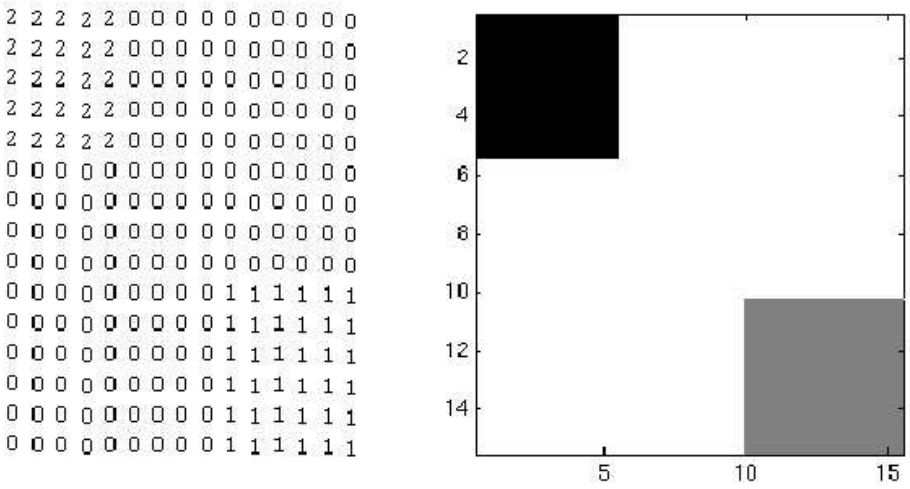


Figure 2.3: Grey-scale image with three grey levels.

(3,1,2), (3,2,2), (3,3,2), (3,4,2), (3,5,2)

(4,1,2), (4,2,2), (4,3,2), (4,4,2), (4,5,2)

(5,1,2), (5,2,2), (5,3,2), (5,4,2), (5,5,2)

(10,10,1), (10,11,1), (10,12,1), (10,13,1), (10,14,1), (10,15,1)

(11,10,1), (11,11,1), (11,12,1), (11,13,1), (11,14,1), (11,15,1)

(12,10,1), (12,11,1), (12,12,1), (12,13,1), (12,14,1), (12,15,1)

(13,10,1), (13,11,1), (13,12,1), (13,13,1), (13,14,1), (13,15,1)

(14,10,1), (14,11,1), (14,12,1), (14,13,1), (14,14,1), (14,15,1)

(15,10,1), (15,11,1), (15,12,1), (15,13,1), (15,14,1), (15,15,1)},

where $x_3 = 1$ stands for light grey and $x_3 = 2$ for black. Sets in higher-dimensional spaces can represent other image attributes, such as colour and time varying components.

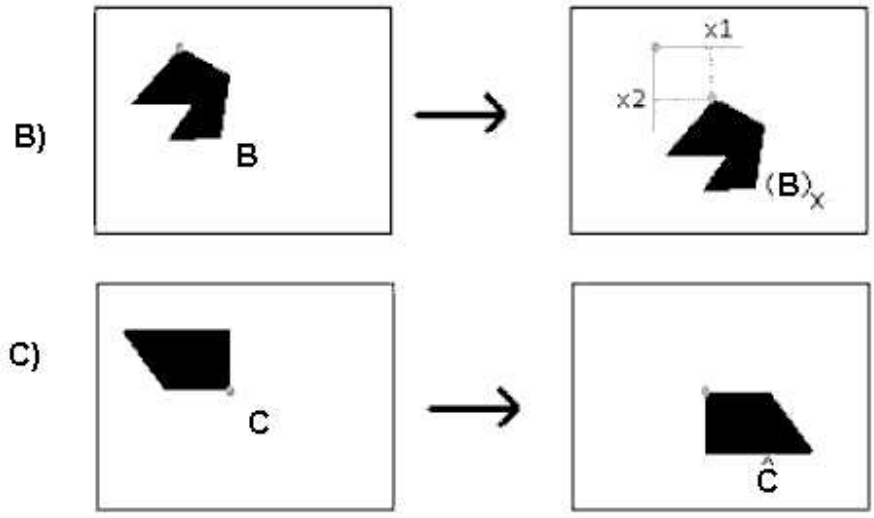


Figure 2.4: Translation (B) and Reflection (C).

2.3.1 Basic Morphological Operations

The two most used morphological operations are *dilation* and *erosion* and most of the morphological algorithms developed by human experts to perform a particular task make heavy use of these operators.

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

$$(B)_x = \{w | w = b + x, \{b \ \in B\}\} \tag{2.1}$$

The reflection of set C, denoted $\widehat{C}$, can be defined as

$$\widehat{C} = \{w | w = -c, \{c \in C\}\} \tag{2.2}$$

Figure 2.4 shows examples of translation and reflection.

If A and K are sets in 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

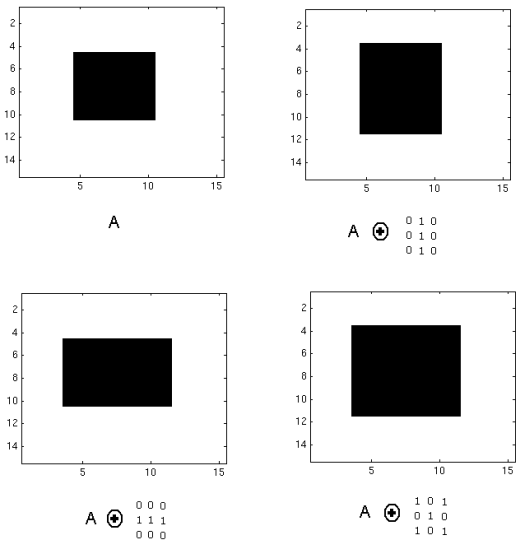


Figure 2.5: Examples of dilation of the image in Figure 2.2 using different SEs.

$z \in Z^2$ 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:

$$A \oplus K = \{z|(\widehat{K})_z \cap A \neq \emptyset\} \tag{2.3}$$

Usually the set K is called *structuring element* (SE or *kernel*). The origin of K may be arbitrarily chosen pixel (usually belonging to K but that is not a restriction). The SE is often small and its shape directly modifies the shape of set A. Figure 2.5 illustrates how different SEs can produce a thickening effect on the image in Figure 2.2.

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. So, the erosion of A by K is:

$$A \ominus K = \{z|(\widehat{K})_z \subseteq A\} \tag{2.4}$$

This operation is illustrated in Figure 2.6 where different SEs produce a thinning effect on the image in Figure 2.2.

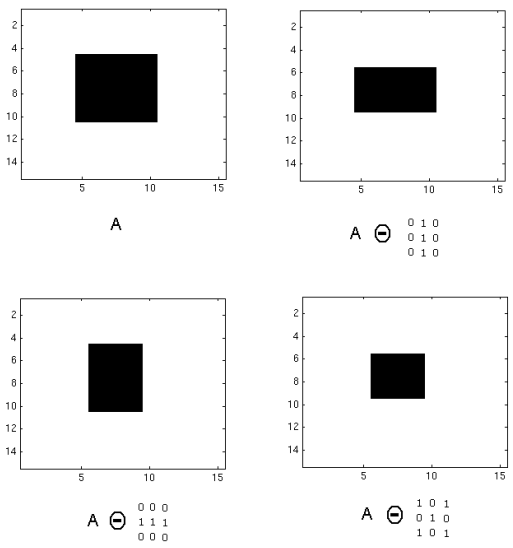


Figure 2.6: Examples of erosion of the image in Figure 2.2 using different SEs.

Equations 2.3 and 2.4 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 for linear filtering.

Gonzalez and Woods [42] present some 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).

2.3.2 Automatic Approach to MM

MM is a very flexible tool that has been used for a wide range of applications including biomedical imaging, document processing, pattern recognition, microscopy, inspection and robot vision [35]. The versatility of MM suggests that the technique has a huge potential which may not have been completely explored yet.

Two main factors need to be considered in the design of an MM algorithm: the SEs

(also known as morphological filters or *kernels* particularly when referring to binary images) and the sequence of morphological operators. Morphological filters are an important class of non-linear digital signal processing filters, which have found a range of applications. However, few generic design methods exist for morphological filters. The selection of the best type of filters to use and the order in which to apply them are normally application-specific.

In the recent book *Hands-on Mathematical Morphology* [35], Dougherty and Latufo dedicate the final chapter to the automatic design of morphological operators with a detailed review to the general approach to automatic morphological image processing:

> The key to successful morphological image processing is the selection of SEs. There are a myriad of algorithms for a multitude of imaging applications, but in each and every instance, algorithm performance depends on the SEs. The classical approach to morphological processing is to have a human being, or group of human beings, use intuition and an understanding of the goals to design algorithms based on erosions, openings, hit-or-miss transforms and other basic morphological operators. This approach can work well if the task can be described in elementary geometric terms and the images under consideration are not too complex. It breaks down in situations where satisfactory filtering might require hundreds, or even thousands, of SEs.

The characteristics of MM imply that the algorithms designed by humans are limited by their understanding of the problem at hand and their experience in using morphological operators combined with SEs.

Figure 2.7 shows an example of a particular task we may want to perform on a binary image. The original image (o) contains several features including a moon, a tree, some stars and a mountain. Let us imagine that our goal is to extract the *mountain* (the goal image (g)). The simplest method would be to extract the mountain by hand using image

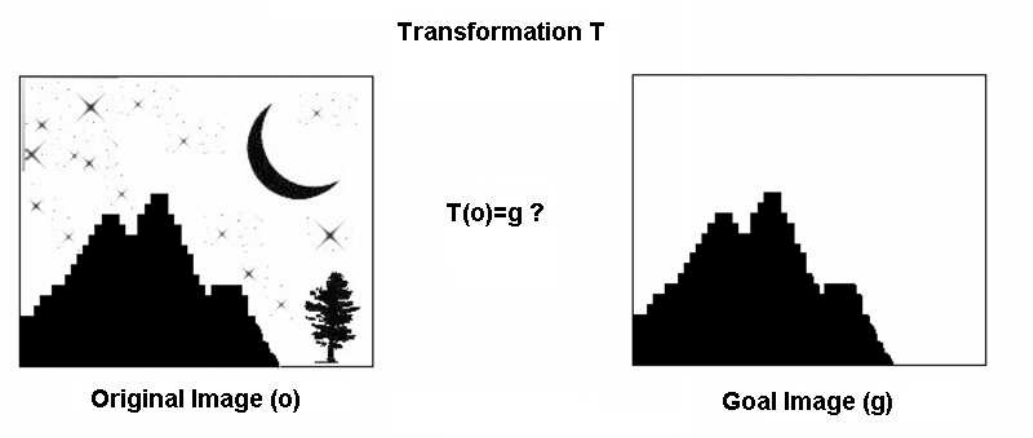


Figure 2.7: Transformation from Original Image (o) to Goal Image (g).

editing software. This would be too tedious if there were many similar images to process. The challenge is therefore to create a transformation algorithm T that applied to the image (o) produces the image (g). In the ideal case, the *extract-mountain* algorithm should be able to extract mountains from all images with similar characteristics as (o). In the general case this is not a simple task and requires a reasonable level of image processing expertise.

It would be clearly beneficial to have an automatic system that finds the *extract-mountain* algorithm T by itself, with the user providing some input-output image pairs as examples and obtaining an operator (algorithm T) as an output. To build that kind of system several approaches have been suggested in the literature, namely optimal operators based on Statistics [15], Probabilities [36], Artificial Intelligence [121], Greedy Algorithms [48], PAC Learning [16], Iterative Design [56], Switching Algorithms [57], Artificial Neural Networks [130, 153], Fuzzy Sets [91, 86] and Hybrid Human-Machine Design [14].

Most recently Evolutionary Algorithms have also been applied to this problem and Section §2.3.3 provides a detailed survey.

2.3.3 The Use of GAs in Mathematical Morphology

GAs are the most popular technique of EC and have been widely used for optimisation problems in morphological image processing. Some of the areas of application are the decomposing of SEs, the design of morphological filters for grey level images and the design of MM algorithms for binary images. Additionally MM have provided grounds for the development of GAs theory. We will describe these ideas below.

Decomposing of SEs

A SE can usually be decomposed into smaller SEs that offer an alternative way of obtaining the same result. For implementation reasons it is desirable to have a sequence of small SEs rather than bigger arbitrarily shaped SEs (the use of large SEs is not efficient). The problem of decomposing SEs for binary images has been addressed using a parallel implementation of Genetic Algorithms, and relatively good results have been obtained [8].

Real-coded GAs

There is an interesting interconnection between different areas in Computer Science. Ideas from MM have provided grounds for the development of a new type of real-coded GA. A crossover operator based on morphological operators proposed by Barrios *et al.* showed interesting results in empirical experimentation, the use of morphological crossover operators showed an improvement in the applications explored altough the examples might not be statistically significant [17, 18].

Design of Morphological Filters for Grey-Level Images

The design of morphological filters for grey-level images is an optimisation problem in itself, which involves a large number of variables according to the filter size and the image resolution. Harvey and Marshall [54, 52, 53, 55] demonstrated how simple GAs can be employed

in the search of one (grey-level) morphological filter obtaining optimum performance for a given image processing task. Furthermore, the research was extended to an evolutionary approach for morphological algorithm design looking for suitable sequences of four morphological operations. Even though the search was limited to fixed-length sequences of morphological operators, a `do-nothing` operator was introduced so as to effectively allow the evolution of variable size sequences of morphological operators. An interesting result of this research was that the `do-nothing` operator appeared very often in the chromosomes, indicating the importance of allowing the exploration of variable-length solutions.

Kraft *et al.* [80, 79] extended the research using parallel GAs. It is obvious that when the image size and the filter size are large, the feasibility of an optimum filter design becomes unpredictable. However, the results show that parallel GAs are a useful tool for the design of morphological filters.

In an attempt to automate morphological filter design for target detection in grey-scale images, Nong *et al.* [93] implemented chromosomes with a fixed number of components and the Mean Square Error (MSE) as their fitness function, proposing also new crossover and mutation operators to improve convergence. They showed some experimental results which demonstrate the effectiveness of their method.

The adaptation of structural elements using GAs for texture analysis in grey-scale images was the purpose of research carried out by Köppen and Teunis [69]. The main contribution was the introduction of a fitness function based on histogram distribution which could be generally expanded to every case of adaptation of morphological sequences to certain image filtering tasks. The methodology proposed shows the ability of GAs to find masks which are able to represent a pattern in a textured image. However, the methodology failed to adapt masks on non-regular textures like those on the surface of castings.

Design of MM Algorithms for Binary Images

Yu *et al.* [152] proposed a simple approach for 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 of the individual with respect to 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 SE.

Yoda *et al.* [151] opened a very interesting research area 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 variety of structuring elements to four different regular structures, they demonstrated that the proposed method can obtain good solutions. Although the required computational effort increases sharply when the task to perform is more complex, they suggested that it is possible to find an MM algorithm that extracts a particular feature from a binary image in a general purpose framework.

Bala and Wechsler [9] presented a methodology for combining mathematical morphology and GAs to generate high-performance shape discrimination operators on binary images. The encoding scheme consists of 4 operators with 32 different structuring elements allowing the repetition of an operator. The results show a feasible approach for shape discrimination, including an operator that discriminates among the shapes according to previously specified classes. They have also defined a new fitness function which is a performance index based on the average of responses yielded by the operator.

Loncaric and Dhawan [84, 85] devised a shape representation called Morphological Signature Transform (MST) that uses the areas of images iteratively eroded by multiple SEs as shape descriptors. GAs are used for the selection of optimal structuring elements in the MST method for shape representation and shape matching. The experiments have shown that the optimal structuring element enables accurate shape matching and is robust to noise.

Brigger and Kunt [27] show how GAs can be used for the automatic optimisation of arbitrary shaped structuring functions in skeleton decomposition. The application to geometrical shape representation in binary images shows that these features are useful for image coding. According to the results, GAs provide a robust and efficient search in a complex search space with the introduction of a shape-oriented crossover operator.

Huttunen *et al.* [63] provides an approach for the optimisation of morphological filters for noisy conditions (also referred as to *soft-morphological filters*) using GAs. The object of the optimisation is to find optimal or near-optimal filters. The results of the experiments show that the filters may be optimised if the parameters of population size (around 50) and mutation rate (around 0.03) are properly selected. The optimisation criterion with various noise percentages used is MSE. They also studied the effect of various noise percentages.

2.4 Motivation to Explore GP for Morphological Image Processing

Faced with an image processing problem, an MM practitioner would draw on his own knowledge and experience to develop solutions to the problem. This would normally involve the use of published algorithms developed by the experts in the field. If the initial solution is not good enough, the practitioner has to continue improving it and in the process of doing so is likely to improve his own expertise.

As in all areas of Computer Science, Mathematical Morphology has developed its own

set of high-level algorithms such as, for example, skeletonisation, morphological smoothing, distance transform, etc. However, there are relatively few of these algorithms and the use of basic operators – erosion and dilation – in the design of specific algorithms is very common. This particular aspect of MM poses a challenge to the developers. The number of possible structuring elements from which to choose is very large and this, combined with the possible choices of morphological operator sequences, makes morphological image processing something of a *black art*.

All the efforts towards an automatic approach for morphological image processing are limited by the lack of well defined criteria for the selection of SEs and morphological operators. Another limitation is the lack of measures for deciding whether an MM algorithm has been successful in solving the problem at hand.

Research described in this thesis is motivated by the success of GP in other difficult design fields. It is built on the hypothesis that by using GP it is possible to construct programs (i.e. algorithm implementations) for morphological image processing which perform as well as programs constructed by a human expert.

At the onset of this research it was apparent that it is possible to define a set of MM functions and terminals on which GP could operate. It was also clear that using GP it is possible to evolve algorithms which solve general domain problems. Although the use of GAs in the context of MM has provided useful insights into the potential of Evolutionary Algorithms for morphological image processing, there are a number of questions that have not been answered, or at least could be re-framed in the context of Genetic Programming:

* Most of the applications using GAs have a limited number of morphological operators. Would the increase in the number of morphological operators result in better algorithms?

* GA approaches usually have a fixed number of morphological operators. Would the inclusion of a variable number of morphological operators result in better algorithms?

* GA approaches have not explored the application of irregular SEs. Would the combination of irregular and regular SEs result in better algorithms?

* There is no reason to restrict the search to a particular size of SEs. Would the combination of SEs of different sizes result in better algorithms?

* GA approaches have not combined morphological operators with other operators (like logical). Would the combination of logical and morphological operators result in better algorithms?

* Most of the automatic approaches have restricted the evaluation of the success of the image processing operations to the Mean Square Error and Mean Average Error applied pixel-wise as the measure of fitness. Are there any other, possibly better, measures to evaluate the success of an automatically generated algorithm?

* The ideal case would be to come up with an automatic design technique able to solve different problems using similar features. This is known as *generalisation*. How good is GP as a general tool for the automatic design of high-quality MM algorithms?

* GP is an expensive technique from the computer resources point of view (memory load and processing time). How can we make the search of morphological algorithms feasible and effective?

In this thesis we will attempt to answer all of these questions.

2.5 Chapter Summary

This Chapter has given an introduction to GP as a powerful technique for automatic design of computer programs. In particular, the use of GP in the field of Image Processing has been reviewed.

It has been pointed out that an important sub-field of Image Processing, namely Mathematical Morphology, has been little explored within the GP paradigm. An introduction

to MM has been provided, followed by a review of research on automatic programming and the particular contribution of Evolutionary Computation in this context.

It has been concluded that there has been little research on GP applied to MM, and that the underlying field of the automatic design of morphological algorithms are in early stages. These findings have motivated the work presented in this thesis. In Chapter §3 we state the hypotheses that this thesis aims to evaluate and we outline the proposed approach to exploring Genetic Programming as a technique for the automatic generation of MM algorithms.

Chapter 3

A GP Approach to Mathematical Morphology

Chapter Outline

This chapter presents a GP approach to morphological image processing. A set of hypotheses to answer the open questions of Section §2.4 is discussed in Section §3.1. An important part of any automatic approach to image processing is the way to evaluate how similar two images are. This issue is explored in Section §3.3. The approach is explained in Section §3.4 including the data used, a set of experiments to test the hypotheses from Section §3.1, analysis and discussion. Section §3.5 presents the testing of the algorithms evolved in the implementation using a set of images not included in the training set. The conclusions are presented in Section §3.6 including the limitations of the study of this chapter.

3.1 Hypotheses

The interesting results of GAs on the automatic generation of MM algorithms suggest that there is a huge search space to be explored. For instance, Yu *et al.* [152] provide a GA approach to image segmentation by means of morphological operations. The results were promising but the search was restricted to four morphological operators. Furthermore, the search was restricted to sequences of one *closing* (dilation after erosion) followed by one *opening* (erosion after dilation) using only one squared structuring element. Is it really necessary to restrict the search space that much?

Another interesting example is the GAs research by Yoda *et al.* [151]. The search was again restricted in three different approaches. First, it was restricted to sets of two morphological operators and two structuring elements combined with two logical operators. The second approach included the combination of a whole GA run (they name it *era*) with logical operators in order to recover over- and under-extracted data. The third approach increased the number of structuring elements to four and the length of the sequences was increased up to 30. This is still a highly restrictive approach, since it is using only a reduced number of structuring elements and training sets.

Perhaps the main restriction on those investigations was due to computational resources but it was also due to the nature of GAs which use fixed-length chromosomes (i.e. MM sequences). GP is represented using tree-like structures while GA is represented using strings. GP is likely to perform better than GAs since it is a technique capable of broadening the search space in comparison to GAs [12]. The additional capabilities that we propose include the use of sequences of MM operators of variable length, the use of structuring elements of various sizes, the introduction of *irregular* structuring elements and the combination with logical operators. It is clear that fixed length sequences in GAs do not allow many different ways to approach the evolution of morphological algorithms. GP is more flexible and allows the inclusion of sequences of variable size. This thesis intends to

explore and test the hypotheses discussed below:

* *Application related hypotheses*

    * *The inclusion of irregular SEs in addition to regular SEs will result in better performance than with the use of regular SEs alone.* It can be observed that in most MM algorithms developed to date regular SEs such as squares, lines and crosses are used more often than irregular SEs. This is understandable because it is more natural for human programmers to use simple regular structures. An automatic programming technique such as GP is not constrained in this way and hence can easily explore a larger search space including irregular structuring elements (see Figure 3.1). The availability of a larger search space may result in new, previously unknown, algorithms which may perform better.

    * *Small SEs will perform tasks faster and better than large SEs.* This is expected because several small kernels can perform the same task as one large kernel. The advantage of using small kernels is the increased diversity of the search space. Another advantage is that computation using a small structuring element is more efficient.

    * *Combination of Morphological and Logical Operators will achieve better results than morphological operators alone.* Combining these operators is a common programming practice in image processing and hence it is worth exploring in the context of GP.

* *Methodology related hypotheses*

    * *A larger training set will lead to a better performance than a smaller training set.* This is expected because a training set with a bigger number of images would provide more information to GP in order to evolve more accurate algorithms.

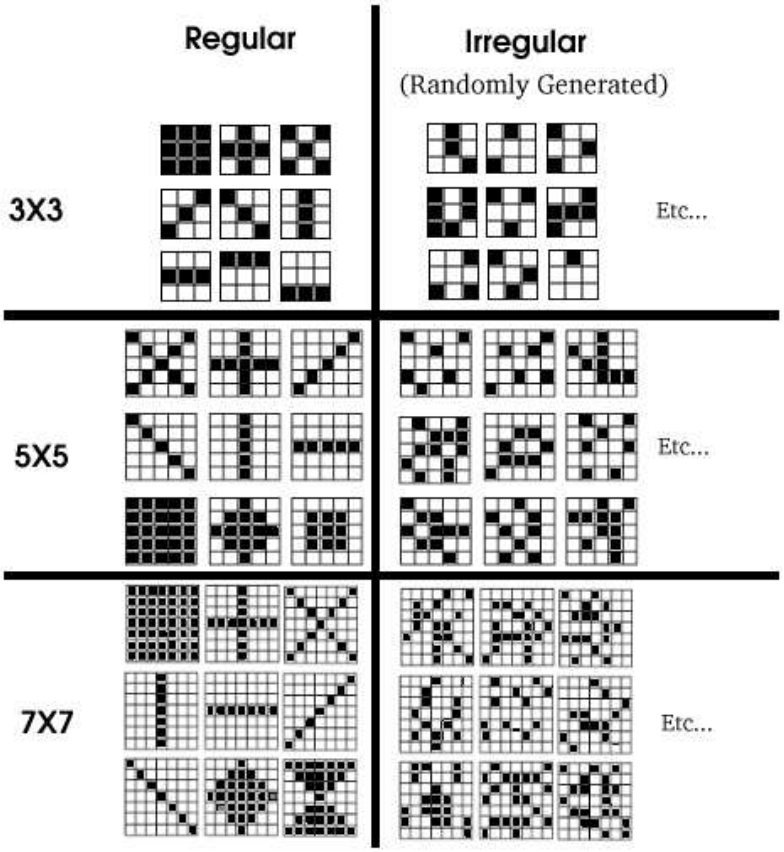


Figure 3.1: Regular and irregular SEs of sizes 3×3, 5×5 and 7×7

On the other hand, this may have the disadvantage of slowing down the GP search.

* *A training set with bigger images will perform better than a training set with smaller images.* This is expected because bigger images would include more data that would allow GP to discriminate among features. Again, this may have the disadvantage of slower search.

3.2 Problems of GP Applied to Images

When GP is applied to real-world binary images it faces several constraints that have to be addressed in order to make the search feasible and effective.

* *Implementation Feasibility.* It seems that GAs implemented for the automatic discovery for morphological algorithms were restricted due to limited computational resources. The populations used are relatively small (50-200 individuals on 100 generations [152] and 100 individuals on 6-50 generations [151]). Since the implementations to solve other problems using GP usually make use of much bigger populations over a large number of generations, we should investigate how to make the search feasible and effective in these conditions.

* *Training set size.* GP usually uses very big training sets including thousands of examples to drive the evolution. When using images this is not feasible since applying a big number of GP individuals (programs) to a big set of images (test set) may require a huge amount of processing time and memory allocation depending on the primitive set, the size of the images and the resolution.

* *Size of the images.* If the images where the GP generated algorithms are meant to be applied are large there is a problem in the creation of a suitable training set. The inclusion of high-resolution images in the training set requires larger computational resources.

* *Creating the training set.* In some real-world applications it is difficult to create a good training set. If, for example, face detection is attempted, it is usually necessary to create a training set using an image editing software on images such as the one shown in Figure 3.2. This requires subjective skills of the expert and does not guarantee that the training set is accurate. The task of creating a training set may be difficult and tedious.

* *Pixel detection and feature detection.* When applied to real-world binary images the problem of detection has to be extended to features instead of pixels. The problem is that the task of counting features by hand it is not practical most of the

Figure 3.2: Example of image for face detection where subjective skills are needed to create a training set.

times. Counting the number of features detected or mis-detected is a task not always easy to do, particularly when there are features overlapping or features with similar characteristics.

* *Ambiguous Images.* There may be cases of ambiguous images, images with overlapping or similar features. In those cases it is difficult to define a training set even for imaging experts. GP training may be biased due to the creation of a training set that must be based on the observer's assumptions.

3.3 Fitness Functions

In order to measure how fit an individual is in a GP population it is necessary to introduce a suitable Fitness Function. When GP is used for image processing it is difficult to define an equation that adequately measures the differences between input and output images. Particularly in binary images this is an issue that has been approached in different empirical

ways.

3.3.1 Sensitivity/Specificity Dilemma

When Poli [101, 100] started exploring GP for Image Analysis he confronted the dilemma of maximising both *sensitivity* and *specificity*. Basically the problem is that no detection or segmentation algorithm can perfectly detect the features of interest without mis-detections or over-detections.

Our objective is to convert the original image, o, into the goal image, g. 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):

$$SV = \frac{TP}{TP + FN}$$

$$SP = \frac{TN}{FP + TN}$$

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. (These true/false positive/negative numbers are obtained by integrating over every pair of (x_i, y_i) coordinates in the original image.)

We shall illustrate the concepts of sensitivity and specificity using as an example of the image shown in Figure 3.3. Its size is 640×480 and it contains four features, namely circles, squares, rings and stars. The task is to find an algorithm which extracts the squares only. An algorithm that converts the image in Figure 3.3 into the image in Figure 3.4 provides the highest sensitivity and the highest specificity, since it accurately detects the squares without false positives or false negatives.

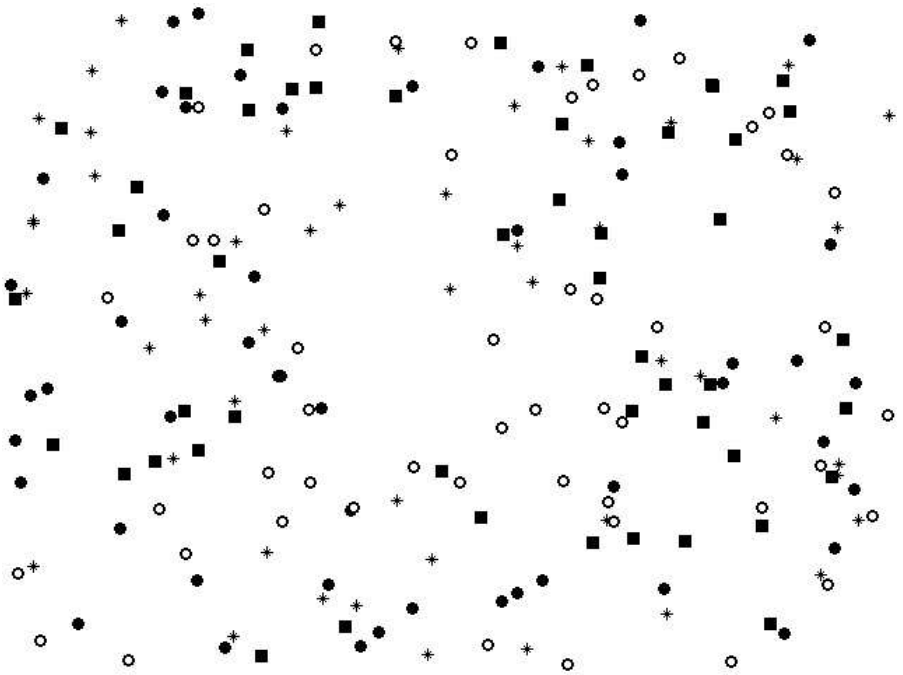


Figure 3.3: Example of a binary image with four features: squares, circles, stars and rings

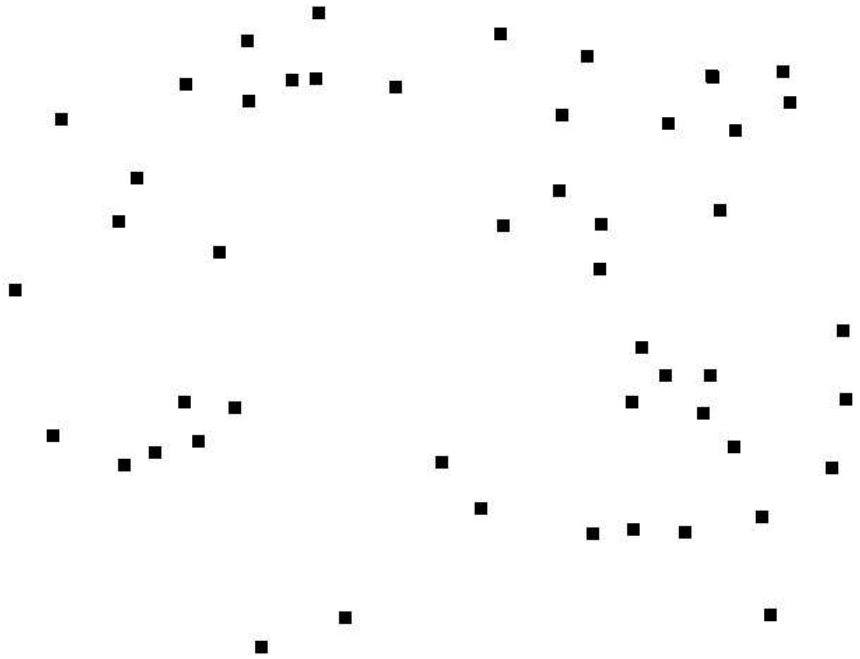


Figure 3.4: Example of image detection of squares from Figure 3.3 with high sensitivity and high specificity.

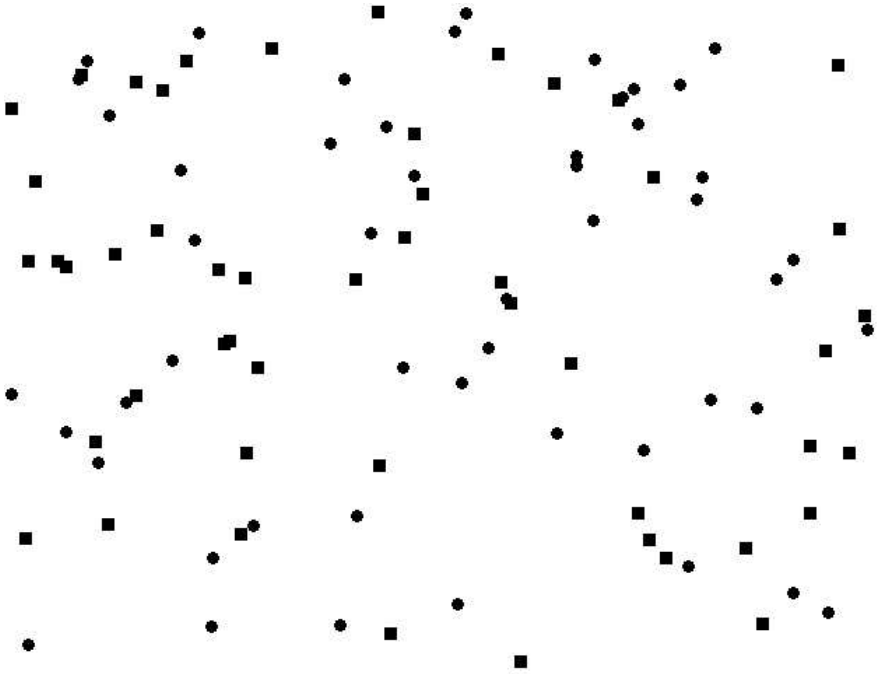


Figure 3.5: Example of image detection of squares from Figure 3.3 with high sensitivity and low specificity.

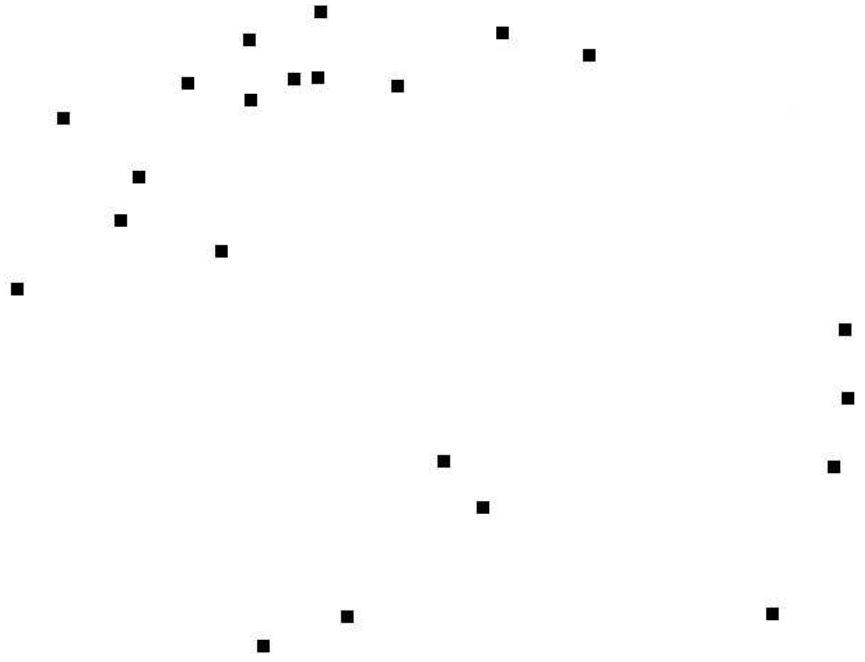


Figure 3.6: Example of image detection of squares from Figure 3.3 with low sensitivity and high specificity.

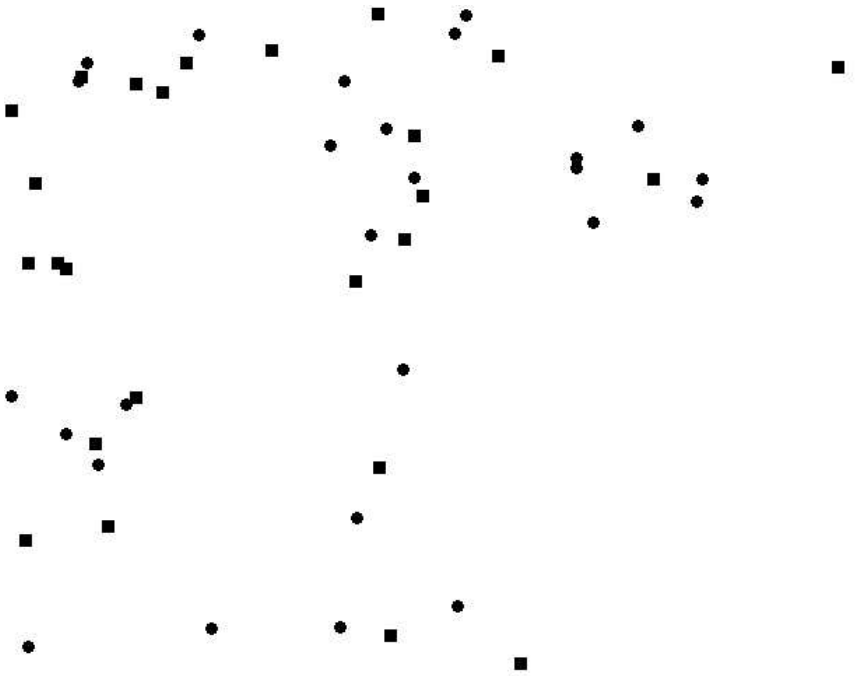


Figure 3.7: Example of image detection of squares from Figure 3.3 with low sensitivity and low specificity.

An image in Figure 3.5 represents an example of high sensitivity and low specificity (i.e. it accurately finds the squares but it also detects the circles which is not desirable). An image in Figure 3.6 represents an example of low sensitivity but high specificity (i.e. it does not find any other features but squares, however it does not detect all the squares present in Figure 3.3). An image in Figure 3.7 represents an example of low sensitivity and low specificity (i.e. it does not detect all the squares present in Figure 3.3 and it has several non-square features as well).

The first fitness functions proposed by Poli [101, 100] intended to minimise the number of false positives and false negatives (i.e. $f = FP+FN$ and $f = FP^2+FN^2$). The problem found was that minimising their squared sum does not necessarily lead to the maximisation of both sensitivity and specificity. Poli found a fitness function that provided better results:

$$f = FP + FN \; exp \; (10 \; (\frac{FN}{P} \; - \; \alpha)) \tag{3.1}$$

where P is the number of pixels belonging to the structures to be detected and α is a domain dependent parameter. The introduction of P and α requires *a priori* knowledge about the problem at hand. First, a clear understanding of the pixels that belong to the features to be detected is needed. Second, an estimation of what kind of algorithm is required (i.e. a high value of α bias the search toward highly sensitive algorithms while a low value of α biases the search toward highly specific algorithms).

In later research, Poli and Cagnoni [106, 105] proposed that a user might drive the selection for Image Enhancement. This approach is an efficient way to produce solutions of real-life problems without using a fitness function. Nonetheless it has the disadvantage of the subjective evaluation by a user and the requirement of hundreds of evaluations (meaning many user-hours spent on selecting images).

3.3.2 Similarity f_A

To tackle the problem of shape feature detection Yoda [151] 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

$$f_A = \frac{(f \cdot g)}{\sqrt{(f \cdot f)}\sqrt{(g \cdot g)}} \tag{3.2}$$

where f and g are two digital images and

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

The function f_A tries to maximise the number of matching *black* pixels in the images f and g. The optimum is $f_A = 1$ when all the pixels match. The worst case is $f_A = 0$ when none of the pixels match. This function tries to maximise sensitivity since it ignores all the cases where $f(i,j)$ and $g(i,j)$ are different.

3.3.3 Resemblance f_B

We defined a new fitness function f_B known as *resemblance* ($0 \leq f_B \leq 1$) related to the *trade-off* between sensitivity and specificity needed in detection algorithms [81].

$$f_B = \frac{SP^2}{2} + \frac{SV^2}{2} - (SP + SV) \tag{3.3}$$

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 $SP = SV = 0$. The difference between Eqs. 3.3 and 3.1 is that f_B is normalised and that f_B does not introduce domain knowledge (i.e. f_B intends to be a generic fitness function for binary image processing). Different to $f = FP^2 + FN^2$ [101, 100], f_B focuses in specificity and sensitivity instead of minimising false positives and false negatives.

3.4 Experiments

3.4.1 Data Used and Process Steps

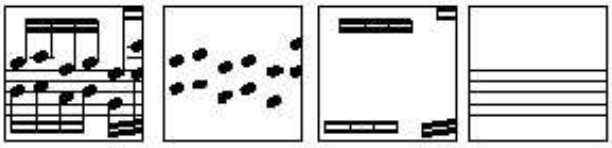
In order to test the hypotheses discussed in Section §3.1 we propose to use data with the following characteristics.

* Use a training set from a real-life problem.

* Use a training set where several features related to shape may be extracted to test the *generalisation* of the proposed method.

* Use a training set where the image size can be varied.

* Use a training set where the number of images used can be varied.



o g_1 g_2 g_3

Figure 3.8: Examples of binary transformations. The original fragment of a music sheet (o) and three possible goals extracted by hand: noteheads (g_1), beams (g_2) and staffs (g_3).

* Use a test set different from the training set using images with the same resolution to prove unbiased generalisation.

* Use a test set using bigger images than the training set images to visualise unbiased generalisation.

The GP approach is used to find a sequence of morphological operators in the MM algorithm's search space to convert an image into another one containing only a particular feature of interest. We chose musical sheets as examples with the objective to extract three different features, namely noteheads, beams and staffs. Examples of the desired binary transformations are shown in Figure 3.8.

We performed a total of 1512 GP runs combining 3 features to extract (noteheads, beams and staffs), 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). All the experiments were implemented using Lilgp, a popular GP toolkit written in C [160].

The terminal set includes primitives that have the following form:

$$x(yz[w])$$

where: $x \in \{e, d\}$ represents a morphological operator (erosion and dilation); $y \in \{R, I\}$ 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. The number of structuring elements was chosen to balance regular and irregular (i.e. we identify 11 comonly used regular structuring elements). For example, the primitive $e(R3[7])$ represents erosion using the seventh regular structuring element of size 3×3.

We use a function set including two functions, `EVAL1` and `EVAL2` of arity 1 and 2, respectively, which simply execute their arguments. `EVAL1` and `EVAL2` are used to transform the GP tree into a linear representation.

The process is visualised in Figure 3.9. The first step is to build a set of examples of correct feature extraction to be used as training sets in GP (Figure 3.9A). Next, it is necessary to define the type, size and number of structuring elements to be made available for GP (Figure 3.9B). After the *primitive set* is defined (Figure 3.9C) we start the GP search to obtain a (near) optimum tree representing an MM algorithm (see Figure 3.9D). The algorithm includes a sequence of MM operations; note that this is not a fixed-length sequence. The best evolved sequence, according to the fitness value, is also analysed visually (Figure 3.9E) to decide whether or not the numerical fitness value is reflected by the image quality. If it is, the problem is solved. Otherwise one needs to run the algorithm again. As shown in Figure 3.9D, at evaluation time each GP tree is transformed into a linear representation, which, when read from left to right, represents an MM algorithm. This is applied to the training set in order to evaluate the fitness of the corresponding tree.

The GP parameters we have used in our experiments are similar to those suggested in Koza [71]. Specifically, a 0.9 crossover rate and 0.1 mutation rate were used. Mutation was based on the ramped-half-and-half initialisation method, which was also used to initialise the population.

Tables 3.1–3.3 include the best and the average fitness values for the whole 1512 runs sorted according to each hypothesis to allow better discussion.

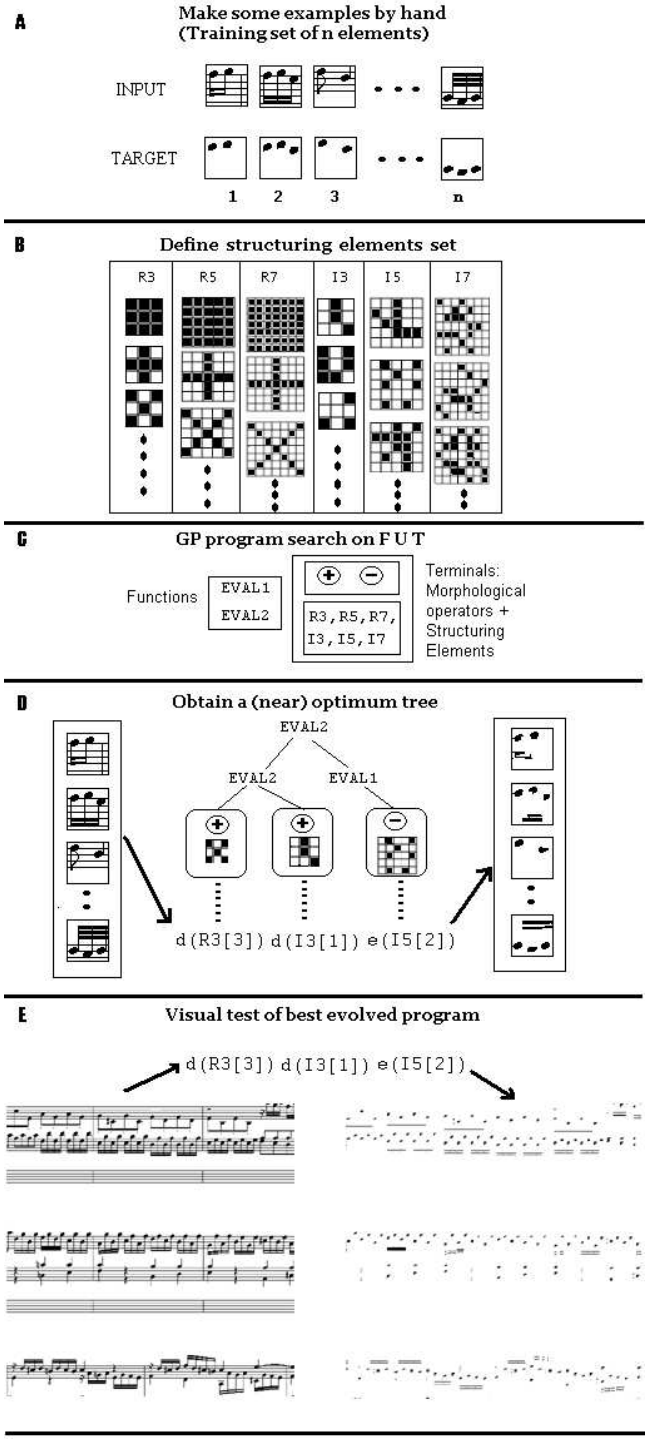


Figure 3.9: Process to obtain an MM algorithm using GP: A) construction of examples by hand, B) choice of structuring elements, C) definition of primitive set, D) running GP obtaining a near optimum tree, E) visual evaluation of best result.

Table 3.1: Best and average numerical fitness values for Structuring Elements type.

	Noteheads				Beams				Staffs			
	f_A		f_B		f_A		f_B		f_A		f_B	
	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg
Structuring Elements type												
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

3.4.2 Testing the *Irregular SEs vs. Regular SEs* Hypothesis

When an MM algorithm is designed by hand, 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 we, as humans, are more likely to understand regularities such as squares, lines, triangles, etc. 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. So, we include also irregular structuring elements (selected randomly) in the GP search space. Analysing Table 3.1 we may observe that the use of irregular kernels provides slightly better performance than the use of regular kernels only for the studied problem.

3.4.3 Testing the *Small SEs vs. Big SEs* Hypothesis

There are no accepted general principles for choosing the size of structuring elements. Human programmers have developed the expertise in combining the use of SEs of different sizes. In order to emulate this characteristic we tested the system performance on structuring elements of sizes 3×3, 5×5 and 7×7 such as those shown in Figure 3.1. In total we included 11 regular and 11 irregular structuring elements of each size.

Analysing Table 3.2 we may observe that every column has a slight improvement when using bigger structuring elements. From this observation we may infer that bigger SEs

Table 3.2: Best and average numerical fitness values for Structuring Elements size

	Noteheads				Beams				Staffs			
	f_A		f_B		f_A		f_B		f_A		f_B	
	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg
Structuring Elements size												
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

allow GP to cope better with the complexities in the training set.

3.4.4 Testing the *Big Training Set vs. Small Training Set* Hypothesis

The number of images in the training set is likely to affect the results of training. Intuitively, on the one hand a bigger number of images should provide more information to the GP system to learn the task and generalise. On the other hand, a bigger number of images may result in a slower learning process and consumes more computer resources. The smallest training set which produces plausible results would be a reasonable compromise. To learn if the number of elements in the training set affects the result quality we used training sets of four different sizes: 1, 5, 15 and 25.

Table 3.3 shows that 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.

3.4.5 Testing the *Big Image Sizes vs. Small Image Sizes* Hypothesis

To find out whether or not the size of the images in the training sets affects the quality of the results, we constructed (by hand) training sets containing images of size 16×16, 32×32 and 64×64 for each one of the different features to look for. The set of images

Table 3.3: Best and average numerical fitness values for Noteheads, Beams and Staffs

	Noteheads				Beams				Staffs			
	f_A		f_B		f_A		f_B		f_A		f_B	
	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg
Training Set Size												
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

Table 3.4: Best and average numerical fitness values for Noteheads, Beams and Staffs

	Noteheads				Beams				Staffs			
	f_A		f_B		f_A		f_B		f_A		f_B	
	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg
Image Size												
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

shown in Figure 3.8 belongs to the 64×64 training set.

The results in table 3.4 show that it is preferable to use bigger training images. For instance, the 64×64 examples produced better results than the others.

3.5 Testing the Generalisation of the Evolved Algorithms

In order to test the ability of GP to generalise we applied the best evolved algorithms to a test set of images. The test set includes ten images with the same resolution of the training set, but this time each image shows a whole music score sheet (see Figure 3.10). The images in the testing set are of size 512×512.

Tables 3.5, 3.6 and 3.7 show the results of a detailed analysis of the best evolved algorithms. The first column in each table represents the image ID, the second column the

Table 3.5: Visual analysis of a notehead detector generated using GP

Notehead detection on test images using $e(R3[3])e(R3[9])d(R3[10])e(I3[9])$.				
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 3.6: Visual analysis of a beam detector generated using GP

Beam detection on test images using $e(R3[0])d(R3[3])e(R3[0])d(R3[8])$.				
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+

number of features actually present in the image, the third column the number of features correctly detected (true positives), the fourth column the number of features mis-detected (false negatives) and the fifth column the number of other regions present in the image (false positives). Note that we are counting regions not pixels; the values of columns $2-5$ were obtained according to a subjective visual counting.

In addition to numerical results returned by the fitness functions, we have analysed visually the results produced by the evolved MM algorithms. Although this analysis was

Table 3.7: Visual analysis of a staff detector generated using GP

Staff detection on test images using $e(R5[9])e(R7[10])d(R7[10])$.				
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+

not exhaustive (i.e. we focused on those MM algorithms that had a high fitness value and those with interesting string features) we observed that fitness function f_B consistently produced programs with well balanced outputs, whereas function f_A often had a tendency to be either overly sensitive or overly specific. The degree of accuracy is very difficult to evaluate visually and, depending on the evaluator, visual quality does not always match the numerical fitness values obtained.

The image in Figure 3.10 corresponds to the image labelled with ID 1 in Tables 3.5, 3.6 and 3.7.

3.5.1 Noteheads

GP easily found several algorithms to solve the *noteheads* extraction problem.Table 3.5 shows the results of the evolved algorithm $e(R3[3])e(R3[9])d(R3[10])e(I3[9])$ applied to 10 test images. The evolved algorithm obtained perfect sensitivity (having detected all true positives without false negatives) and a high specificity value (a small number of false positives). This is considered to be a very good result.

Figures 3.11, 3.12, 3.13 and 3.14 show, step by step, the process of the MM noteheads-

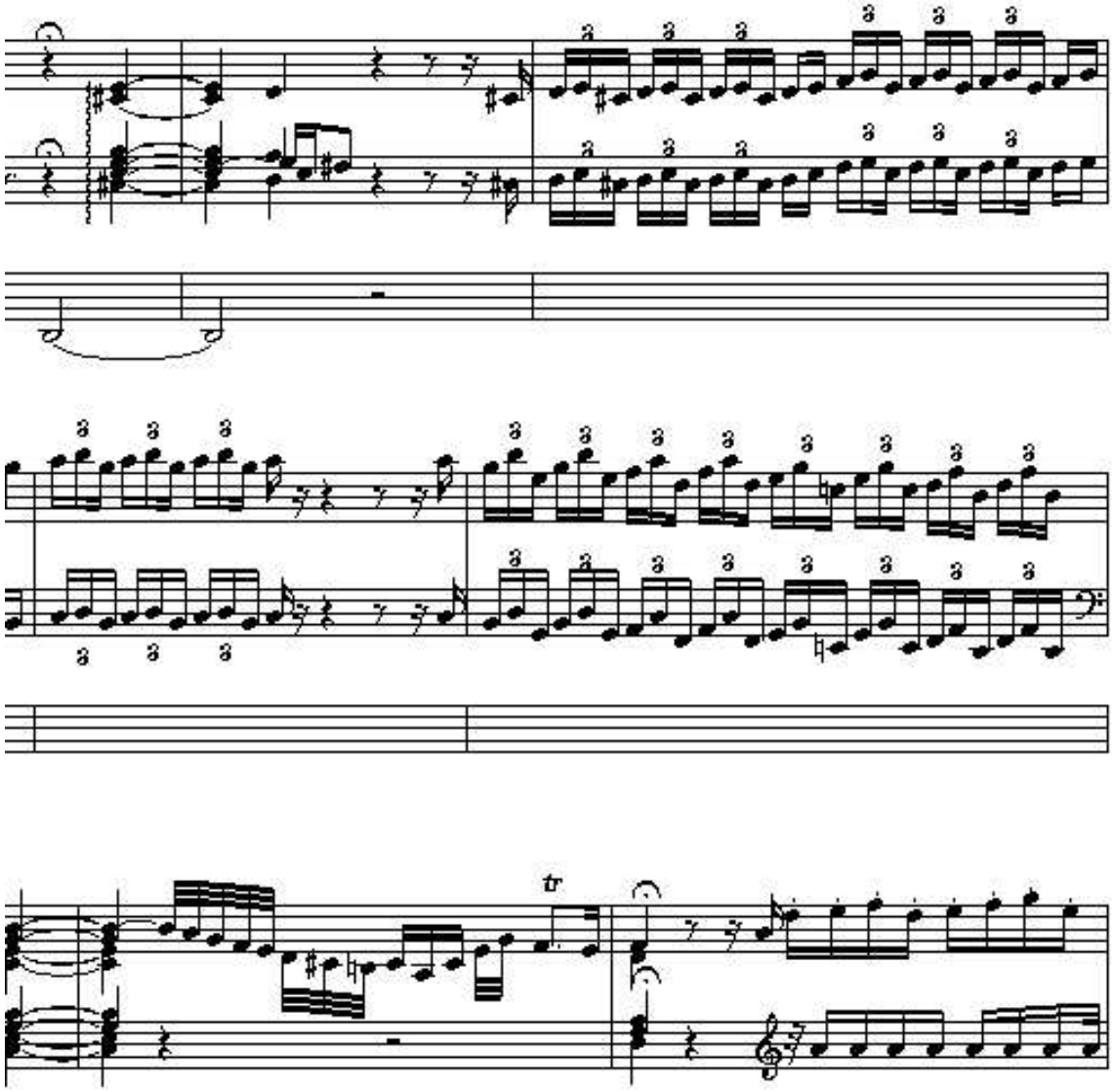


Figure 3.10: Example of testing image not in the training set

detector algorithm generated by GP. The SEs are represented from top to bottom and left to right, e.g. (010;010;010) represents the following SE:

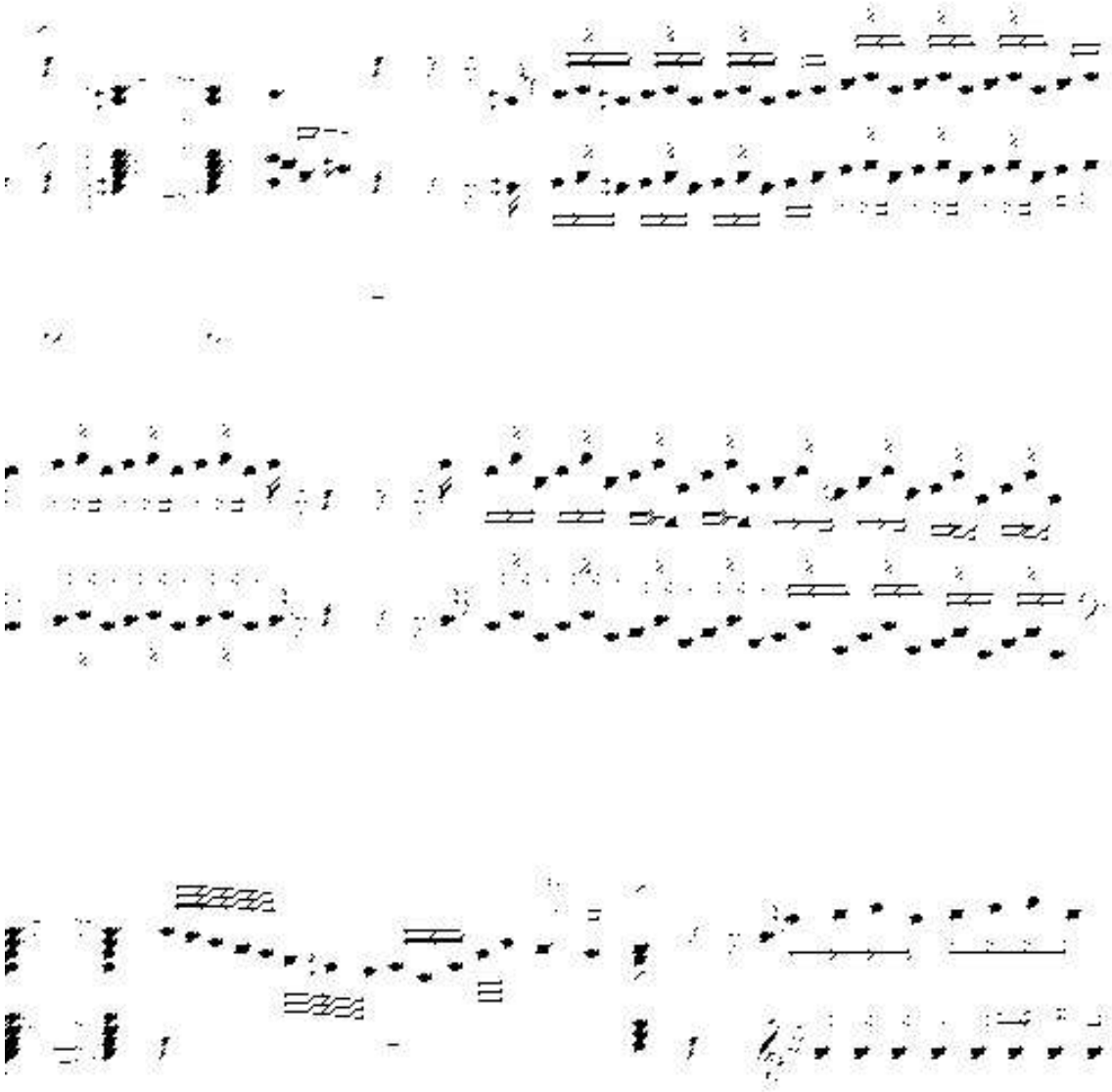
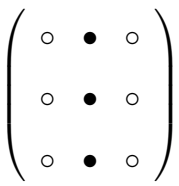


Figure 3.11: Notehead detector step 1: Output generated after e(001;010;100) is applied to the image in Figure 3.10.



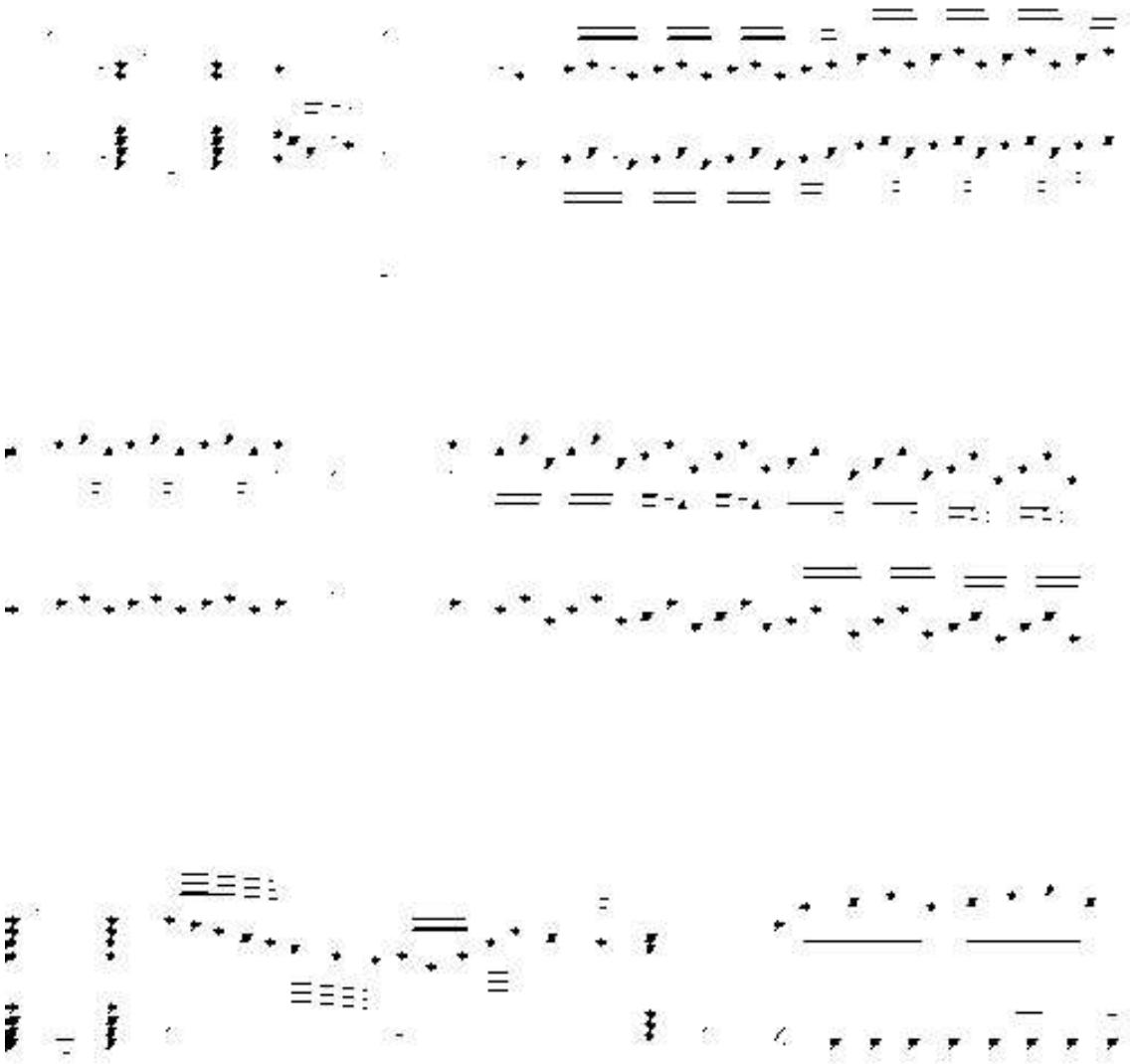


Figure 3.12: Notehead detector step 2: Output generated after e(111;000;000) is applied to the image in Figure 3.11.

3.5.2 Beams

The *beams* extraction problem is difficult. Beams can be mismatched with staffs, noteheads or other features present in a musical sheet. In spite of that, some GP algorithms generated good approximations to the desired outcome. Table 3.6 shows the results of the *beam-*

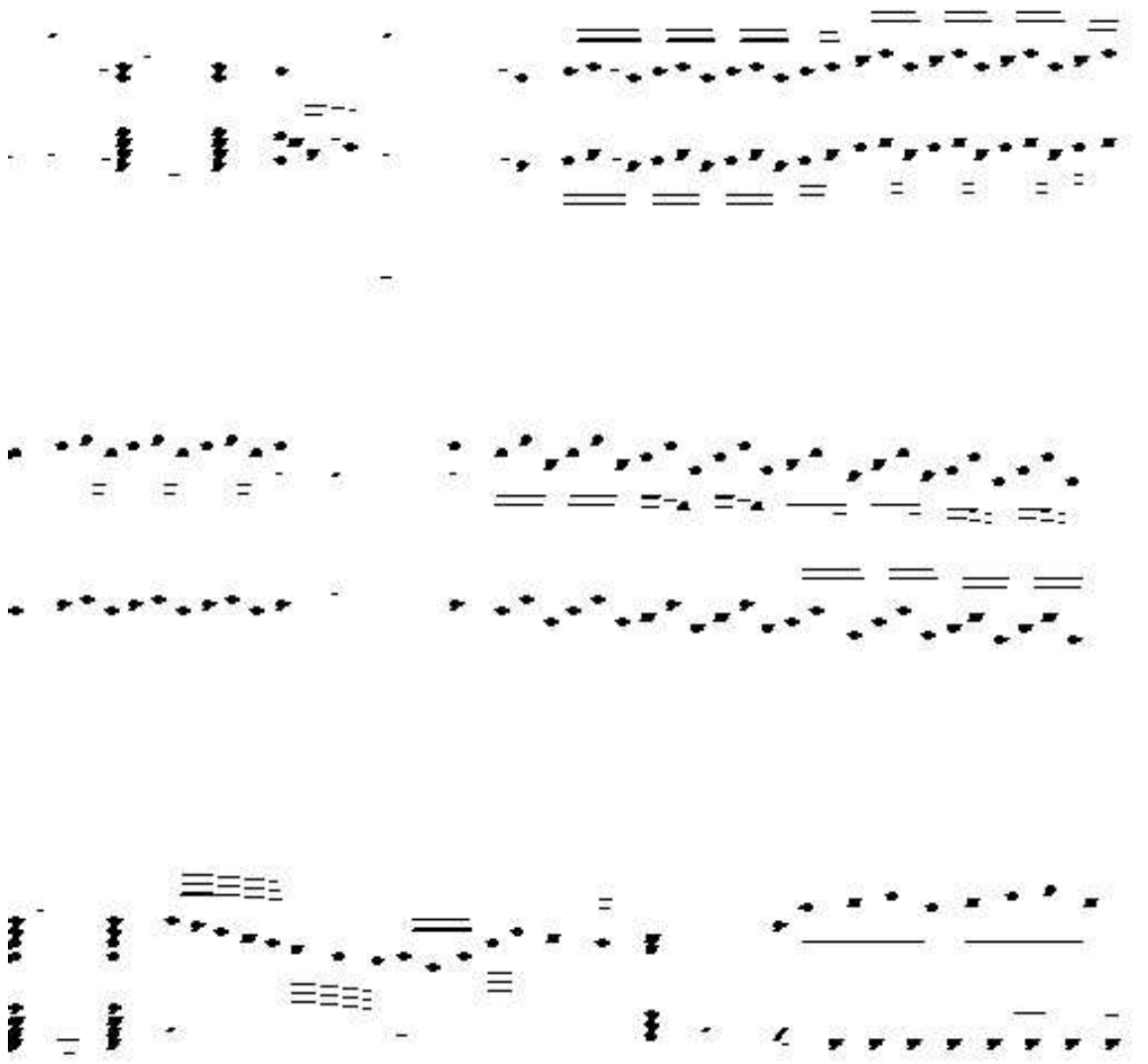


Figure 3.13: Notehead detector step 3: Output generated after d(000;000;111) is applied to the image in Figure 3.12.

detector algorithm $e(R3[0])d(R3[3])e(R3[0])d(R3[8])$ where we may observe a considerable number of mis-detections and a large number of other regions present. Most of these regions are very small (some of them including just 3 pixels). So, even when the number of *Other regions* detected may be rather high, the visual quality of the results is only weakly

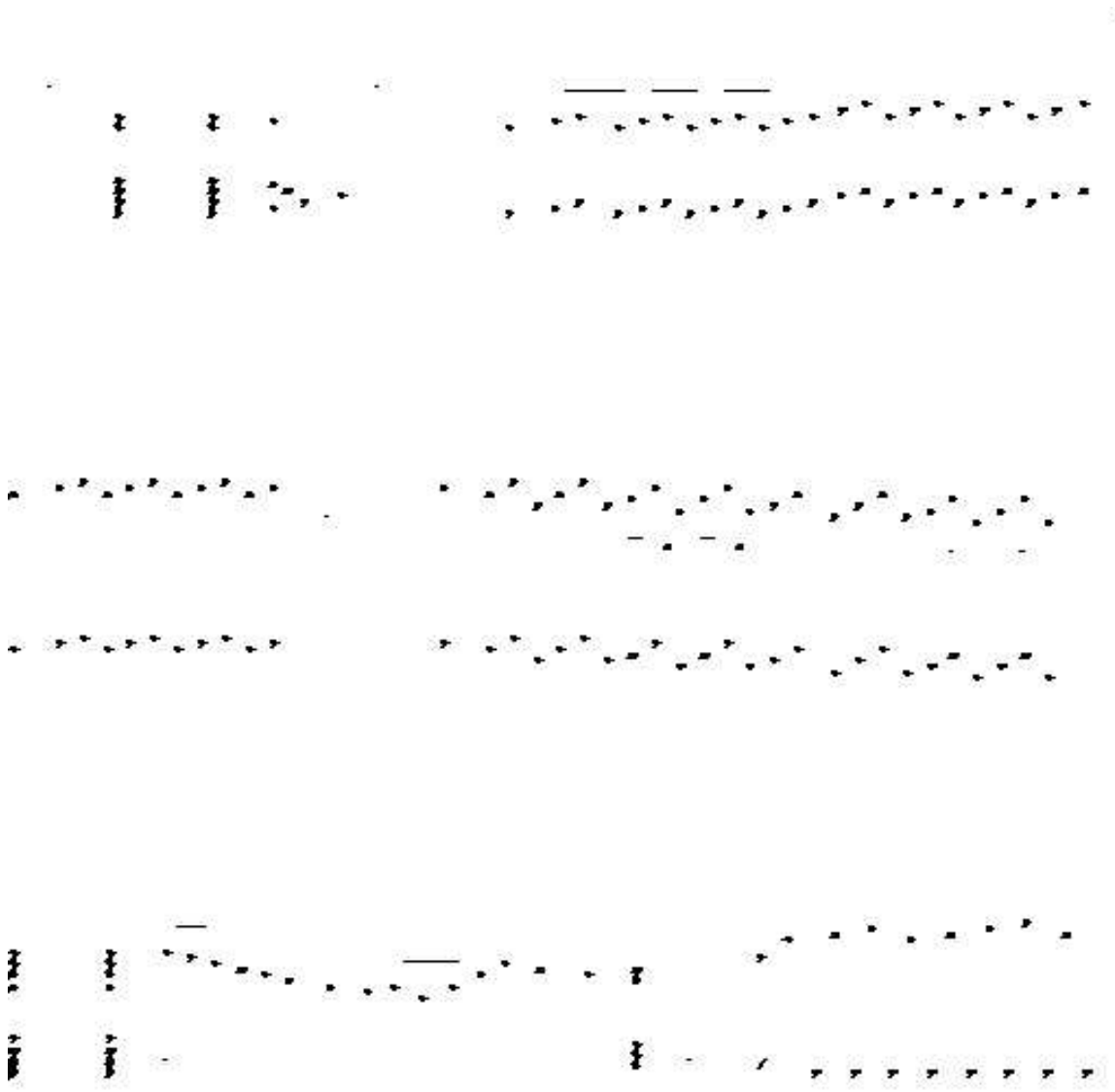


Figure 3.14: Notehead detector step 4: Output generated after e(001;011;000) is applied to the image in Figure 3.13. This is the final output generated by the notehead-detecting MM program e(R3[3])e(R3[9])d(R3[10])e(I3[9]) evolved by GP when applied to the image in Figure 3.10.

affected by these misclassifications. Figures 3.15, 3.16, 3.17 and 3.18 show the sequence of a GP-generated program applied to the image Figure 3.10.

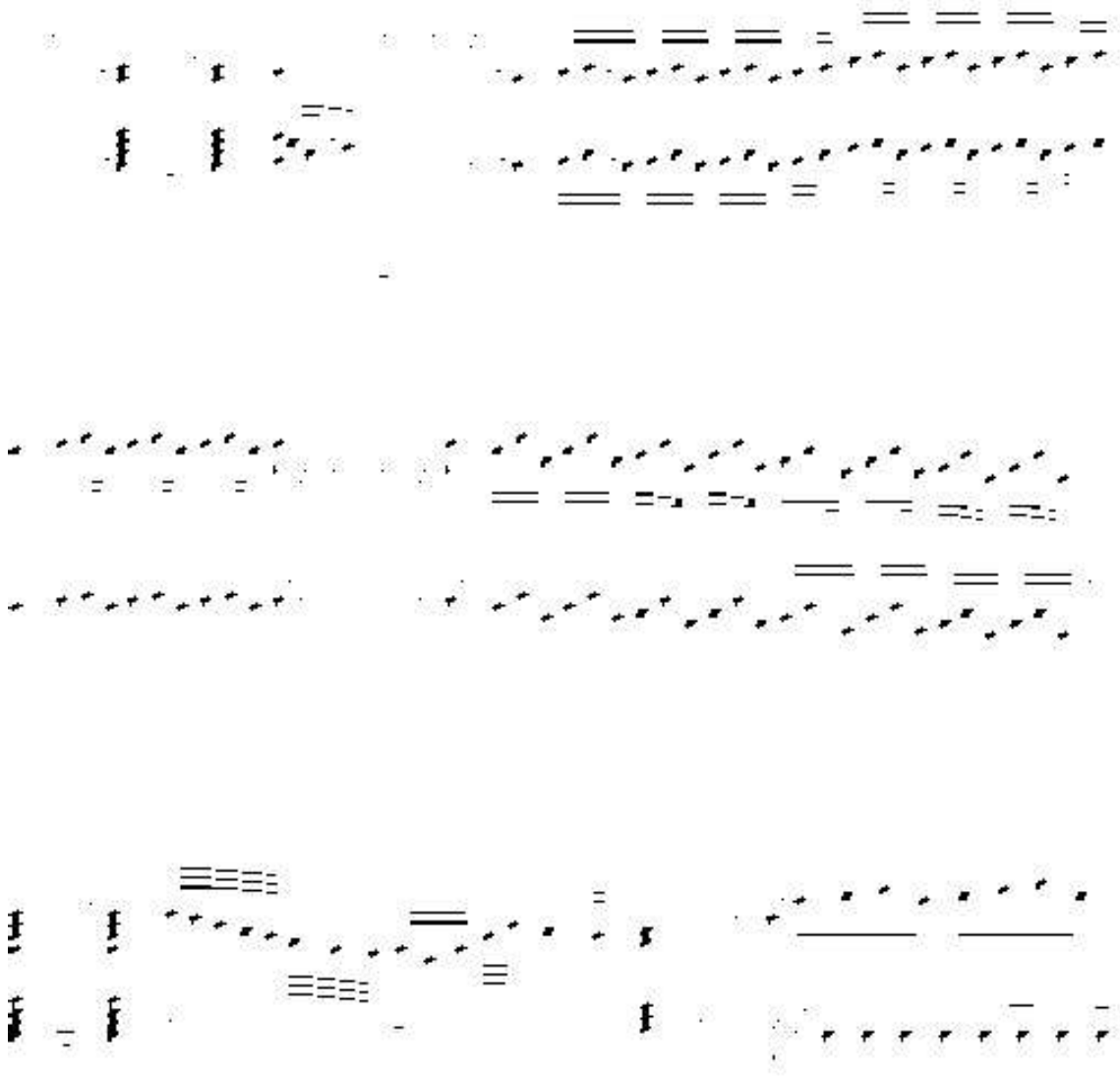


Figure 3.15: Beam detector step 1: Output generated after e(111;111;111) is applied to the image in Figure 3.10.

3.5.3 Staffs

Contrary to expectation, the *staff* extraction problem presented many difficulties for GP using only MM operators. It seems that GP is trying to find the right sequence but the available structuring elements are not enough to perform the task requested. The results

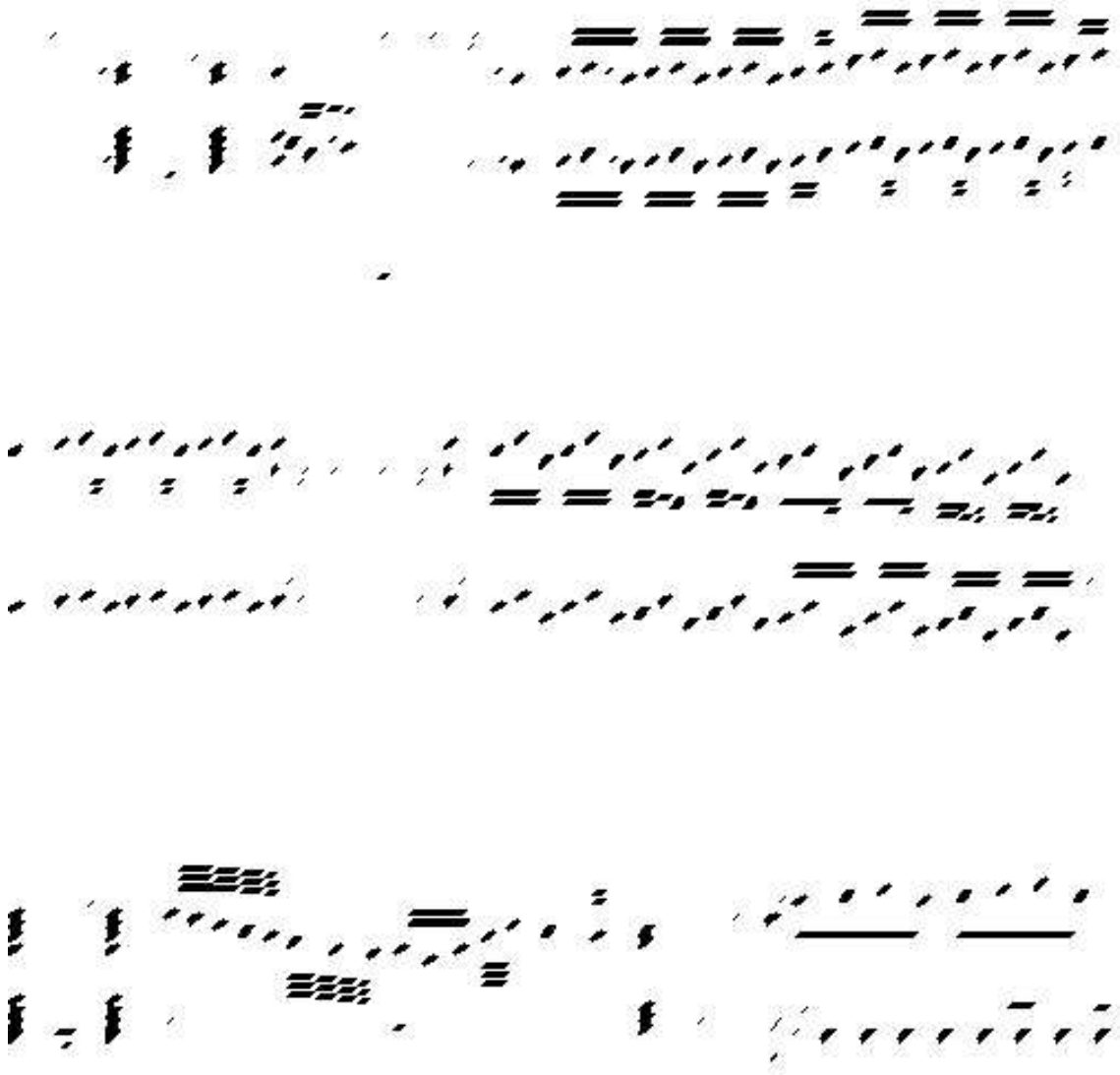


Figure 3.16: Beam detector step 2: Output generated after d(001;010;100) is applied to the image in Figure 3.15.

obtained were either tending to completely white images or showed mismatches between the staffs and other features. Table 3.7 shows the GP *staff-detector* $e(R5[9])e(R7[10])d(R7[10])$ with 100% sensitivity but low specificity. There is a high number of false positives mostly due to beam-like structures that are similar to staffs in the sense that they both are

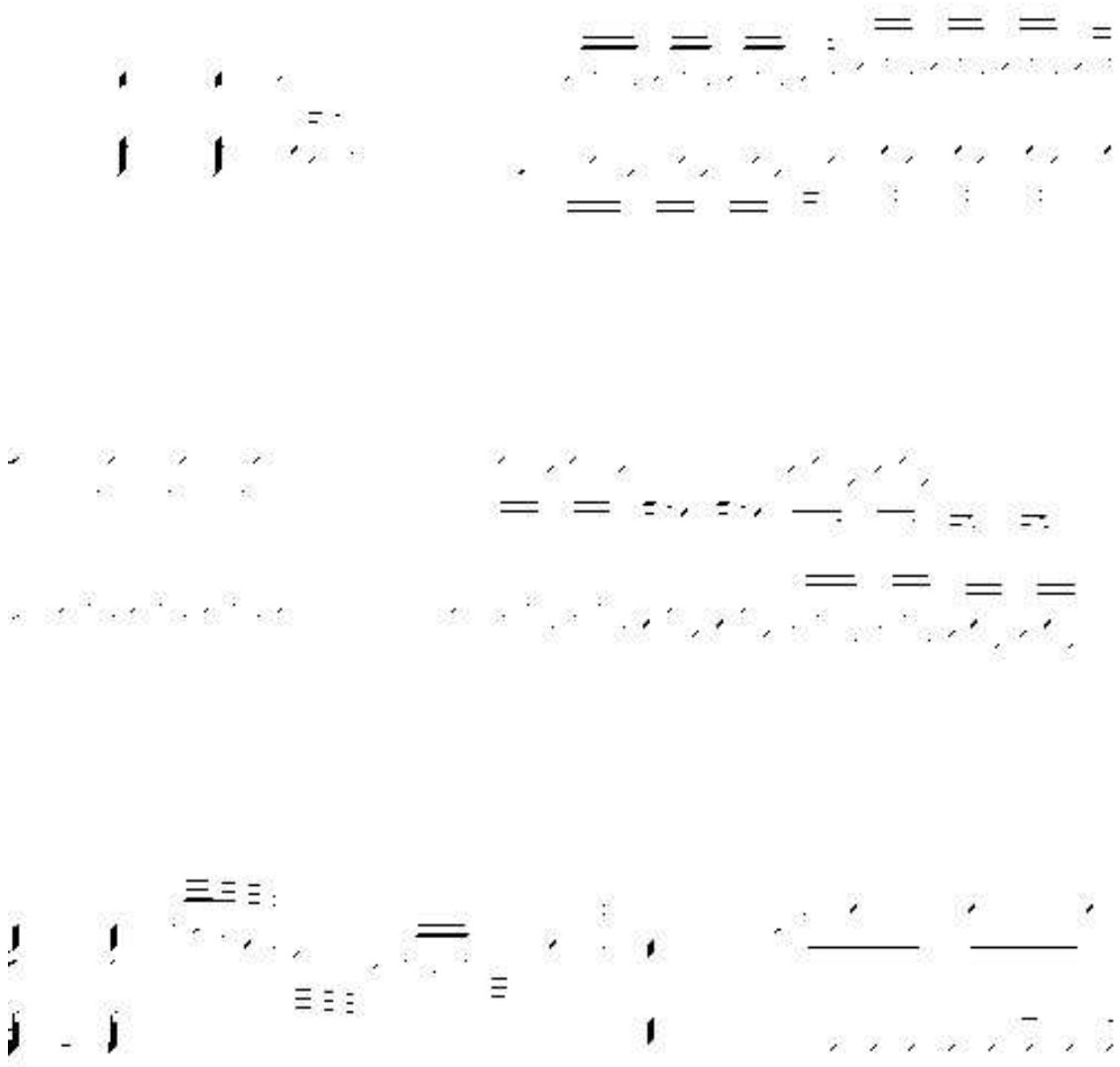


Figure 3.17: Beam detector step 3: Output generated after e(111;111;111) is applied to the image in Figure 3.16.

horizontal features. This is illustrated in Figures 3.19, 3.20 and 3.21 where it is shown step by step how GP accurately finds the staffs, but also finds it difficult to remove most of the beams present in the test image. These results emphasise the need to look for specialised algorithms when we are attempting to detect features that may be confused

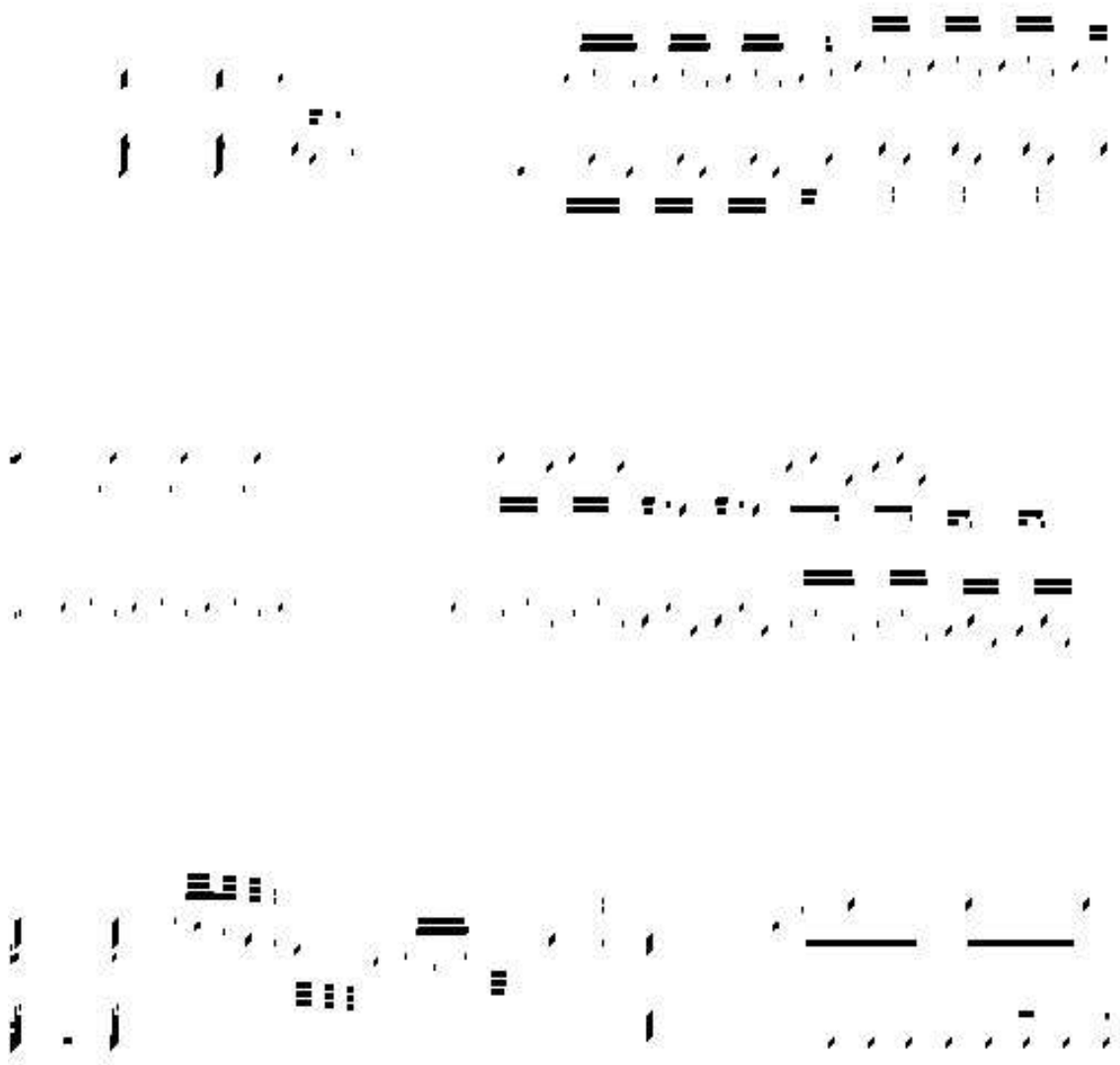


Figure 3.18: Beam detector step 4: Output generated after d(001;001;001) is applied to the image in Figure 3.17. Output generated by the beam-detecting MM program e(R3[0])d(R3[3])e(R3[0])d(R3[8]) evolved by GP when applied to the image in Figure 3.10.

with other features with similar characteristics.

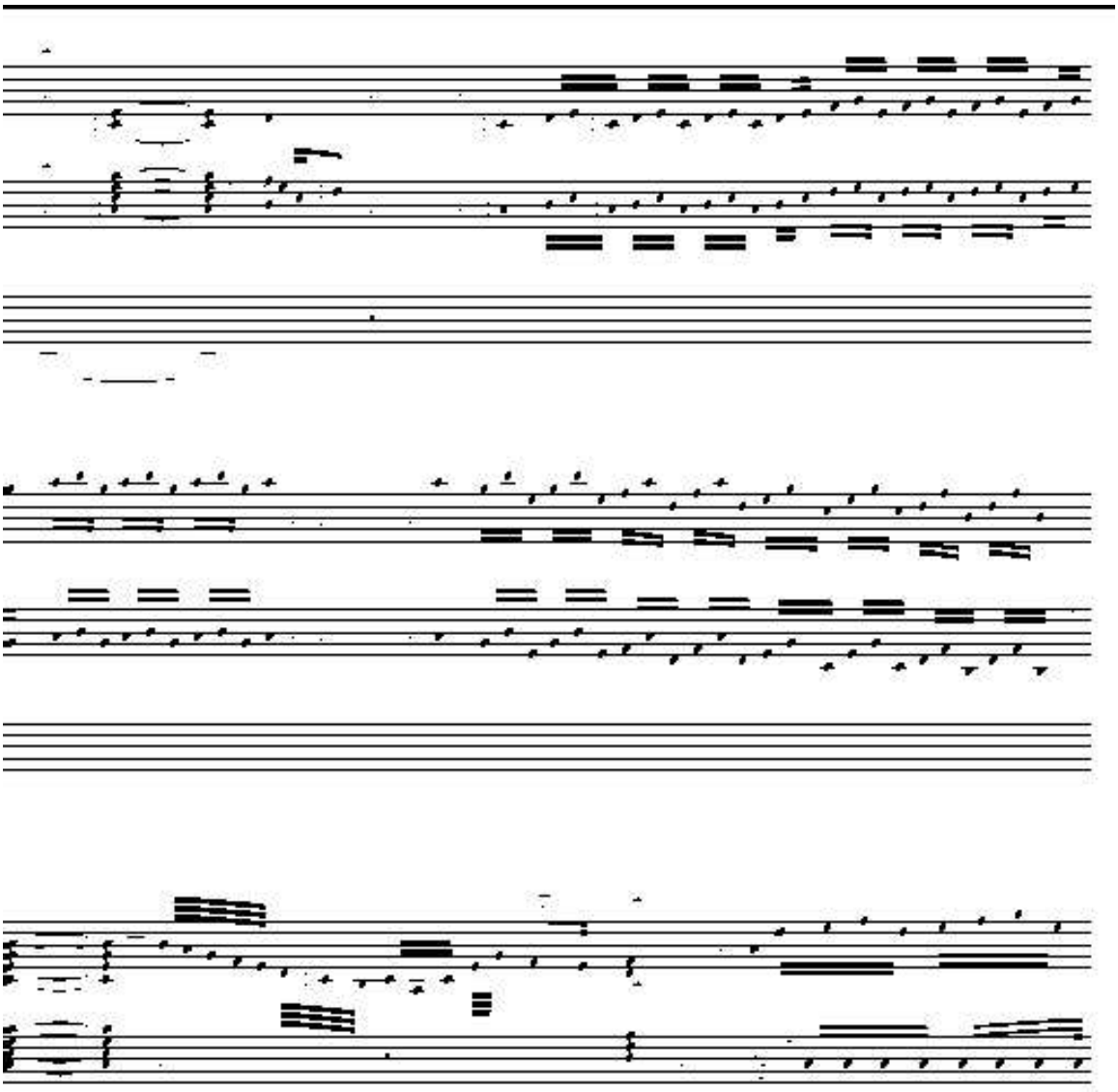


Figure 3.19: Staffs detector step 1: Output generated after e(11111;00000;00000;00000;00000) is applied to the image in Figure 3.10.

3.6 Conclusions

In this chapter we have presented an approach to morphological image processing based on the GP paradigm. We tested several hypotheses in order to get better insights about GP

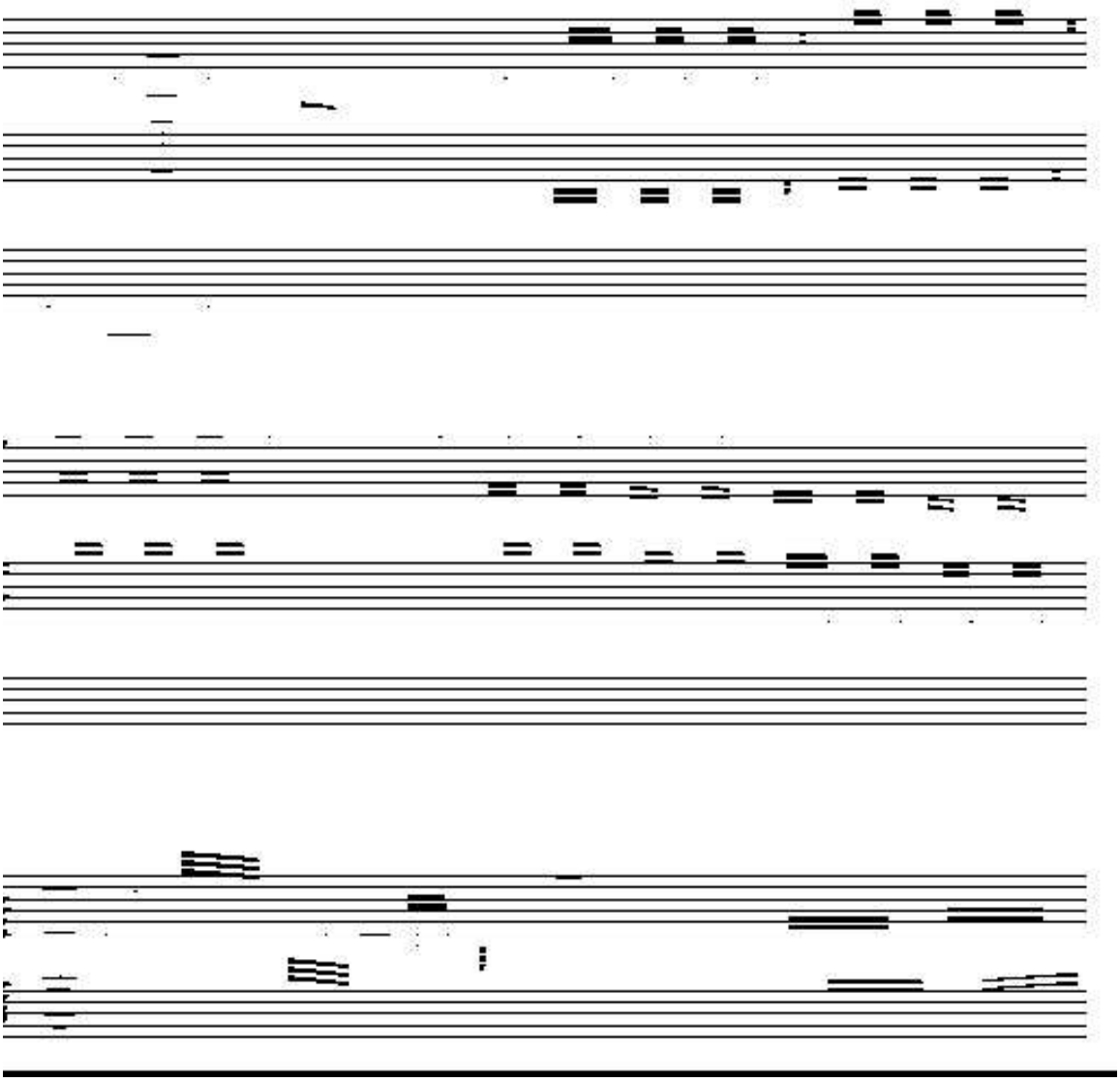


Figure 3.20: Staffs detector step 2: Output generated after e(0000000;0000000;0000000;0000000;0000000;0000000;1111111) is applied to the image in Figure 3.19.

for morphological image processing. We have studied the behaviour of two different fitness functions using a variety of regular and irregular structuring elements. We have applied this approach to the extraction of three features in musical score sheets. In our approach we have avoided using human expertise (i.e. the three different MM feature extraction

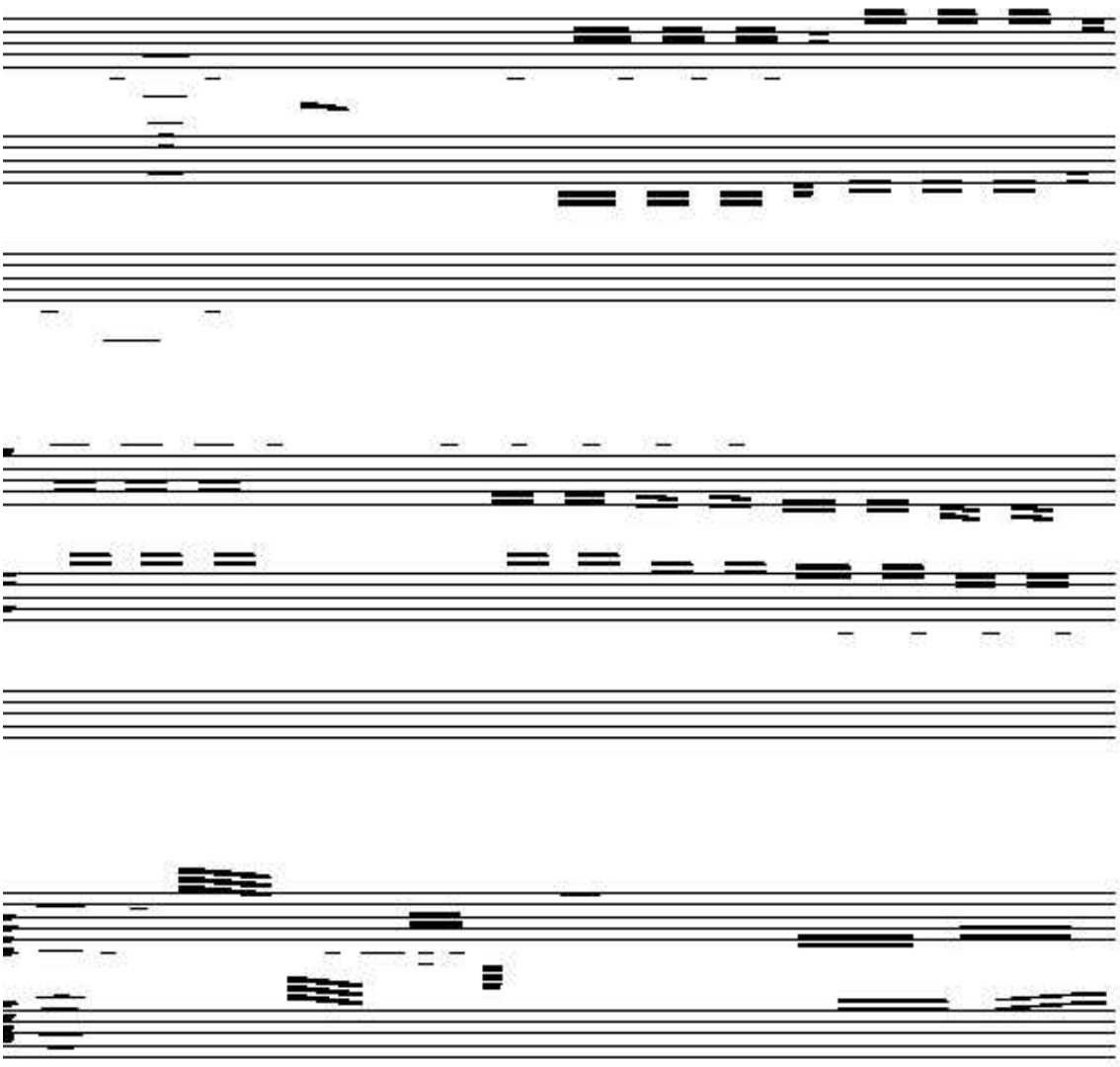


Figure 3.21: Staffs detector step 3: Output generated after d(0000000;0000000;0000000;0000000;0000000;0000000;1111111) is applied to the image in Figure 3.20. Final output generated by the line-detecting MM program e(R5[9])e(R7[10])d(R7[10]) evolved by GP when applied to the image in Figure 3.10.

algorithms were obtained using GP in exactly the same way). We have also investigated how the size of the training set of images and their number affect the results quality.

We have observed that GP is indeed capable of evolving MM algorithms. As a result of

this work we have identified a set of promising directions for further research. For example, the use of irregular structuring elements has proved to be useful. Bigger image sizes in the training sets provide better results. The number of images in the training set did not affect the results noticeably. We have also concluded that fitness function f_A has a tendency to maximise either sensitivity or specificity, which is undesirable. The fitness function f_B is better in this respect but we believe there is still scope for improvement.

3.6.1 Limitations

Although the results obtained in this chapter are promising, there are several lines of research that could be explored to improve the results. Perhaps the most restrictive problem is the amount of computational resources needed to implement the GP system on morphological images. Due to the heavy computation and memory loads required in GP for image analysis (see also Poli and Cagnoni [106]), all the experiments were performed using a population of 50 individuals run for 100 generations. The number of evaluations was relatively small to be able to get a true conclusion about the real potential of GP for morphological image processing.

In the MM approach described in Section §3.4 the sizes of the images in the training set were 16×16, 32×32, 64×64 and 128×128. The number of images in the training set were 1, 5, 15 and 25 in different experiments. All the experiments were performed using 5000 fitness evaluations. In GP much bigger populations (around 50000) are usually considered over a larger number of generations (around 1000). In our preliminary experiments (using bigger populations) very often the programs were terminated by the system before finishing because of memory overload.

In order to evaluate how the population size and the number of images (size 32×32) in the training set affected the maximum number of generations accepted before the process is killed we made a total of 720 GP runs combining 3 features to extract (heads,

Training Set Size	Population Size		
	500	1000	5000
1	1000	500	100
5	350	180	35
10	180	90	25
15	120	60	10

Table 3.8: Estimated generation when a process is terminated.

hooks and lines), 4 different numbers of training set elements (1, 5, 10, 15), 3 different population-sizes/number-of-generations combinations (500/1000, 1000/500 and 5000/100) and 20 different random seeds. The results are shown in Table 3.8.

It is evident that the number of images in the training set affects considerably the maximum number of generations. If we consider that each image is represented as a matrix of 16×16, 32×32 or 64×64 bytes (i.e. 256, 1024 or 4096 bytes each image), and we consider that GP needs to generate many internal images to represent different branches of the trees during the evolution, then a possible explanation to the fast memory overload is the image encoding. Chapter §4 presents a possible solution to this problem.

Another important issue is the loss of information due to long sequences of the same morphological operator. A long sequence of erosions will provide a completely white image and a long sequence of dilations will provide a completely black image. In either case the morphological operators are not able to proceed further in the search of improved alternatives. In Chapter §5 this issue is re-visited and a way forward proposed.

3.7 Chapter Summary

Experiments whose role was to investigate the feasibility of using the GP paradigm for morphological image processing for binary images were presented in this chapter. The experiments have shown that GP is indeed able to evolve MM algorithms. There are some problems with the current approach. One of them is the need for very large computational

resources. Chapter §4 aims to tackle this problem by means of the GP technique known as Sub-Machine-Code-GP (SMCGP). Another problem is that the consecutive use of the same morphological operator in an MM algorithm causes the loss of image information. Chapter §5 aims to solve this problem by combining MM operators with logical operators.

Chapter 4

Sub-Machine-Code GP Approach to MM

Chapter Outline

Sub-Machine-Code GP (SMCGP) is a technique to speed up GP and to extend its scope based on the idea of exploiting the internal parallelism of sequential CPUs. This chapter presents a SMCGP approach to obtain algorithms for binary images. SMCGP is used to find programs to convert a binary image into another containing just a particular characteristic of interest. The results reported are compared with the GP approach to obtain MM algorithms for binary images presented in Chapter §3.

4.1 Sub-Machine-Code GP

GP is usually quite demanding from the computation load and memory use point of view. One of the ideas that has been applied to improve the performance of GP is SMCGP [107, 104]. SMCGP is a GP variant aimed to exploit the intrinsic parallelism of sequential CPUs. SMCGP extends the scope of GP to the *evolution of parallel programs running on sequential computers*. These programs are faster as, thanks to the parallelism of the CPU, they perform multiple calculations during a single program evaluation. SMCGP presents the idea of computing multiple fitness cases in parallel during a single execution of a program. The potential of SMCGP is already available and very promising.

Some instructions in a CPU are performed in parallel and independently for all the bits in the operand. For example, the bitwise AND operation is performed internally by the CPU activating concurrently a group of AND gates within the arithmetic logic unit (see Figure 4.1).

If we see the CPU as a SIMD computer we can imagine that each of its 1-bit processors is able to produce a result after each instruction. Most of the CPUs do not allow handling single bits directly. All the values are usually loaded into the CPU and the results produced are packed into bit vectors.

The differences from a normal GP system are that in a SMCGP system:

* The function set should include operations which exploit the parallelism of the CPU.

* The result produced by the evaluation of a program should be interpreted as a bit vector. Each bit of this vector represents the result of a different 1-bit processor.

* The terminal set could include input variables and constants. These should also be interpreted as bit vectors where each bit represents the input to a different 1-bit processor.

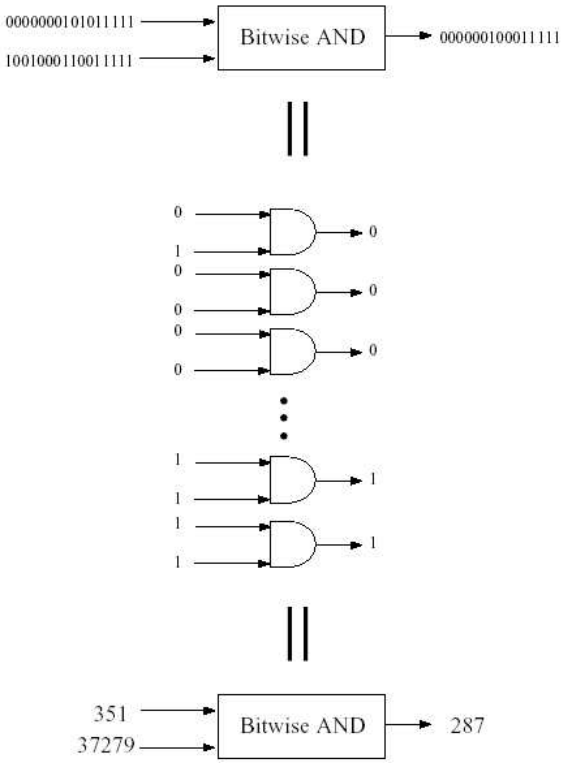
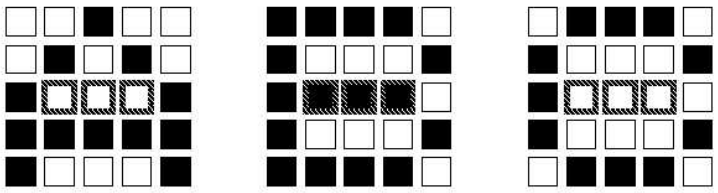


Figure 4.1: Three different ways to look at bitwise AND using a 16-bits CPU processor. From [107]

4.2 Sub-Machine-Code GP approach to Image Processing

In their first paper on SMCGP, Poli and Langdon [107] suggested an application for character recognition re-framed as classification. They demonstrated the impressive potential of SMCGP for this task. Figure 4.2 shows the original bitmaps, with their corresponding bit-strings and the SMCGP generated subexpressions showing accurate classification.

Adorni *et al.* have suggested the use of SMCGP for the efficient design of low-level vision programs [4, 3]. The applications include SMCGP approaches to plate detection and character recognition. The results obtained show that SMCGP is able to generate programs that are both accurate and efficient.



A=00100010101000011111110001
B=11110100011111101000111110
C=01110100011000010000101110

Subexpression	Input pattern A	B	C
x	00100010101 **000** 11111110001	11110100011 **111** 01000111110	01110100011 **000** 01000101110
(SL (SL (SL x)))	00010101000 **111** 11110001000	10100011111 **010** 00111110000	10100011000 **010** 00101110000
(NOT (SL (SL (SL x))))	11101010111 **000** 00001110111	01011100000 **101** 11000001111	01011100111 **101** 11010001111
(SR x)	00010001010 **100** 01111111000	01111010001 **111** 10100011111	00111010001 **100** 00100010111
(XOR (NOT (SL (SL (SL x)))) (SR x))	11111011101 **100** 01110001111	00100110001 **010** 01100010000	01100110110 **001** 11110011000

Figure 4.2: A,B and C bitmaps with their corresponding bit-strings. Below there are some subexpressions and their corresponding bit-string. The last subexpression presents correct classification. From [107]

The idea of this chapter is to apply SMCGP to MM. As explained in Section §2.3.1, erosion and dilation may be derived from the low-level set operators Union, Intersection, Translation and Reflection. For instance, a dilation may be derived from translation and intersection according to Eq. §2.3. This is trivial for $1D$ MM operators, but may be non-trivial for $2D$ and higher dimensions operators and it is recommended as a subject for further research.

4.3 Experiments

We have experimented with SMCGP to evolve transformation algorithms for the same music score sheets as used in Chapter §3. We aim to transform an original image into a goal image containing only a particular feature. The features analysed are noteheads, beams and staffs.

We noted that the binary image representation was suitable for improvement. An alternative representation may be obtained using bitwise representation. Instead of using

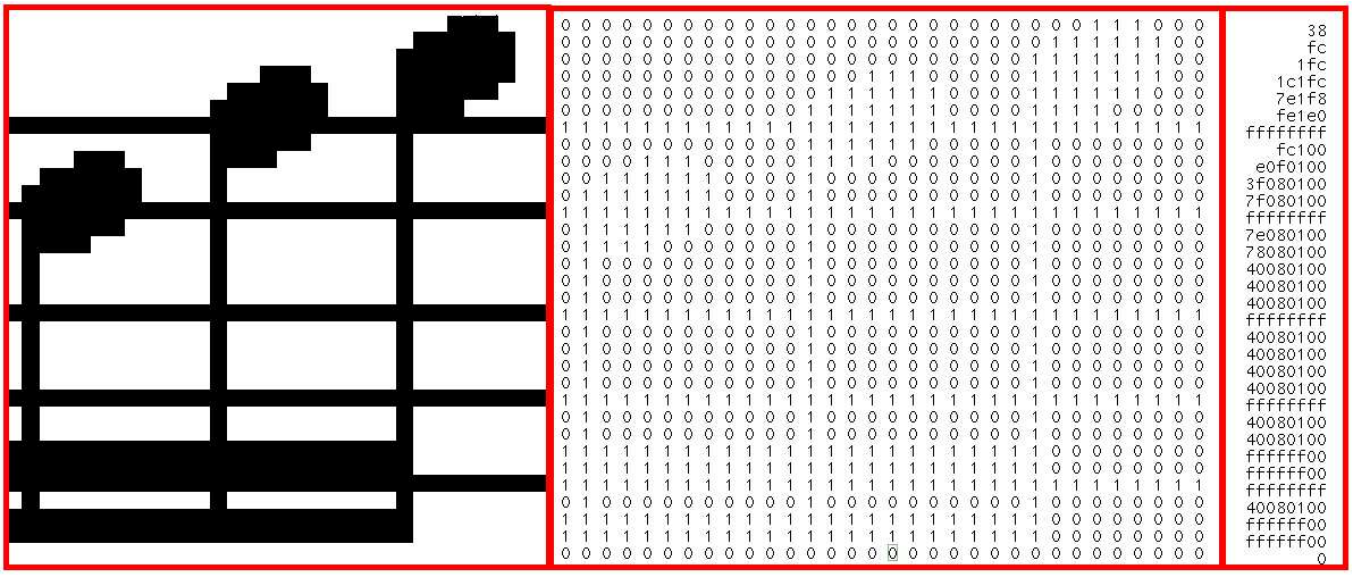


Figure 4.3: Binary image (left). Representation with 1024 bytes, 8 bits per pixel (centre). Representation with 32 words, 1 bit per pixel (right).

Functions		
Arity	Name	Function
2	AND	Bitwise AND
2	OR	Bitwise OR
1	NOT	Bitwise NOT
2	XOR	Bitwise XOR
1	SL	Bitwise shift left
1	SR	Bitwise shift right

Terminal	
X[1]	One binary image
X[2]	represented using 32
...	unsigned long integers
X[32]	(One row each.)

Table 4.1: Functions and terminal used in the SMCGP approach.

a complete 32×32 image represented with 1024 bytes (8 bits per pixel) we propose to represent the same binary image in a 32 word representation (1 bit per pixel). Figure 4.3 shows an example, the image on the left is represented with 1024 bytes, the same image is represented in the centre with $1s$ and $0s$ and the same image is represented using word representation on the right (hexadecimal codification).

For the experiments we used populations of 500, 1000 and 5000 programs over 1000, 500 and 100 generations respectively (to fix the number to 500000 performance evaluations

in each run) using 15 different random seeds. GP was setup using *Lilgp* with a 0.9 crossover rate, tournament selection and 0.1 mutation rate using a *half and half* initialisation method.

Note that encoding the images using unsigned long integers allows to take advantage of the SMCGP paradigm. Each image will be represented as a vector X of 32 unsigned long integers, each item of the vector (32 bits) represents a row in the image. We set up a preliminary experiment to find out whether it is possible to obtain acceptable results using some of these operators on bitwise level, namely union (AND, $\cup$), intersection (OR, $\cap$) and translation (*shift left* (SL) and *shift right* (SR)), complement (NOT) and exclusive-or (XOR) i.e. we use a Function and Terminal set as presented in Table 4.1. The bitwise operations are applied to all the vector items. This means that a logical operator on a 32×32 image requires only 32 bitwise operations.

The fitness function used is *Resemblance* f_b (Eq. 3.3), so the program is evaluated according to the output provided by the SMCGP evolved programs (note that X is evaluated as a whole and not each bit separately like in other SMCGP implementations). We made a total of 3 SMCGP sets of 540 runs combining 3 features to extract (noteheads, beams and staffs), 4 different numbers of training set elements (1, 5, 10, 15), 3 different population-sizes/number-of-generations combinations (500/1000, 1000/500 and 5000/100) and 15 different random seeds to make a total of 1640 runs.

4.4 Results

SMCGP approach impressively speeds up the evolution when the results are compared with those of Chapter §3 (also presented in [110, 111]). This approach is original because SMCGP has not been used before for morphological image processing. When the comparison is possible (recall that in §3.6.1 we remark that GP and MM can not evolve so many generations) the SMCGP approach is up to 10 times faster. Table 4.3 shows a time comparison for some notehead experiments (populations of 500, 1000 and 5000; training set size 1 and 5; times shown in seconds). Experiments for beams and steams presented

Table 4.2: Best Numerical Fitness Values per Population Size.

	Noteheads			Beams			Staffs		
	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
Population Size									
500	0.9587	0.9616	0.9587	0.7977	0.7649	0.7477	0.9368	0.9368	0.9281
1000	0.9608	0.9727	0.9608	0.7715	0.7833	0.7567	0.9368	0.9354	0.9306
5000	0.9541	0.9561	0.9383	0.7575	0.7717	0.7594	0.9399	0.9398	0.9392

Table 4.3: Time Comparison for Noteheads (Seconds).

	Noteheads					
	500		1000		5000	
	1	5	1	5	1	5
MM	99.73	1079.98	77.15	494.16	124.35	716.92
SMCGP	25.54	60.92	33.50	57.01	21.58	56.33

similar behaviour. The SMCGP approach is also able to evolve larger populations for a longer number of generations. Indeed, the SMCGP approach finishes approximately 90% of the runs properly while the GP and MM approach under the same circumstances hardly run over all the training sets (see Table 3.8).

Table 4.2 shows the best fitness function values per population size for each set of 540 runs. Comparing the values of Table 4.2 to the best fitness function values of columns f_B in Tables 3.1–3.3 we observe that the fitness function values obtained for SMCGP are better in all cases. However, when the resulting image output of the program was analysed visually we observed that the images were completely white (i.e. the programs have a tendency to maximise the number of True Negatives).

Figures 4.4, 4.5 and 4.6 show the programs with the highest fitness values for noteheads, staffs and beams respectively. When applied to any image they lead to either the original image or completely white images (where 0 refers to white). This observation is confirmed by the analysis of the program code. A simple way to review this is to simplify all the boolean expressions, e.g.$(AND(X, X)) = X$, $(XOR(0, X)) = X$, $(XOR(X, X)) = 0$, etc.

Let us take as an example the program in Figure 4.4. The program may be simplified step by step as follows:

* (XOR (AND X X) (XOR (XOR X X) (XOR (XOR (XOR X X) (XOR X (XOR (XOR X X) X))) (AND (XOR X (XOR X (AND X X))) X))))
* (XOR *X* (XOR *0* (XOR (XOR *0* (XOR X (XOR *0* X))) (AND (XOR X (XOR X *X*)) X))))
* (XOR X (XOR 0 (XOR (XOR 0 (XOR X *X*)) (AND (XOR X *0*) X))))
* (XOR X (XOR 0 (XOR (XOR 0 *0*) (AND *X* X))))
* (XOR X (XOR 0 (XOR *0 X*)))
* (XOR *X* (XOR *0 X*))
* (XOR *X X*)
* *0*

4.5 Limitations of the SMCGP approach to Image Processing

The evolved programs tend to maximise the number of true negatives which is not desirable. The fitness function used is defined to maximise both sensitivity and specificity and it is unlikely that its form is the sole cause of this problem. The most likely reason for the observed behaviour is that the features (represented by black pixels) are small in comparison with the total image area. It is therefore easier for GP to maximise the number of true negatives. Alternatives to solve this problem are to use a more specific fitness function or to allow shifts *up* and *down* as well as *left* and *right*.

```
=== BEST-OF-RUN ===
              generation: 334
                   nodes: 29
                   depth: 7
                    hits: 0
TOP INDIVIDUAL:


-- #1 --
                    hits: 0
             raw fitness: 0.9720
    standardized fitness: 0.0280
        adjusted fitness: 0.9727
TREE:
 (XOR (AND X X)
      (XOR (XOR X X)
           (XOR (XOR (XOR X X)
                     (XOR X
                          (XOR (XOR X X) X)))
                (AND (XOR X
                          (XOR X
                               (AND X X))) X))))
```

Figure 4.4: Example of a run with high fitness for noteheads extracted from *Lilgp*.

The design of a better fitness function should also take into account the fact that the same operator is applied to *all* the lines in the image and not all the lines contain the same information nor they have the same characteristics of input and output. Due to the use of $1D$ logical bitwise operators the programs obtained are unlikely to obtain good visual results when used in $2D$ images with a different size.

4.6 SMCGP approach Conclusions

We have presented an SMCGP approach to speed up the evolution of algorithms for binary images. We have used logical bitwise operators as the GP Function Set. The algorithms have not produced visually correct results in spite of the high fitness function values achieved by evolution.

The fitness values for the training set are better than those obtained in the GP and

```
=== BEST-OF-RUN ===
              generation: 46
                   nodes: 41
                   depth: 11
                    hits: 0
TOP INDIVIDUAL:


-- #1 --
                    hits: 0
             raw fitness: 0.9361
    standardized fitness: 0.0639
        adjusted fitness: 0.9399
TREE:
 (AND (AND (XOR (SR (AND X
                          (SR (AND (SL (SL X))
                                   (XOR X
                                        (AND (NOT (SL X))
                                             (XOR (SL X) X)))))))
                (AND X
                     (AND X X))) X)
      (OR (AND (OR X X)
               (AND X X))
          (SL (XOR (SL (NOT X)) X))))
```

Figure 4.5: Example of a run with high fitness for staffs extracted from *Lilgp*.

MM approach. Even though the speed up is very promising the obtained algorithms with SMCGP cannot be generalised, in contrast to those obtained with the GP and MM approach. This may be due to the fact that bitwise operators, as we code them are $1D$ and take into account only $1D$ (horizontal) features but completely ignore the vertical features and the overall $2D$ coherence of the image data. The speed-up obtained with the use of SMCGP offers significant advantages for the development of MM algorithms and further research should be carried out to overcome the problems encountered in the initial investigation described in this chapter.

4.7 Chapter Summary

This chapter presented an attempt to solve the problem of evolving programs for binary images using large GP populations due to very large computer resources (time and memory)

```
=== BEST-OF-RUN ===
              generation: 95
                   nodes: 7
                   depth: 4
                    hits: 0
TOP INDIVIDUAL:


-- #1 --
                    hits: 0
             raw fitness: 0.7464
    standardized fitness: 0.2536
        adjusted fitness: 0.7977
TREE:
 (XOR (NOT (XOR (NOT X) X)) X)
```

Figure 4.6: Example of a run with high fitness for beams extracted from *Lilgp*.

required for evolutionary techniques. The approach presented is based on the SMCGP paradigm. The results show an improvement in the speed of the evolution. However, the evolved algorithms tend to maximise the true negatives. Chapter §5 explores the idea of using both morphological and logical operators to evolve programs for binary images and aiming for a better quality of the GP algorithms obtained.

Chapter 5

Genetic Programming without the User Feedback

Chapter Outline

This chapter presents a preliminary experiment which explores to what extent the image analysis programs can be evolved without any user feedback. In Chapters §3 and §4 the user provides a visual feedback by means of visually selecting the *best* results obtained. In contrast, this chapter studies what fitness function can correspond to visual outcomes. Experimental set-up is described in Section §5.2. The terminal set includes both morphological and logical operators. The results are presented in Section §5.3. The conclusions of these experiment are presented in Section §5.4 where a brief analysis is also provided.

5.1 User as a component of a fitness function

Chapter §3 described experiments and presented results which demonstrate that GP can evolve MM programs which, when executed, generate good quality results on fairly complex images. The results presented in the thesis and the discussion that followed focused on the *best* results obtained using GP. One of the points made in the discussion was that the high value of a fitness function did not always correspond to the *visual* quality of the results as assessed by the user. It has become clear that the user feedback has played a key role in the selection of the best results, which, in turn, affected the choice of a fitness function, the conclusions about the size and the choice of the SEs, image size, and all other parameters. This concern was further enhanced by results of the experiments in Chapter 4 where the programs evolved using logic operators, without user intervention, resulted in visually poor results in spite of high values returned by the fitness function.

Stimulated by these observations, we have decided to carry out a short exploratory work attempting to get an insight into the question "What fitness function corresponds better to the visual outcomes?".

In the experiments we have used a fitness function which, in place of the user, can bias the search towards either specificity or sensitivity (but without explicit user intervention). Secondly, instead of using several small images with hand-selected target features, we have used just one training image with the features appearing in different configurations and contexts. We have opted for difficult images in which the features resemble one another. Finally, as a terminal set we have used both MM and logic operators. The reason for this was that in "real life" programming it is very rare to use MM operators alone. Given the level of difficulty of the test images, the inclusion of both kinds of operators would give GP a better "chance" to evolve quality solutions.

5.2 Experiments

5.2.1 Image Data

Training Set

In order to test GP in a more complex environment than the one presented in Chapter §3 we used images of size 640×480. Each training set includes two images, one source image and one target image both of size 640×480. The source image includes four different features, namely squares, circles, stars and rings. Each source image contains 20 features of each type to make a total of 80. The features are distributed randomly in the image and they may overlap to make the detection more complex. In order to test how well GP is able to cope with different resolutions we use 5 different feature sizes. Figures 5.1, 5.2, 5.3, 5.4 and 5.5 show examples of the five different source images used in the training set. The target image includes only one of the four features (e.g. Figure 5.6 shows the square target image of the source image shown in Figure 5.3). The images are synthetic and have been created using a C program. This approach makes it easy to generate the target images without user intervention and time-consuming editing, which would be very tedious for images containing a large number of features as our training images do.

For example, Figure 5.6 shows the image containing the squares from Figure 5.3 and Figure 5.7 shows the image containing the circles from Figure 5.2.

Each training set includes only one feature size (e.g. Figure 5.3 shows an image containing four features of size 3). We did not combine different sizes in the training sets.

Test Set

Test images are 640×480 including the same features with the same size but distributed randomly in a different way. The test images are not included in the training set.

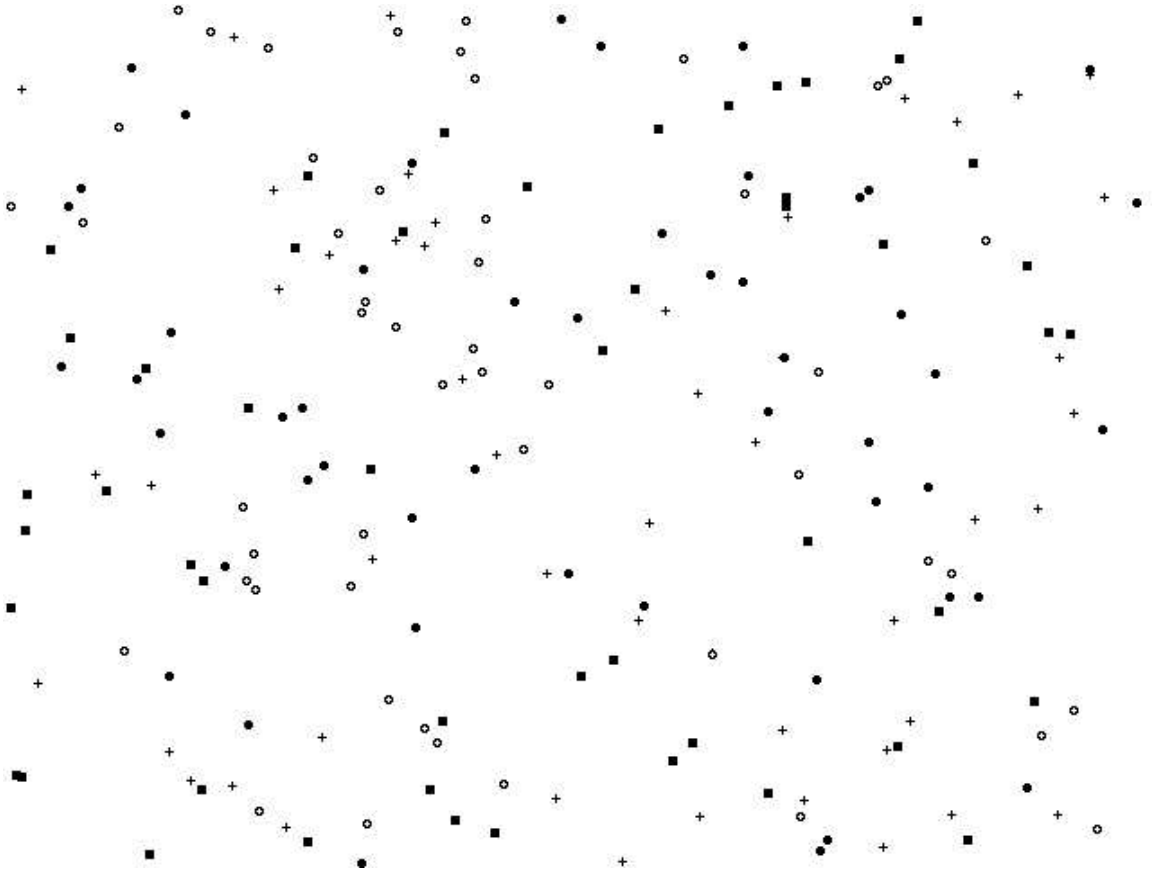


Figure 5.1: Original Image (size 1) containing four features: squares, circles, rings and stars.

5.2.2 The Image Processing Task

The task was to generate programs that accurately detect one of the features. This task is interesting because it is not easy even for human experts. Different tasks may require different sequences of different operators. If we demonstrate that GP search suitably varies for different training sets, this may suggest that the fitness function pushes the search in the right direction.

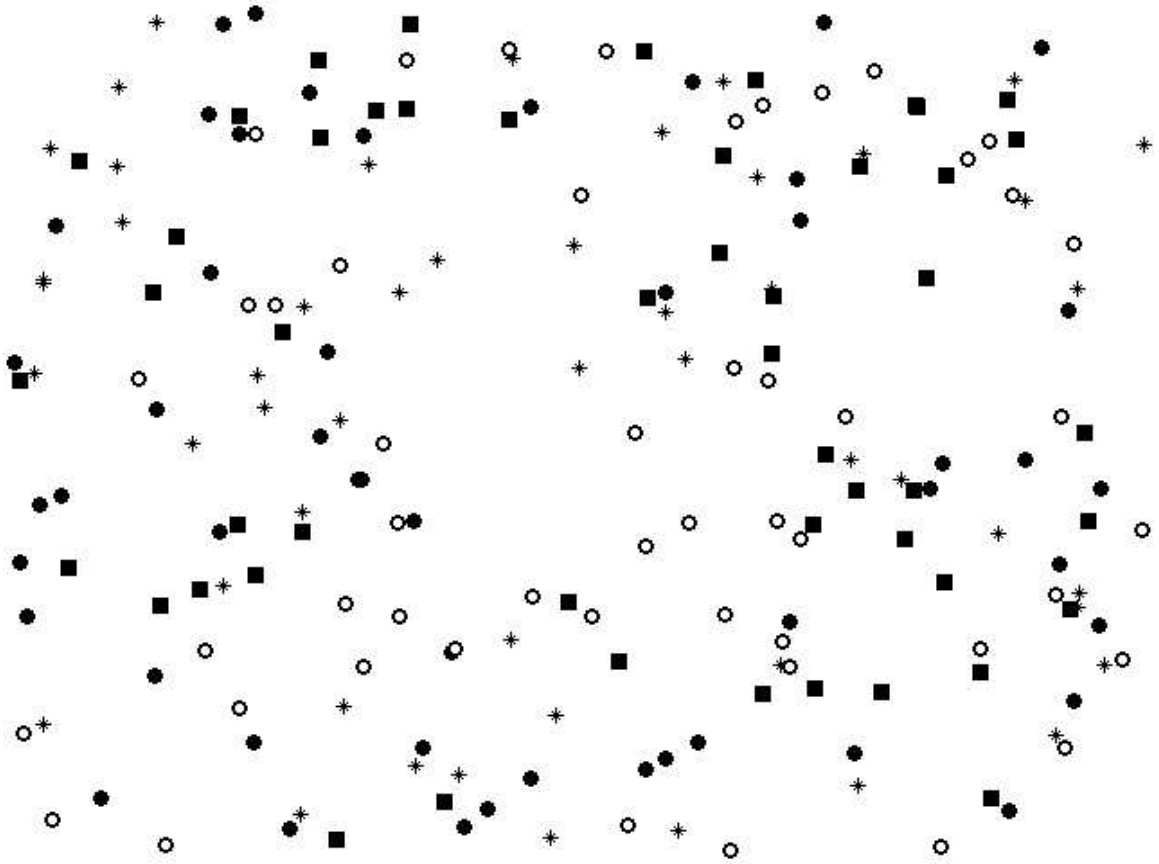


Figure 5.2: Original Image (size 2) containing four features: squares, circles, rings and stars.

5.2.3 Function and Terminal set

The Function and Terminal set is composed of logical and morphological operators as shown in table 5.1. EVAL is used to generate the tree-like structure needed in *Lilgp*. The terminals are generated randomly at the beginning of every run.

We have included a variable β to balance the number of logical and morphological terminals. If $\beta = 0.5$ then the number of logical or morphological operators are selected with equal probability. A higher value of β biases the search toward logical operators. The reason for including this variable was that logical and morphological operators modify the image in a very different way. Whilst logic operators modify the whole content of

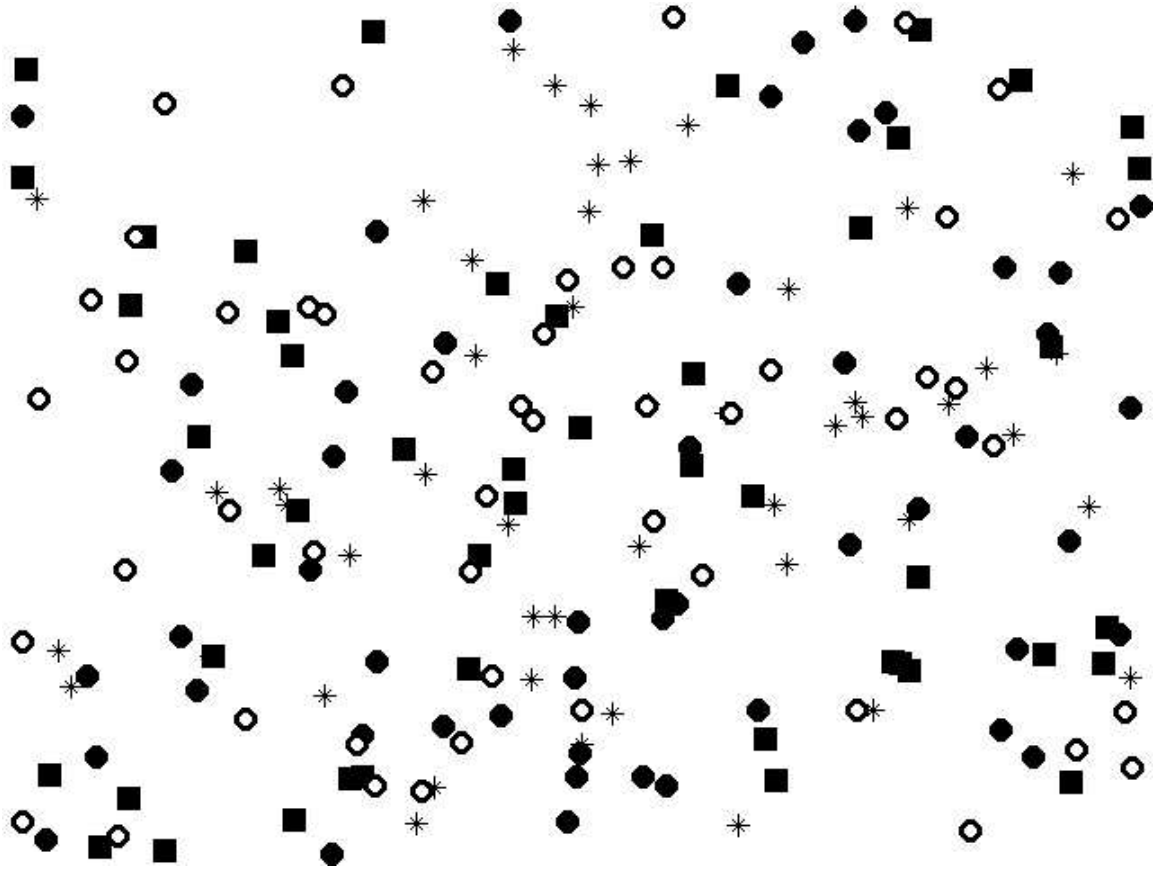


Figure 5.3: Original Image (size 3) containing four features: squares, circles, rings and stars.

the image, morphological operators modify the shapes inside the image. Logical operators modify the image as a whole because most of the pixels are modified with one operator while morphological operators usually modify the borders of the shapes only. Logical operators are computationally less expensive than morphological (i.e. logical operators require two nested cycles while morphological operators require a two nested cycle plus a three–nested cycle).

The SEs k are generated randomly at the beginning of each run. Chapter §3 showed the importance of using irregular SEs, therefore the kernels are drawn from the entire search space of 3×3 SEs. There are 512 options of 3×3 SEs (i.e. $2^9 = 512$). We represent

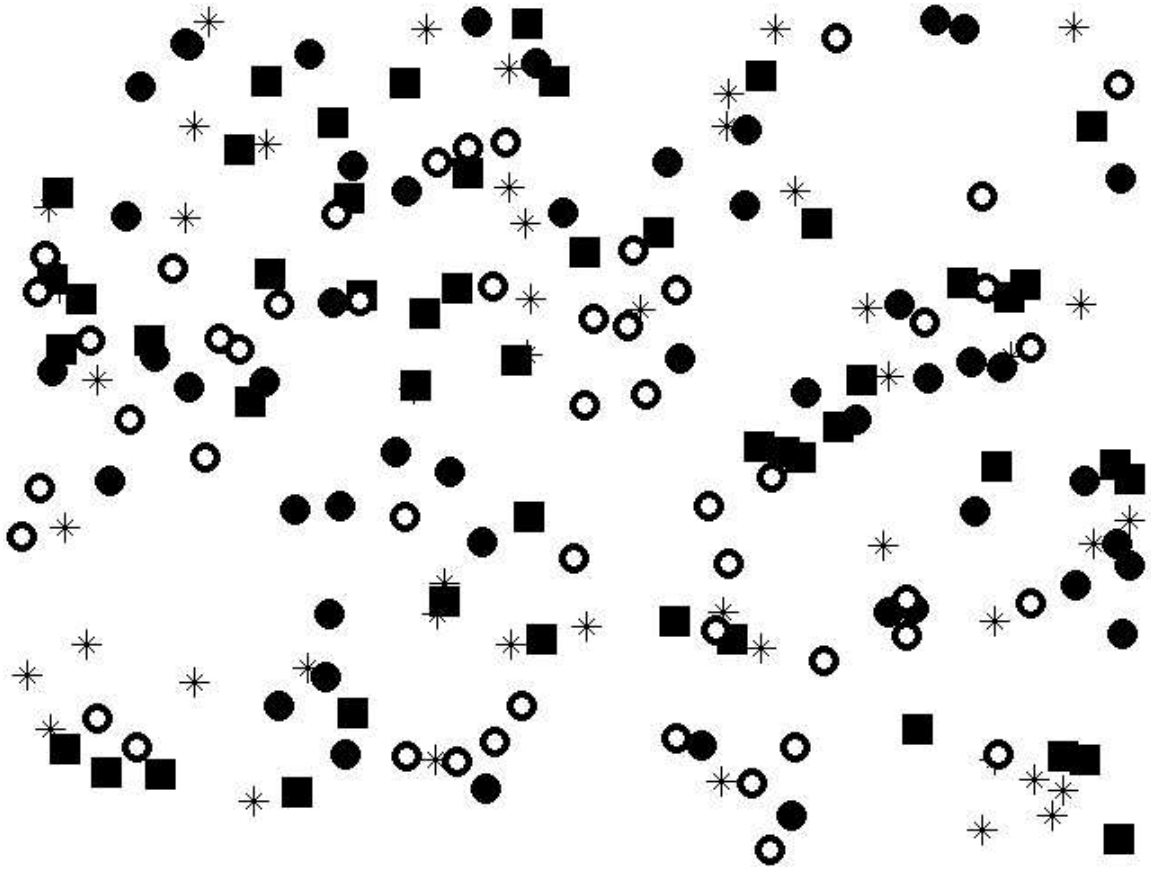


Figure 5.4: Original Image (size 4) containing four features: squares, circles, rings and stars.

every k value as the corresponding binary value read left-right and top-bottom (e.g. $e(8)$ corresponds to erosion using SE $(000, 001, 000)$).

5.2.4 Evaluation

For every evaluation the tree is converted into a sequence of operators. The sequences of operators are read from left to right. Every sequence starts with a default (not included) *store* operator which saves the original image into a buffer. When an operator &, |, $\sim$ or $\wedge$ appears in the sequence the respective logical operator is applied to the image (i.e. AND (&), OR (|), NOT ($\sim$), XOR ($\wedge$)). The logical operators AND, OR and XOR need two

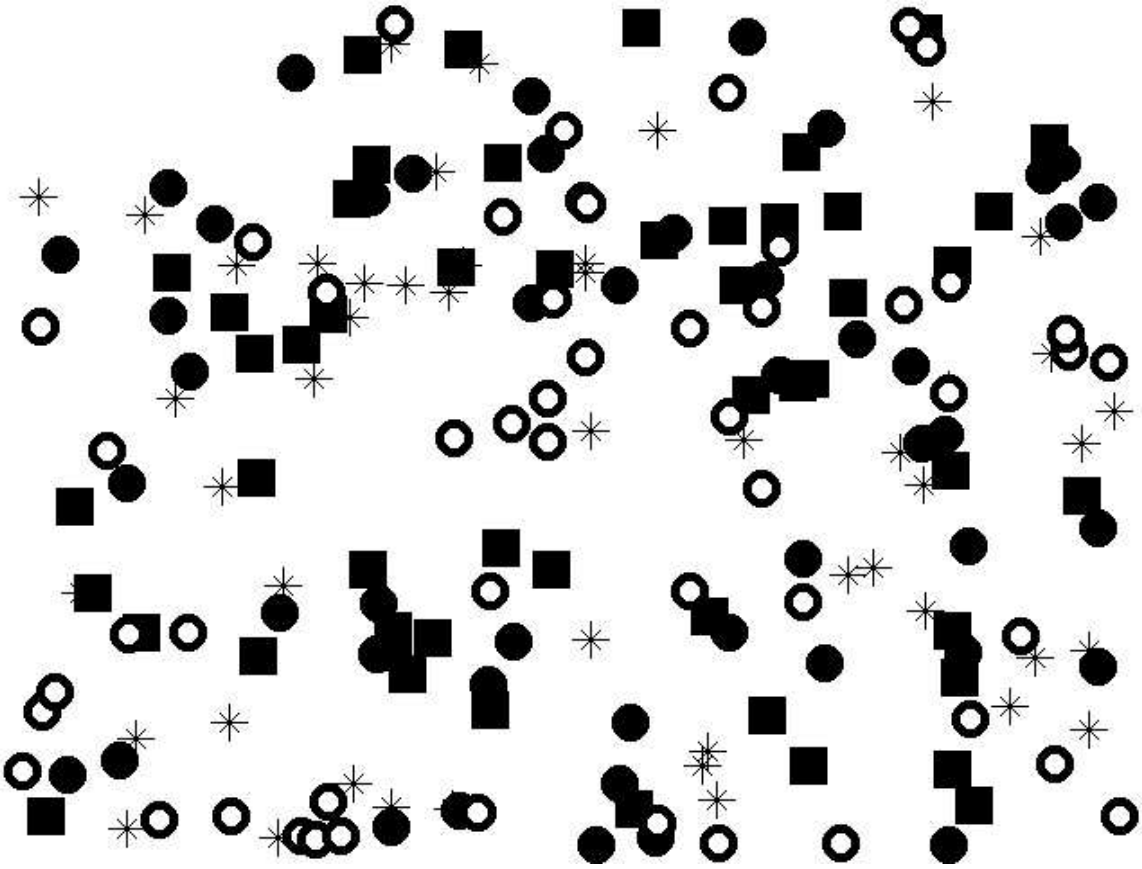


Figure 5.5: Original Image (size 5) containing four features: squares, circles, rings and stars.

input images, therefore the stored images is used together with the current image. When a *store* (s) operator appears in the sequence, the image obtained with the sequence so far substitutes the image currently saved in the buffer.

For example, the program $\sim d(238)\&d(462)se(185)\wedge$ read from left to right represents the storage of the image in the buffer (default initial operation), followed by a logical NOT, followed by a dilation using the SE corresponding to 238, followed by a logical AND, followed by a dilation using the SE corresponding to 426, followed by the storage of the current image in the buffer, followed by an erosion using the SE corresponding to 185 and finally a logical XOR of the current image with the buffer.

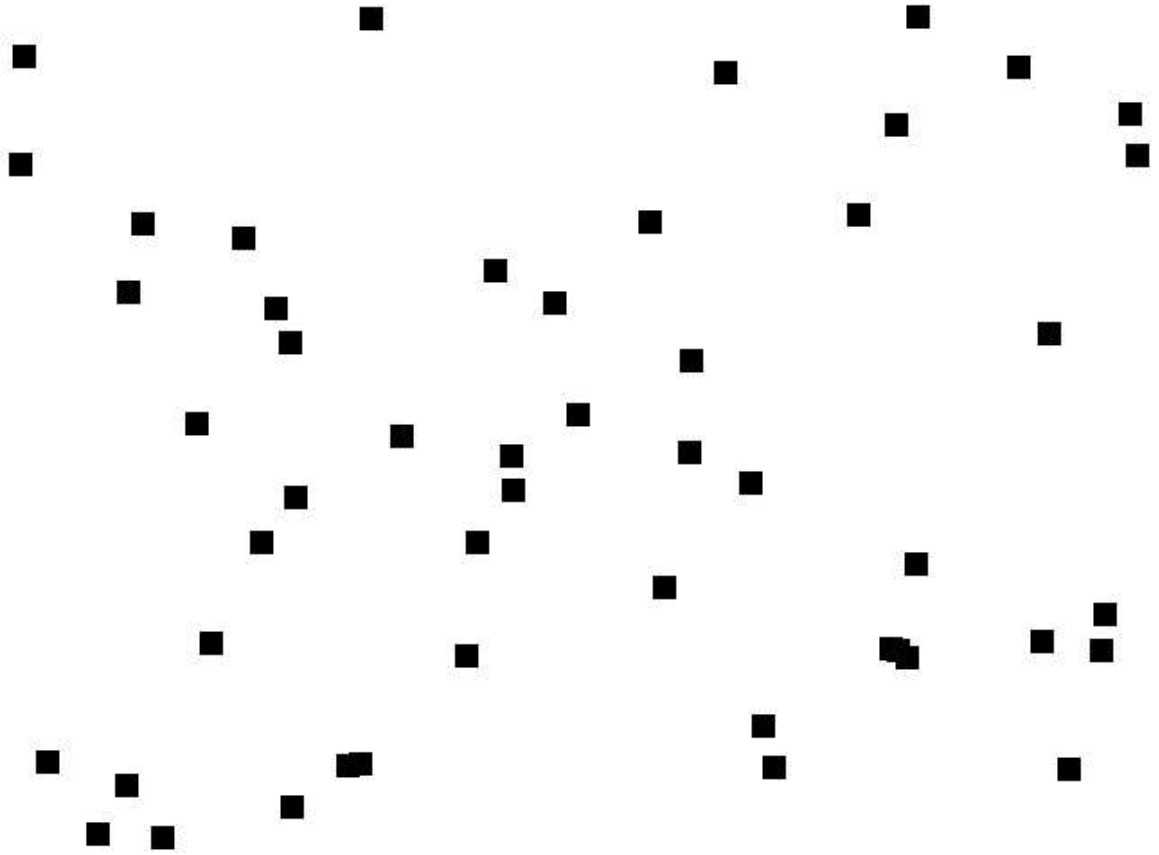


Figure 5.6: Expected squares from Figure 5.3

Functions	
Symbol	Action
&	$\cup$, Union, AND
|	$\cap$, Intersection, OR
$\wedge$	Exclusive-or, XOR
$\sim$	Negation, NOT
s	Store operator
$e(k)$	Erosion using kernel k (random [0..511])
d(k)	Dilation using kernel k (random [0..511])

Terminal		
Function	Arity	Description
EVAL	2	Tree generator

Table 5.1: Functions and terminal used in the hybrid approach.

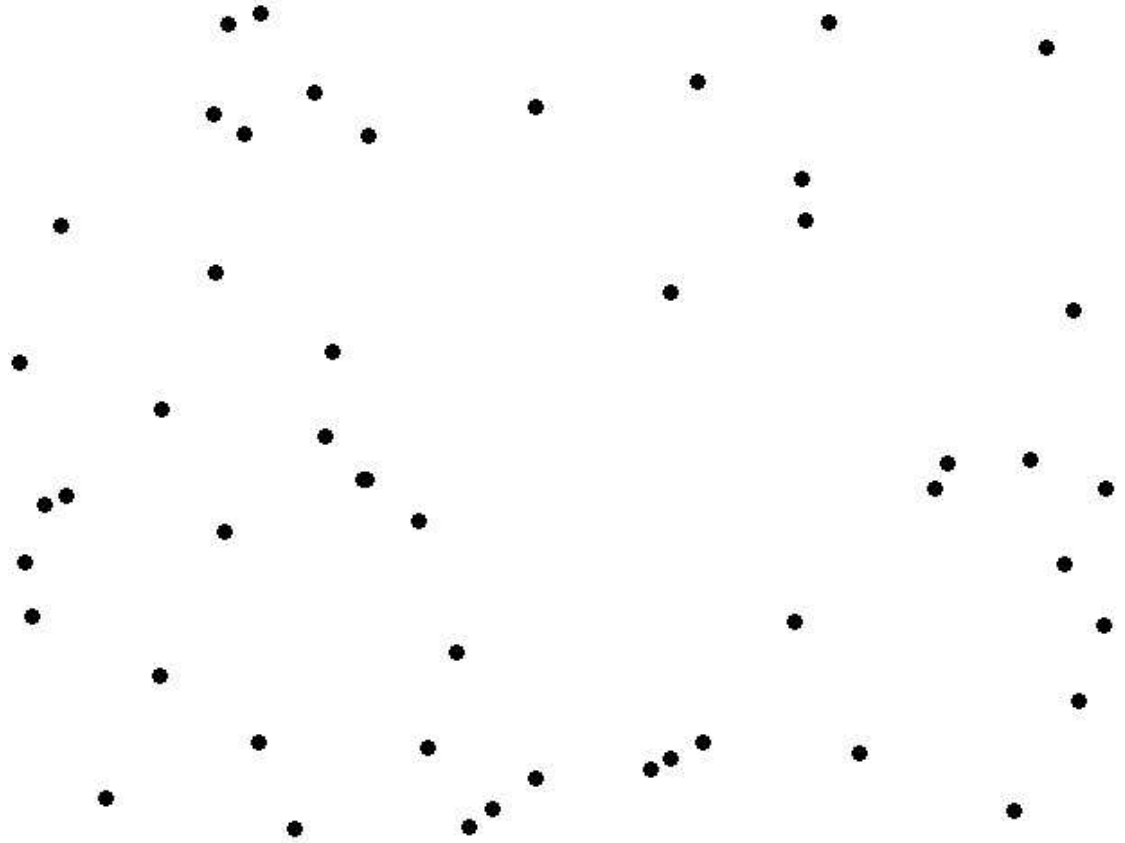


Figure 5.7: Expected circles from Figure 5.2

5.2.5 Fitness Function

The fitness function used is a modification of the Resemblance function f_B (Eq. 3.3) which includes a variable α (0..1) that allows tuning to bias 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:

$$f_C = 1 - \frac{\sqrt{(1-(SP \times (1-\alpha)))^2 + (1-(SV \times \alpha))^2}}{\sqrt{2}}. \tag{5.1}$$

5.2.6 The experimental set up

A set of 180 experiments was prepared. These includes 5 different resolutions, 4 different features and 9 different random seeds. The run values fixed were $\alpha = 0.9$, $\beta = 0.7$, crossover rate 0.9, mutation rate 0.2. Mutation was based on the ramped-half-and-half initialisation method, which was also used to initialise the population. All the experiments were implemented using Lilgp [160].

5.2.7 Environment

For the experiments in this Chapter we used a Linux cluster with Red Hat Linux 8.0 using one master node (CPU dual Intel Xeon 2GHz, 2GB Memory) and 22 client nodes (CPU Dual Athlon MP 1900+, 1.6GHz, 1GB Memory). In order to measure the system's performance we ran a set of preliminary experiments with different numbers of fitness evaluations (i.e. maximum number of generations of 500, 1000, 1500 and 2000 and populations sizes 500, 1000, 1500 and 2000). Our conclusion was that the cluster was hardly able to cope even with the minimum configuration (i.e. 500×500 fitness function evaluations). In the light of this we have decided to use the minimum configuration, so as to ensure that the experiments terminate.

5.3 Results

Figures 5.8, 5.9 and 5.10 show the output of a GP algorithm evolved for the detection of the ring, square and star features in Figure 5.3. Likewise, Figures 5.11 and 5.12 show results of the detection for features in Figure 5.3. These results are typical for most of the algorithms evolved.

Tables 5.2– 5.5 present the best numerical fitness values for each run. It is generally observed that for all the features at least one program achieved a fitness value over 0.94. This may lead to the conclusion that the evolution is pushing GP to solve the adequate

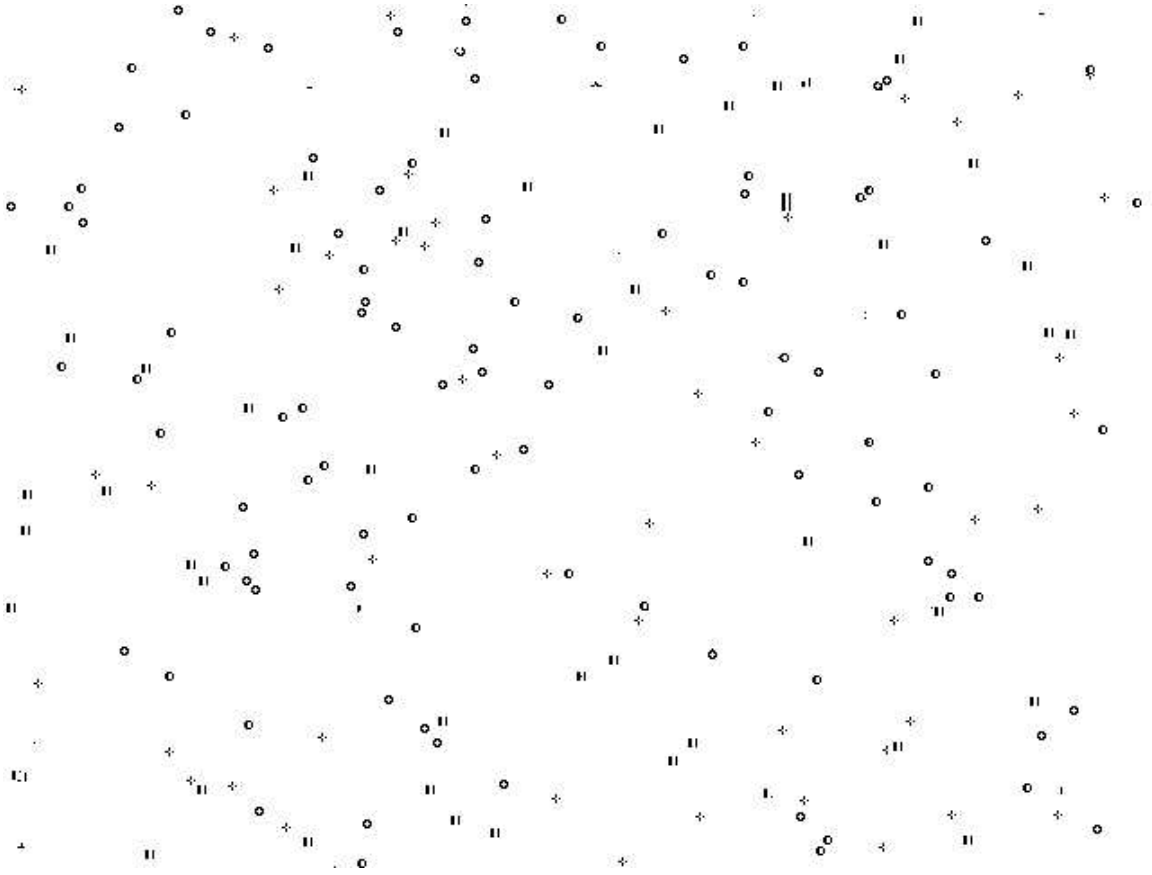


Figure 5.8: Rings detector. This is the final output provided by the ring-detecting program $\sim d(238)\&d(462)e(185) \wedge se(508)e(246)\wedge$ evolved by GP when applied to the image in Figure 5.1

feature detection regardless of the shape. There seems to be no significant difference between feature sizes since most fitness functions reach values over 0.9. Another observation was that not all the runs terminate properly, hence the result is not available (NA). It is likely that the runs did not terminate due to server overload. Each run took between 4 and 7 hours.

Figure 5.9: Squares detector. This is the final output provided by the square-detecting program $d(74)\wedge \mid e(251)$ evolved by GP when applied to the image in Figure 5.1

5.4 Discussion

The visual quality of the results returned in this set of experiments is considerably worse than of those obtained in Chapter 3. This is in spite of using a wider set of operators (i.e. both MM and logical) and of using a larger set of features for training. These results strongly support the notion that the user's feedback forms a very important part of the fitness evaluation. This form of feedback is normally implicit and it is rarely mentioned as a factor in Evolutionary Computation applied to images.

However, the analysis of the evolved algorithms (as opposed to the resulting images)

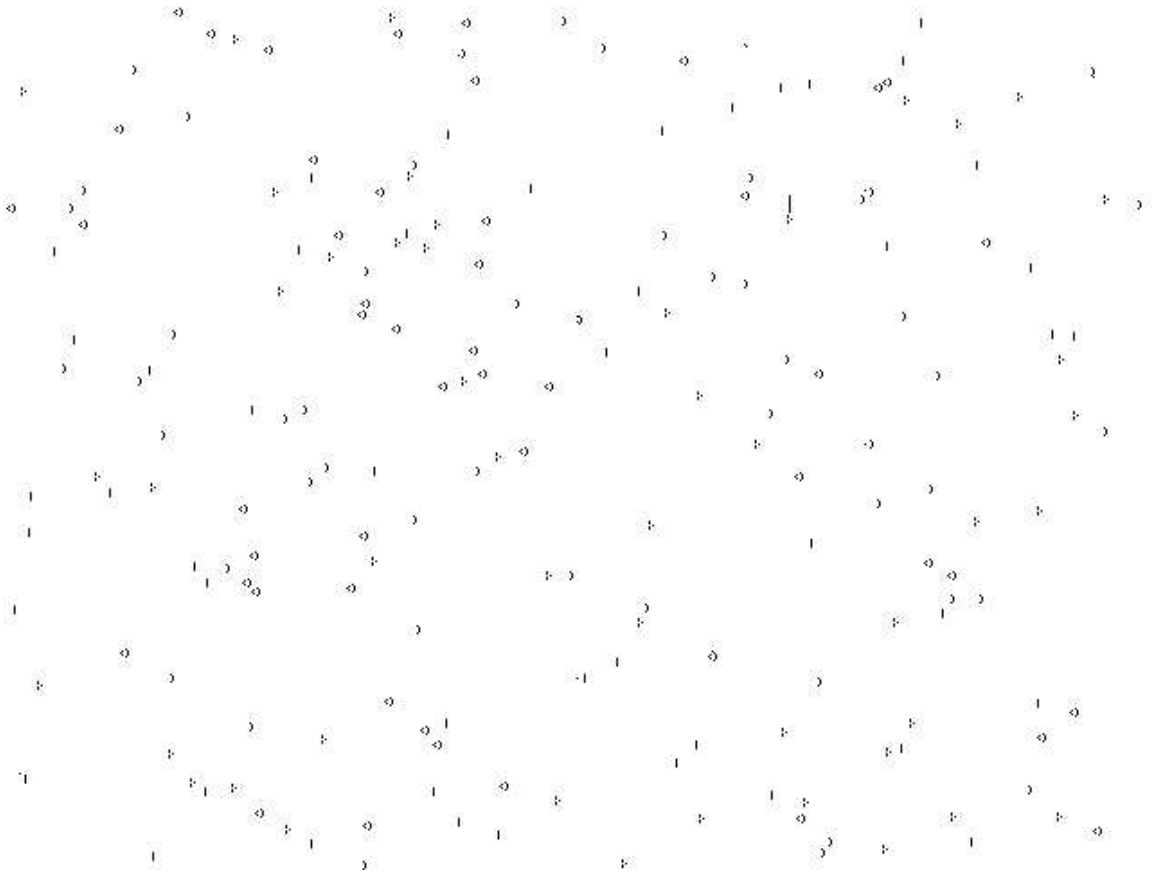


Figure 5.10: Stars detector. This is the final output provided by the star-detecting program $\sim e(138)\&$ evolved by GP when applied to the image in Figure 5.1

is more encouraging. The feature extraction problem posed in this set of experiments was difficult. It would therefore be very unlikely to evolve even partially effective algorithms randomly. On the inspection, the evolved algorithms show interesting traits and sequences which might have been evolved and recognised as logical by a human programmer.

As an example let us consider the evolved algorithm $d(74)\wedge \mid e(251)$ applied to the image in Figure 5.1. The first step $d(74)$ is a dilation using an irregular SE. The second step is an XOR using the original image; this removes most of the stars and rings. The third step is an OR with the original image; this enlarges the squares and circles. Finally the fourth step is an erosion $e(251)$ which eliminates part of the circles and the squares.

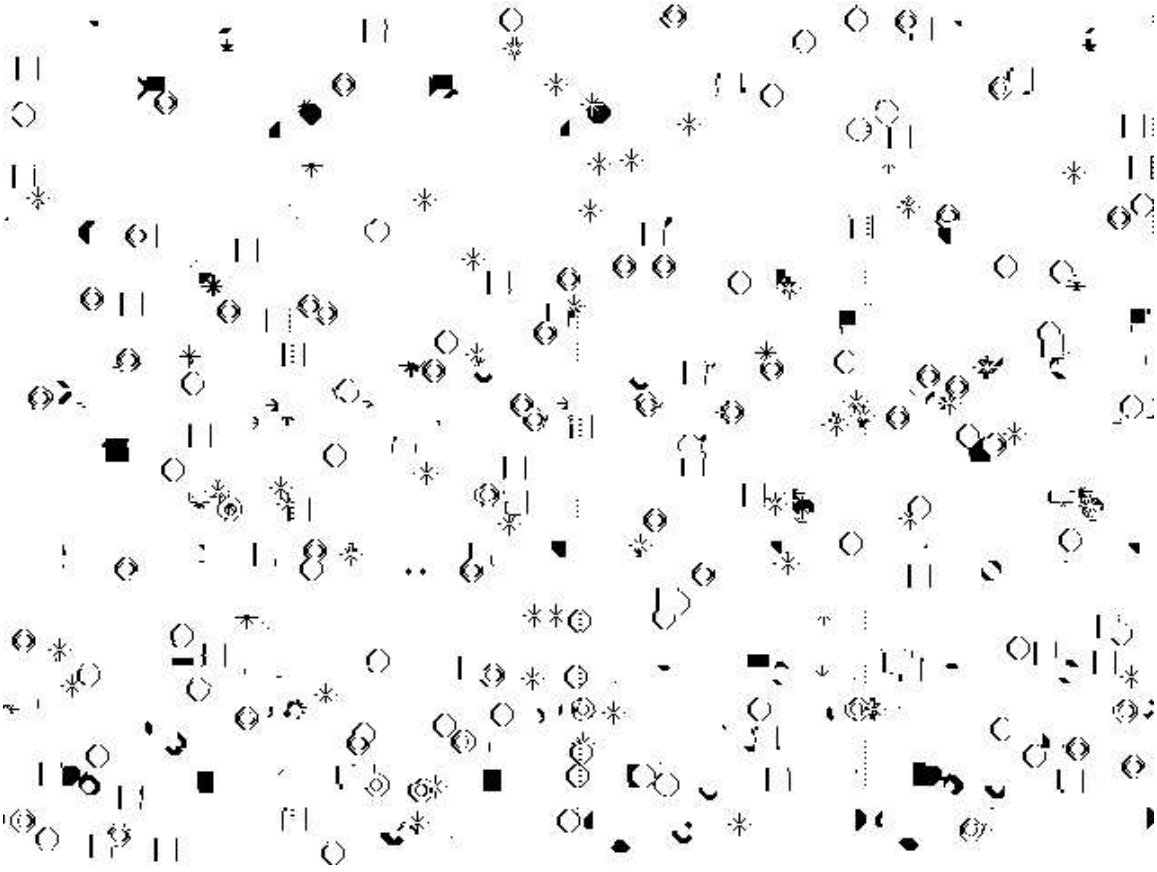


Figure 5.11: Rings detector. This is the final output provided by the ring-detecting program $\sim d(238)\&d(462)e(185) \wedge se(508)e(246)\wedge$ evolved by GP when applied to the image in Figure 5.3

The parameters α and β were put in place partially to replace the explicit user feedback. Whereas the experiments were inconclusive regarding β (the balance between the MM and the logical operators), setting α to a high value (i.e. $\alpha = 9$), to bias the fitness function towards very high specificity, consistently led to better visual quality of the resulting images than when α was low. This is consistent with the observations made in Chapter 3.

The experiments have highlighted the inadequacy of a fitness function which is based on individual pixels and does not take into account human perception. The need for a perceptually based fitness function is discussed in the concluding Chapter 6.

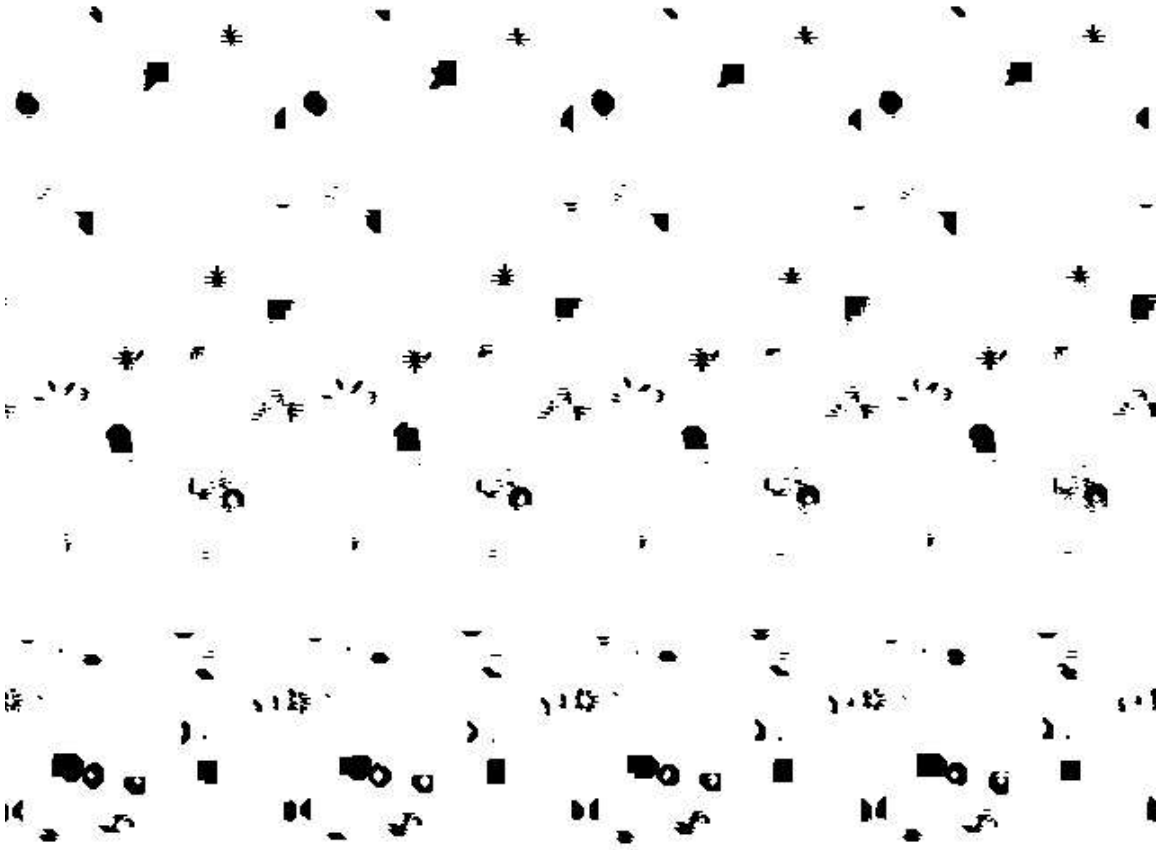


Figure 5.12: Squares detector. This is the final output provided by the square-detecting program $\wedge e(267) \wedge e(142)e(267) \wedge d(193)e(265)d(9)d(182)$ evolved by GP when applied to the image in Figure 5.3

5.5 Chapter Summary

This chapter described a preliminary set of experiments to explore the performance of the GP algorithms evolved totally automatically, without explicit or implicit user feedback. The experiments have confirmed the importance of the role that the user plays in the evaluation of the pictorial outputs. Whilst visually the results could be much improved, the algorithms contain effective, although incomplete, operator sequences.

Table 5.2: Best numerical fitness values for Square Detectors.

	Best Fitness for Square Detectors				
	Feature Size				
	1	2	3	4	5
Runs					
1	0.798818	0.783523	0.780784	0.775885	0.769051
2	0.798818	0.783523	0.780784	0.775885	0.769051
3	0.947817	0.943099	0.933182	0.929711	NA
4	0.948798	0.943099	0.931338	0.929745	NA
5	NA	0.944189	0.931095	0.929812	NA
6	NA	0.783523	0.780784	0.748584	NA
7	NA	0.761179	0.780784	0.775885	NA
8	NA	0.942951	0.926524	0.853889	NA
9	NA	0.942839	0.926689	0.865183	NA
NA means Not Available					

Table 5.3: Best numerical fitness values for Circle Detectors.

	Best Fitness for Circle Detectors				
	Feature Size				
	1	2	3	4	5
Runs					
1	0.820240	0.821542	0.813145	0.811260	0.805648
2	0.820240	0.821542	0.813145	0.811260	0.805648
3	0.947770	0.946664	0.929652	0.934687	0.925630
4	NA	0.946672	0.933240	0.934680	0.925630
5	NA	0.946665	0.933270	0.934654	0.925590
6	NA	0.821542	0.641787	0.811260	0.644148
7	NA	0.746412	0.716023	0.811260	0.773692
8	NA	0.936418	0.840953	0.917778	0.809275
9	NA	0.840353	0.641786	0.871008	0.644148
NA means Not Available					

Table 5.4: Best numerical fitness values for Ring Detectors.

	Best Fitness for Ring Detectors				
	Feature Size				
	1	2	3	4	5
Runs					
1	0.849446	0.862156	0.850385	0.845347	0.840962
2	0.849446	0.862156	0.850385	0.845347	0.840962
3	NA	0.948235	0.941237	0.938996	0.863007
4	NA	0.948210	0.944260	0.939015	0.931851
5	NA	0.947374	0.939235	0.939034	0.748219
6	NA	0.862156	0.850385	0.845347	0.643103
7	NA	0.862156	0.850385	0.845347	0.643103
8	NA	0.919204	0.933268	0.934803	0.643103
9	NA	0.942405	0.937697	NA	0.643102
NA means Not Available					

Table 5.5: Best numerical fitness values for Star Detectors.

	Best Fitness for Star Detectors				
	Feature Size				
	1	2	3	4	5
Runs					
1	0.892949	0.895277	0.910033	0.917760	0.923306
2	0.892949	0.895277	0.910033	0.917760	0.923306
3	0.950536	0.947066	0.918326	0.946531	0.932250
4	0.949900	0.947810	0.942752	0.946499	0.945119
5	0.949905	0.947059	0.943487	0.946515	0.945151
6	0.892949	0.895277	0.746185	0.917760	0.840356
7	0.892949	0.895277	0.640735	0.917760	0.746416
8	0.932004	0.933459	0.806562	0.935049	0.746416
9	0.946457	0.933685	0.900301	0.935173	0.840356

Chapter 6

Conclusions

Chapter Outline

This chapter presents the conclusions of this thesis. Section §6.1 briefly summarises the topics that this thesis has explored. Section §6.2 describes the contributions of this thesis. Section §6.3 explains to what extent the open questions stated in the motivation have been answered and discusses the limitations of this thesis. Finally section §6.4 discusses ideas for further research.

6.1 Summary

GP applied to Computer Vision is in its early stages but it is very promising and it is believed that it will lead to the discovery of powerful new programs. In the literature, GP has shown some success in image analysis applications but it has been mainly used

for linear image processing and searching for filters, leaving the non-linear side practically unexplored. MM is a versatile non-linear technique that allows the combination of exploratory features in several ways.

This thesis provides an investigation on GP applied to morphological image processing. There have been other attempts to achieve automatic programming of MM algorithms in the literature, but just a few attempts to evolve MM algorithms using GP and other EC techniques. A popular technique in MM is to construct sequences of morphological operators based on concatenations of the two basic operations: dilation and erosion. The sequences of morphological operators are usually constructed according to the intuition and experience of human programmers and practitioners. From the pragmatic point of view, a sequence of morphological operators extracts information from images, but it is usually difficult to design adequate sequences of morphological operators to solve particular problems for image analysis. This usually requires heuristics and also requires the decomposition of the the general problem into sub-problems in an *ad hoc* way [124, 35, 14].

This thesis has explored the approach of constructing a GP system that automatically designs sequences of morphological operators. Three aspects of morphological image processing have been investigated. The extensive set of experiments aimed at evolving MM programs using GP produced satisfactory results, but also identified inefficiencies when using big populations over a large number of generations. This led to the exploration of SMCGP as the means of speeding up the evolution. The use of SMCGP made the evolution of big populations over a large number of generations feasible but did not produce good visual results. Finally, the combined use of the logical and MM operators was explored and the role of the user feedback, as an implicit fitness function was examined.

6.2 Contributions

This thesis has made a number of contributions to the fields of Mathematical Morphology and Genetic Programming.

The literature review of the state of the art of automatic morphological image processing with emphasis on EC applied to morphological image processing demonstrated that there was a gap in GP applied to morphological image processing where our main contributions have been made.

* We explored the feasibility of GP as a technique for the automatic generation of MM programs. We have indeed demonstrated that it is possible to evolve effective morphological programs using GP. Moreover, the obtained programs are fairly easy to understand and easy to analyse step by step. This is not necessarily the case for other evolutionary techniques.

* It is interesting to note that one important limitation for human practitioners using MM is the understanding of the results obtained when using irregular SEs. A contribution of this thesis is the GP exploration of irregular SEs. Through the experiments we have demonstrated that they are useful for the automatic creation of morphological algorithms for image processing.

* We have also explored the question of the design of SEs which is important for MM. We have concluded that using bigger SEs tends to produce visually better output images, but big SEs also lead to slower performance.

* Previous approaches using EC techniques were limited to sequences of morphological operators with fixed sizes. One of our contributions is to allow the sequences to have operators of variable size. The use of variable sizes of the sequences has proved to be useful.

* We explored the inclusion of various image sizes in the training set and we did not find an advantage of using a particular size. Nonetheless, an observation was that it is better to use only one big image containing many features in the training set.

This seems to lead to better GP evolved programs, probably due to the availability of richer contextual information.

* We performed a preliminary research on the SMCGP technique to speed up the implementation. Although SMCGP succeeded in this respect, it did not succeed in providing a good quality of visual results.

* In the course of this work, we studied three different fitness functions and various sets of functions and terminals. Our conclusion is that all these features have to be adapted to the problem in hand in order to make the requirements to the GP system as precise as possible.

In conclusion, these contributions are significant because they have demonstrated the feasibility of the automatic programming of effective morphological algorithms (i.e. program implementations) using GP. Other researchers may build on these contributions, using them as the basis for exploring the automatic programming of morphological image processing in higher dimensions, for example grey-level or $3D$ shape. Work described in Chapters §3, §4, §5 of this thesis has been published [110, 111], cited [11, 115], or accepted for publication [109] in peer-review outlets, thus confirming that it is valued by the research community.

These contributions are substantial because they present a study of a well understood problem using a bottom-up approach from the very basic level. We included a description of the problem, hypotheses, experiments using several approaches, results, analysis, statistics, conclusions, limitations and have identified lines for further research.

6.3 Discussion

The evidence provided in this thesis allow us to confirm that using a variable number of morphological operators benefits the automatic programming of morphological image

processing using GP. However, it is important to point out that the experiments presente may not be statistically significant to generalise the presented results.

It is also confirmed that the use of irregular SEs improves the programs evolved. This issue has been analysed empirically. The analysis can be improved further by carrying on statistical analysis of the SEs that appear more often in the evolution. Another issue is that depending on the sort of features we are looking for, it may be useful to seed some SEs related to the shapes. We have not explored this idea on this thesis.

One of our hypotheses was that there is no reason to restrict the search to a particular size of SEs. We were proved wrong, because for implementation reasons it is better to use small SEs. Nonetheless, larger SEs have been useful as well. We suggest that it may be useful to look for an alternative implementation (e.g. using SMCGP) to exploit the benefits of using larger SEs.

The preliminary exploration of combining logical and morphological operators suggests that it may be beneficial to the performance and to the program quality. However, the study has to be much more substantial to find out to what extent the combination is useful (e.g. including variations of β in experiments similar to those in Chapter §5).

We found that our knowledge about the design of fitness functions that accurately capture the behaviour of the human vision is very poor. There must be better ways to model the expected results from evolutionary algorithms than to simply count pixels. So far f_C appears to be an interesting fitness function to continue exploring, tuning the values of α for different sub-problems of the general signal understanding problem (see below).

As regards generalisation, we may say that GP has proved to be a flexible technique that is pushing the evolution according to the problems that we intend to solve and obviously improves the results of those of a random search. Whether GP is the *best* technique for morphological image processing is an open question from our point of view.

We have come up with some ideas to make the GP implementation feasible and effective. Exploratory work described in Chapter §5 has suggested the idea that an attractive

alternative may be to sample only those algorithms which modify a significant number of the pixels in the image. This idea may save lot of computational effort and may lead in turn to improved performance.

From the personal perspective we have found that the task of evolving morphological algorithms is far more difficult than initially believed.

6.4 Future Research

Frequently, the systems that automatically design sequences of morphological operators are based on Artificial Intelligence or closely related disciplines. In their recent book *Foundations of Genetic Programming* [83], Langdon and Poli put forward their view about Artificial Intelligence (AI) foundations as a whole:

> The foundations of AI are fundamental principles which are common to all disciplines within AI, be they artificial neural networks, evolutionary computation, theorem proving, etc. The common feature of this techniques is search (although the representation being used to express solutions and the search used may be radically different). In our opinion search (be it deterministic or stochastic, complete or incomplete, blind, partially sighted, heuristic, etc.), the related representation, operators and objective functions are the foundations of AI.

In this context, it is interesting to note that MM has not been systematically explored from the AI point of view. This thesis explored the extraction of features in binary images, but it is indeed possible to use GP and other AI techniques to solve the general problem of signal understanding.

6.4.1 Signal Understanding problem.

For the problem of Signal Understanding, Teller [142] offers a succinct definition:

> Given a 'signal' type and a notion of 'symbols' that represent important characteristics of signals of that type, how can we create a process for automatically extracting these symbols from the signals they represent?

In the ideal case we would like to create a system able to automatically extract and detect symbols from a variety of sources. The best approach known so far is the human being, an entity capable to detect, analyse and understand a wide range of signals and to map them adequately to symbols (i.e. humans are capable to map sounds, images and sensations to their corresponding labels or symbols).

Recent results in theory and applications lead to the conclusion that is not possible to obtain a general solution to complex problems (such as Signal Understanding) but it is possible to divide the general problem into sub-problems and then find specific solutions for those sub-problems [89, 83, 149]. A possible approach to solve the problem of Signal Understanding is to divide the problem into sub-problems:

* Classification. The signal s belongs to one of several classes.

* Detection. Is the symbol s present in the signal f?

* Location. Where is the symbol s located into the signal f? The symbol s may be present 0 or more times in the signal f.

* Extraction. Extract symbol s from signal f. This may be done either converting signal f into another signal containing symbol s only or indicating the characteristics of symbol s presence into signal f.

* Recognition. Which symbols are present in signal f?

These sub-problems have been widely explored from the linear processing point of view. Nonetheless, to the best of our knowledge, an approach to solve the general problem of Signal Understanding has not been proposed yet (although Teller and Veloso invested a

big effort to describe the sub-problems involved and they achieved interesting results to solve the sub-problem of classification [145, 143, 144, 142]).

It is arguable that the most promising approach to solve the general problem of Signal Understanding is an hybrid Human-ML approach, where we take advantage of human capabilities to understand and describe abstract signals and ML capabilities to search and learn from complex search spaces (such as the space of signals).

It is also arguable that a promising approach to solve the general problem is starting from binary signals. The problem of Signal Understanding on the binary level is still a complex one and we have to solve it first in order to solve the sub-problems in more complex domains.

Morphological processing in particular and non-linear processing in general are important because the human understanding of signals as symbols has a tendency to use non-linear features. Perhaps the biggest advantage of MM as a technique for Signal Understanding is its ability to describe in mathematical terms what humans understand as *shape*.

First of all, if we receive a signal and we don't know anything about it, it is difficult to know what sort of signal we are dealing with. A statistical measure or a transform into the frequency domain can provide some information about its contents, but it is only when human expertise converts it into a *symbol* that we can understand what sort of signal we are dealing with.

Objective Functions

A practitioner should take advantage of his own expertise and provide an automatic system with the best measure of success according to the sub-problem at hand. As mentioned above, we are still far from solving the general problem of Signal Understanding in the binary level, so it is suggested to start exploring objective functions for binary signals/images. Objective functions may include Hamming distance, similarity [151], sensitivity-specificity

trade-off measures [101, 111], functions for OCR [60], subjective visual evaluation [106] or any other the practitioner's intuition suggests may solve the sub-problem at hand.

It is interesting to note that the state-of-the-art research in ML contemplates the combination of success measures using approaches such as multi-objective optimisation [33, 29].

There are several lines open for further research. Theoretically, it is possible to extend the results of this thesis to evolve morphological algorithms to higher dimensionality (grey-level images, time-scales, multi-layer images). In practice, this requires far more computation resources.

An interesting line of research proposed by Bloomberg [25] is to investigate whether it is possible to apply MM operators on the bitwise level. If that it is possible, then it is very likely that we may explore the advantages of MM and SMCGP to evolve algorithms suitable for a broad kind of binary image processing tasks. An interesting line for further research could be to write morphological functions as combination of bitwise operations to speed up the evolution while conserving the accuracy already obtained with GP and MM.

Another alternative that has been little explored in the image processing field is the use of Multi-objective evolutionary optimisation [29, 33, 81]. Multi-objective optimisation has been developing very fast in the last few years and a better analysis in the context of multi-objective optimisation may lead to better results. Another possibility would be to define binary detection in the context of classifier systems understanding *strength* and *accuracy* as analogues for sensitivity and specificity. If binary image processing is re-defined as a classification problem, a general dilemma about strength and accuracy in automatic classifiers will apply to binary images [70].

It has become clear that the choice of the fitness function is very important and that this topic needs much further work. Whereas f_B was fairly good at achieving a good balance between sensitivity and specificity, it might be more advantageous to have other means of directing the search toward a desired solution. For example, for the detection of small features a highly sensitive algorithm would be better while, if the image contains a

number of features that might be confused one with another, a highly specific algorithm would be more beneficial.

In general, the main limitation of all the existing fitness functions is that they do not take into account the characteristics of the shape itself. It is hard to imagine that the mean-absolute error is the only measure to obtain an optimal filter, particularly when the intention is to find a shape with no apparent statistical correlation (e.g. signature authentication).

Appendix A

Notation

Acronyms

AI	Artificial Intelligence.
CPU	Central Processing Unit.
CPUs	Central Processing Units.
EC	Evolutionary Computation.
FN	False Negative.
FP	False Positive.
GA	Genetic Algorithm.
GAs	Genetic Algorithms.
GP	Genetic Programming.
ML	Machine Learning.
MM	Mathematical Morphology.
NA	Not Available.
OCR	Optical Character Recognition.
SE	Structuring Element.
SEs	Structuring Elements also known as morphological filters or *kernels* (pag. 21).

SIMD	Single Instruction Multiple Data.
SP	Specificity.
SV	Sensitivity.
TN	True Negative.
TP	True Positive.
SMCGP	Sub-Machine-Code Genetic Programming.

Mathematical Notation

$x = (x_1, x_2)$	Tuple representing the coordinates of a pixel in an image.
$x = (x_1, x_2, x_3)$	Triplet representing the coordinates of a pixel and its grey level.
$1D$	One dimensional.
$2D$	Two dimensional.
$3D$	Three dimensional.
$=$	Equal.
$\neq$	Not equal.
$+$	Addition.
$\cup$	Set union operator.
$\cap$	Set intersection operator.
$\subseteq$	Subset equal.
$\emptyset$	Empty set.
$\in$	Set membership operator: $x \in X$ means x is an element of X.
$\notin$	$x \notin X$ means x is not an element of X.
$\sum$	Summation, e.g., the sum of a series of values X_1 to X_n is $\sum_{i=1}^{n} X_i$.
$(B)_x$	Translation of set B by point x. Pag. 20.

$\widehat{C}$	Reflection of set C. Pag. 20.
$\oplus$	Morphological dilation.
$\ominus$	Morphological erosion.
Z^2	Two dimensional integer space.
Z^3	Three dimensional integer space.

Miscellaneous Notation

§	Chapter, Section or Subsection in this thesis.
eq.(1.1)	A numbered equation.
e.g.	For example (Latin *exempli gratia* for the sake of example).
et al.	And others (Latin *et alii*).
etc.	And the rest, and so on (Latin *etcetera*).
i.e.	That is to say (Latin *id est*).
pag.	Page, pages.

Bibliography

[1] M. Acheroy, C. Beumier, J. Bigun, G. Chollet, B. Duc, S. Fischer, D. Genoud, P. Lockwood, G. Maître, S. Pigeon, I. Pitas, K. Sobottka, and L. Vandendorpe. Multi-modal person verification tools using speech and images. In *Proceedings of the European Conference on Multimedia Applications, Services and Techniques (ECMAST 96)*, pages 747–761, Louvain-la-Neuve, 28–30 1996. Available at citeseer.ist.psu.edu/acheroy96multimodal.html on 14/10/04.

[2] G. Adorni, F. Bergenti, and S. Cagnoni. A cellular-programming approach to pattern classification. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391 of *LNCS*, pages 142–150, Paris, 14-15 April 1998. Springer-Verlag.

[3] 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 September 2001. Published 2002.

[4] G. Adorni, S. Cagnoni, and M. Mordonini. Efficient low-level vision program design using sub-machine-code genetic programming. Workshop sulla Percezione e Visione nelle Macchine, available at citeseer.nj.nec.com/539182.html, 2002.

[5] D. Agnelli, A. Bollini, and L. Lombardi. Image classification: an evolutionary approach. *Pattern Recognition Letters*, 23(1-3):303–309, 2002.

[6] J. T. Alander. An indexed bibliography of genetic algorithms in signal and image processing. Report 94-1-SIGNAL, University of Vaasa, Department of Information Technology and Production Economics, 1995.

[7] D. Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In K. E. Kinnear, editor, *Advances in Genetic Programming*, Complex Adaptive Systems, pages 477–494, Cambridge, 1994. MIT Press.

[8] G. Anelli, A. Broggi, and G. Destri. Decomposition of arbitrarily shaped binary morphological structuring elements using genetic algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):217–224, 1998.

[9] J. Bala and H. Wechsler. Shape analysis using morphological processing and genetic algorithms. In *Proceedings of the 1991 IEEE International Conference on Tools with Artificial Intelligence TAI'91*, pages 130–137, Los Alamitos, CA., November 1991. IEEE Computer Society Press.

[10] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, Englewood Cliffs, 2nd edition, 1982.

[11] L. Ballerini and L. Franzèn. Genetic optimization of morphological filters with applications in breast cancer detection. In G. R. Raidl *et al.*, editor, *Applications of Evolutionary Computation, EvoWorkshops 2004*, number 3005 in LNCS, pages 250–259, Berlin, 2004. Springer.

[12] 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, January 1998.

[13] W. Banzhaf, P. Nordin, and M. Olmer. Generating adaptive behavior for a real robot using function regression within genetic programming. 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 35–43, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[14] J. Barrera, E.R. Dougherty, and M. Brun. Hybrid human-machine binary morphological operator design. an independent constraint approach. *Signal Processing*, 80(8):1469–1487, 2000.

[15] J. Barrera, E.R. Dougherty, and N.S. Tomita. Automatic programming of binary morphological machines by design of statistically optimal operators in the context of computational learning theory. *Journal of Electronic Imaging*, 6(1):54–67, 1997.

[16] J. Barrera, R. Terada, R. Hirata, and N.S.T Hirata. Automatic programming of morphological machines by pac learning. *Fundamenta Informaticae*, 41(1-2):229–258, 2000.

[17] D. Barrios, A. Carrascal, D. Manrique, and J. Rios. Optimisation with real-coded genetic algorithms based on mathematical morphology. *International Journal of Computer Mathematics*, 80(3):275–293, March 2003.

[18] D. Barrios, D. Manrique, J. Porras, and J. Ríos. Real-coded genetic algorithms based on mathematical morphology. *Lecture Notes in Computer Science*, 1876:706–712, 2001.

[19] T. Belpaeme. Evolution of visual feature detectors. In R. Poli, S. Cagnoni, H. M. Voigt, T. Fogarty, and P. Nordin, editors, *Late Breaking Papers at EvoISAP'99: the First European Workshop on Evolutionary Computation in Image Analysis and Signal Processing*, pages 1–10, Goteborg, Sweden, 28 May 1999.

[20] K. A. Benson. Evolving automatic target detection algorithms that logically combine decision spaces. In *Proceedings of the 11th British Machine Vision Conference*, pages 685–694, Bristol, UK, 11-14 September 2000.

[21] K. A. Benson. Evolving finite state machines with embedded genetic programming for automatic target detection within SAR imagery. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 1543–1549, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 July 2000. IEEE Press.

[22] K. A. Benson, D. Booth, J. Cubillo, and C. Reeves. Automatic detection of ships in spaceborne SAR imagery. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H. G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, page 767, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[23] K. A. Benson, D. Booth, J. Cubillo, and C. Reeves. On the use of evolution to construct finite state machines and mathematical functions to perform automatic target detection. In *Proceedings of the 3rd IMA conference on image processing: mathematical methods, algorithms and applications*, pages 14–18, Leicester, UK, 13-15 September 2000. IEEE.

[24] T. F. Bersano-Begey, J. M. Daida, J. F. Vesecky, and F. L. Ludwig. A Java collaborative interface for genetic programming applications: Image analysis and scientific inquiry. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 477–482, Indianapolis, 13-16 April 1997. IEEE Press.

[25] D. S. Bloomberg. Implementation efficiency of binary morphology. In H. Talbot and R. Beare, editors, *ISMM 2002 International Symposium of Mathematical Morphology VI*, pages 209–218, Sydney, Australia, April 2002. CSIRO Publishing.

[26] B. J. Bozarth. Programmatic compression of video using genetic programming. In J. R. Koza, editor, *Genetic Algorithms and Genetic Programming at Stanford 2000*, pages 46–53. Stanford Bookstore, Stanford, California, 94305-3079 USA, June 2000.

[27] P. Brigger and M. Kunt. Morphological contour coding using structuring functions optimized by genetic algorithms. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 534–537, Los Alamitos, CA., October 1995. IEEE.

[28] C. Clack, J. Farringdon, P. Lidwell, and T. Yu. An adaptive document classification agent. Research Note RN/96/45, University College London, Computer Science, Gower Street, London, WC1E 6BT, UK, 21 June 1996.

[29] C. A. Coello, D. A. V. Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[30] 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.

[31] J. M. Daida, T. F. Bersano-Begey, S. J. Ross, and J. F. Vesecky. Evolving feature-extraction algorithms: Adapting genetic programming for image analysis in geoscience and remote sensing. In *Proceedings of the 1996 International Geoscience and Remote Sensing Symposium*, pages 2077–2079. IEEE Press, 1996.

[32] J. M. Daida, R. G. Onstott, T. F. Bersano-Begey, S. J. Ross, and J. F. Vesecky. Ice roughness classification and ERS SAR imagery of arctic sea ice: Evaluation of feature-extraction algorithms by genetic programming. In *Proceedings of the 1996 International Geoscience and Remote Sensing Symposium*, pages 1520–1522. IEEE Press, 1996.

[33] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[34] E. R. Dougherty, editor. *Mathematical Morphology in Image Processing*. Marcel Dekker, New York, 1993.

[35] E. R. Dougherty and R. A. Latufo. *Hands-on Morphological Image Processing*, volume TT59. Spie Press, 2003.

[36] E. R. Dougherty and R. P. Loce. *Efficient design strategies for the optimal binary digital morphological filter: probabilities, constraints, and structuring element libraries.*, chapter 2, pages 43–92. Marcel Dekker, New York, 1993. crossref Dougherty:1993.

[37] M. Ebner. On the evolution of edge detectors for robot vision using genetic programming. In H. M. Groß, editor, *Workshop SOAVE '97 - Selbstorganisation von Adaptivem Verhalten, VDI Reihe 8 Nr. 663*, pages 127–134, Düsseldorf, 1997. VDI Verlag.

[38] M. Ebner. On the evolution of interest operators using genetic programming. In R. Poli, W. B. Langdon, M. Schoenauer, T. Fogarty, and W. Banzhaf, editors, *Late Breaking Papers at EuroGP'98: the First European Workshop on Genetic Programming*, pages 6–10, Paris, France, 14-15 April 1998. CSRP-98-10, The University of Birmingham, UK.

[39] A. Esparcia-Alcazar and K. Sharman. Genetic programming for channel equalisation. In R. Poli, H. M. Voigt, S. Cagnoni, D. Corne, G. D. Smith, and T. C. Fogarty, editors, *Evolutionary Image Analysis, Signal Processing and Telecommunications: First European Workshop, EvoIASP'99 and EuroEcTel'99*, volume 1596 of *LNCS*, pages 126–137, Goteborg, Sweden, 28-29 May 1999. Springer-Verlag.

[40] A. Fukunaga and A. Stechert. Evolving nonlinear predictive models for lossless image compression with genetic programming. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 95–102, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

[41] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Mass., 1989.

[42] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Pearson Education, Reading, MA, USA, 2nd edition, 2002.

[43] 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 April 2000. Springer-Verlag.

[44] H. F. Gray and R. J. Maxwell. Genetic programming for multi-class classification of magnetic resonance spectroscopy data. 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*, page 137, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[45] L. Gritz and J. K. Hahn. Genetic programming for articulated figure motion. *Journal of Visualization and Computer Animation*, 6(3):129–142, July 1995. ISSN 1049-8907.

[46] F. Gruau. Genetic micro programming of neural networks. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 24, pages 495–518. MIT Press, 1994.

[47] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD thesis, Laboratoire de l'Informatique du Parallilisme, Ecole Normale Superieure de Lyon, France, 1994.

[48] C. C. Han and K. C. Fan. A greedy and branch and bound searching algorithm for finding the optimal morphological erosion filter on binary images. *Signal Processing Letters*, 1:41–44, 1994.

[49] S. Handley. Predicting whether or not a nucleic acid sequence is an E. coli promoter region using genetic programming. In *Proceedings of the First International Symposium on Intelligence in Neural and Biological Systems INBS-95*, pages 122–127, Herndon, Virginia, USA, 29-31 May 1995. IEEE Computer Society Press.

[50] C. Harris and B. Buxton. Evolving edge detectors. Research Note RN/96/3, UCL, Gower Street, London, WC1E 6BT, UK, January 1996.

[51] C. Harris and B. Buxton. Low-level edge detection using genetic programming: performance, specificity and application to real-world signals. Research Note RN/97/7, UCL, Gower Street, London, WC1E 6BT, UK, 1997.

[52] N. R. Harvey. *New Techniques for the Design of Morphological Filters using Genetic Algorithms.* PhD thesis, University of Strathclyde, Glasgow, UK, 1997.

[53] N. R. Harvey and S. Marshall. *Mathematical Morphology and Its Applications to Image Procesing*, chapter Using Genetic Algorithms in the Design of Morphological Filters, pages 53–59. Kluwer Academic Publishers, 1994.

[54] N. R. Harvey and S. Marshall. Rank-order morphological filters: A new class of filters. In *IEEE Workshop on nonlinear signal and image processing*, pages 975–978, Halkidiki, Greece, 1994.

[55] N. R. Harvey and S. Marshall. The use of genetic algorithms in morphological filter design. *Signal Processing: Image Communication*, 8(1):55–72, Jan 1996.

[56] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. Iterative design of morphological binary image operators. *Optical Engineering*, 39(12):3106–3123, 2000.

[57] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. A switching algorithm for design of optimal increasing binary filters over large windows. *Pattern Recognition*, 33(6):1059–1081, 2000.

[58] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, USA, 1975.

[59] H. S. Hong. Digital image restoration using genetic programming. In J. R. Koza, editor, *Genetic Algorithms and Genetic Programming at Stanford 1999*, pages 68–75. Stanford Bookstore, Stanford, California, 94305-3079 USA, 15 March 1999.

[60] T. Hong and J. Hull. Algorithms for post-processing ocr results with visual inter-word constraints. In *Proc. Int. Conf. on Image Processing*, volume 3, pages 312–15, 1995.

[61] D. E. Hoskins and J. Vagners. Image compression using iterated function systems and evolutionary programming: Image compression without image metrics. In *26th Asilomar Conference on Signals, Systems and Computers.*, 1992.

[62] D. Howard, S. C. Roberts, and R. Brankin. Target detection in imagery by genetic programming. *Advances in Engineering Software*, 30(5):303–311, 1999.

[63] H. Huttunen, P. Kuosmanen, L. Koskinen, and J. Astola. Optimization of soft-morphological filters by genetic algorithms. *Image Algebra and Morphological Image Processing*, 2300:13–24, 1994.

[64] S. Isaka. An empirical study of facial image feature extraction by genetic programming. In J. R. Koza, editor, *Late Breaking Papers at the 1997 Genetic Programming Conference*, pages 93–99, Stanford University, CA, USA, 13–16 July 1997. Stanford Bookstore.

[65] J. Jiang and D. Butler. An adaptive genetic algorithm for image data compression. In J. R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28-31, 1996*, pages 83–87, Stanford University, CA, USA, 28–31 July 1996. Stanford Bookstore.

[66] M. P. Johnson, P. Maes, and T. Darrell. Evolving visual routines. In R. A. Brooks and P. Maes, editors, *ARTIFICIAL LIFE IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 198–209, MIT, Cambridge, MA, USA, 6-8 July 1994. MIT Press.

[67] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Application of genetic programming for multicategory pattern classification. *IEEE Transactions on Evolutionary Computation*, 4(3):242–258, September 2000.

[68] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Genetic programming based pattern classification with feature space partitioning. *Information Sciences*, 131(1-4):65–86, January 2001.

[69] M. Köppen and M. Teunis. Adaptation of morphological masks by genetic algorithms. In *Proc. ANNES'95*, Dunedin, New Zealand, 1995. IEEE.

[70] T. M. D. Kovacs. *A Comparison of Strength and Accuracy-Based Fitness in Learning Classifier Systems*. PhD thesis, School of Computer Science, The University of Birmingham, UK, 2001.

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

[72] J. R. Koza. Simultaneous discovery of detectors and a way of using the detectors via genetic programming. In *1993 IEEE International Conference on Neural Networks*, volume III, pages 1794–1801, San Francisco, USA, 1993. IEEE.

[73] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge Massachusetts, May 1994.

[74] J. R. Koza. *Genetic Programming II Videotape: The next generation*. MIT Press, 55 Hayward Street, Cambridge, MA, USA, 1994.

[75] J. R. Koza, D. Andre, F. H. Bennett III, and M. Keane. *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufman, April 1999.

[76] J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane, and S. Brave. *Genetic Programming III Videotape: Human Competitive Machine Intelligence*. Morgan Kaufmann, 340 Pine Street - 6th Floor, San Francisco, CA 94104, USA, 1999.

[77] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

[78] J. R. Koza and James P. Rice. *Genetic Programming: The Movie*. MIT Press, Cambridge, MA, USA, 1992.

[79] P. Kraft, N. R. Harvey, and S. Marshall. Parallel genetic algorithms in the optimization of morphological filters: A general design tool. *Journal of Electronic Imaging*, 6(4):504–516, 1997.

[80] P. Kraft, M. Nölle, G. Schreiber, S. Marshall, and H. Burkhardt. A parallel genetic algorithm for optimizing morphological filters on inhomogenous workstation clusters. Technical Report TR-402-95-012, 1995. Available at citeseer.nj.nec.com/article/kraft95parallel.html on 14.10.04.

[81] M. Kupinski and M. Anastasio. Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characterisitic curves. *IEEE Transactions on Medical Imaging*, 18, 1999.

[82] W. B. Langdon and P. Nordin. Evolving hand-eye coordination for a humanoid robot with machine code genetic programming. In J. F. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tettamanzi, and W. B. Langdon, editors, *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *LNCS*, pages 313–324, Lake Como, Italy, 18-20 April 2001. Springer-Verlag.

[83] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

[84] S. Loncaric and A. P. Dhawan. Optimal shape description using morphological signature transform via genetic algoritm. *SPIE Proceedings 2030: Image Algebra and Morphological Image Processing IV*, 2030:121–127, 1993.

[85] S. Loncaric and A. P. Dhawan. Near-optimal mst-based shape description using genetic algorithms. *Pattern Recognition*, 28(571–579), 1995.

[86] M. C. Maccarone. Fuzzy mathematical morphology: Concepts and applications. *Vistas in Astronomy*, 40(4):469–477, 1996.

[87] B. Masand. Optimising confidence of text classification by evolution of symbolic expressions. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 21, pages 445–458. MIT Press, 1994.

[88] M. Mignotte, C. Collet, P. Pérez, and P. Bouthemy. Hybrid genetic optimization and statistical model based approach for the classification of shadow shapes in sonar imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2):129–141, 2000.

[89] T. Mitchell. *Machine Learning*. Mc Graw Hill, 1997.

[90] S. K. Mitra, C. A. Murthy, and M. K. Kundu. Technique for fractal image compression using genetic algorithms. *IEEE Transactions on Image Processing*, 7:586–593, 1998.

[91] M. Nachtegael and E. E. Kerre. Connections between binary, gray-scale and fuzzy mathematical morphologies. *Fuzzy Sets and Systems*, 124(11–16):73–85, 2001.

[92] T. C. Nguyen, D. S. Goldberg, and T. S. Huang. Evolvable modeling: structural adaptation through hierarchical evolution for 3-D model-based vision. Technical report, Beckman Institute and Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801, USA, 1993.

[93] Y. Nong, L. Yushu, and X. Qifu. The use of genetic algorithms in morphological filter design. In *Proceedings of the 5th International Conference on Signal Processing*, volume 1, pages 476 –479, 2000. WCCC-ICSP 2000.

[94] 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.

[95] S. K. Pal, D. Bhandari, and M. K. Kundu. Genetic algorithms for optimal image enhaqncement. *Pattern recognition letters*, 15:261–271, 1994.

[96] S. K. Pal, A. Ghosh, and M. K. Kundu. *Soft Computing for Image Processing*. Physica-Verlag, New York, 2000.

[97] S. K. Pal and P. P. Wang. *Genetic Algorithms for Pattern Recognition*. CRC Press, 1996.

[98] S. Perkins. Evolving effective visual tracking through shaping. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1156–1161, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[99] S. Perkins. Evolving robot vision: Increasing performance through shaping. In *1999 Congress on Evolutionary Computation*, pages 381–388, Piscataway, NJ, 1999. IEEE Service Center.

[100] R. Poli. Genetic programming for feature detection and image segmentation. In T. C. Fogarty, editor, *Evolutionary Computing*, number 1143 in Lecture Notes in Computer Science, pages 110–125. Springer-Verlag, University of Sussex, UK, 1-2 April 1996.

[101] 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, 28–31 1996. MIT Press.

[102] R. Poli. Genetic programming for image analysis. Technical Report CSRP-96-1, University of Birmingham, UK, January 1996.

[103] R. Poli. Some steps towards a form of parallel distributed genetic programming. In *The 1st Online Workshop on Soft Computing (WSC1)*, http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/, 19–30 August 1996. Nagoya University, Japan.

[104] R. Poli. Sub-machine-code GP: New results and extensions. In R. Poli, P. Nordin, W. B. Langdon, and T. C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'99*, volume 1598 of *LNCS*, pages 65–82, Goteborg, Sweden, 26-27 May 1999. Springer-Verlag.

[105] R. Poli and S. Cagnoni. Evolution of pseudo-colouring algorithms for image enhancement with interactive genetic programming. Technical Report CSRP-97-5, University of Birmingham, School of Computer Science, January 1997.

[106] 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, 13-16 July 1997. Morgan Kaufmann.

[107] R. Poli and W. B. Langdon. Sub-machine-code genetic programming. In L. Spector, W. B. Langdon, U. O'Reilly, and P. J. Angeline, editors, *Advances in Genetic Programming 3*, chapter 13, pages 301–323. MIT Press, Cambridge, MA, USA, June 1999.

[108] M. I. Quintana, H. A. Montes, and J. C. Cuevas. Machine learning as an alternative approach to mathematical morphology. In *International Symposium on Mathematical Morphology*. Communicated on September 11th, 2004.

[109] M. I. Quintana, R. Poli, and E. Claridge. Approach for morphological algorithm design for binary images using genetic programming. *Genetic Programming and Evolvable Hardware*. Accepted, manuscript in progress.

[110] M. I. Quintana, R. Poli, and E. Claridge. Genetic programming for mathematical morphology algorithm design on binary images. In M. Sasikumar, J. J. Hegde, and M. Kavitha, editors, *Proceedings of the International Conference KBCS-2002*, pages 161–170, Mumbai, India, 19-21 Dec 2002. Vikas.

[111] 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 April 2003. Springer-Verlag.

[112] A. Quirin and J. Korczak. Evolutionary approach to discovery of classification rules from remote sensing images. 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 391–402, University of Essex, England, UK, 14-16 April 2003. Springer-Verlag.

[113] P. J. Rauss, J. M. Daida, and S. Chaudhary. Classification of spectral imagery using genetic programming. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H. G. Beyer, editors, *Proceedings of the Genetic and Evolutionary*

Computation Conference (GECCO-2000), pages 726–733, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[114] C. W. Reynolds. An evolved, vision-based behavioral model of coordinated group motion. In Meyer and Wilson, editors, *From Animals to Animats (Proceedings of Simulation of Adaptive Behaviour)*. MIT Press, 1992.

[115] D. Rivero, J. R. Rabu nal, J. Dorado, and A. Pazos. Using genetic programming for character discrimination in damaged documents. In G. R. Raidl *et al.*, editor, *Applications of Evolutionary Computation, EvoWorkshops 2004*, number 3005 in LNCS, pages 349–358, Berlin, 2004. Springer.

[116] S. C. Roberts and D. Howard. Evolution of vehicle detectors for infrared linescan imagery. In R. Poli, H. M. Voigt, S. Cagnoni, D. Corne, G. D. Smith, and T. C. Fogarty, editors, *Evolutionary Image Analysis, Signal Processing and Telecommunications: First European Workshop, EvoIASP'99 and EuroEcTel'99*, volume 1596 of *LNCS*, pages 110–125, Goteborg, Sweden, 28-29 May 1999. Springer-Verlag.

[117] S. C. Roberts and D. Howard. Genetic programming for image analysis: Orientation detection. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H. G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 651–657, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[118] D. Robilliard, M. Chami, C. Fonlupt, and R. Santer. Using genetic programming to tackle the ocean color problem. In *Proceedings of Ocean Optics XV*, pages 124–127, October 2000.

[119] B. J. Ross, F. Fueten, and D. Y. Yashkir. Edge detection of petrographic images using genetic programming. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector,

I. Parmee, and H. G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 658–665, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[120] S. J. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[121] M. Schmitt. Mathematical morphology and artificial intelligence: An automatic programming system. *Signal Processing*, 16(4):389–401, April 1989.

[122] M. Schoenauer. *Evolutionary Computation*. MIT Press, 2004.

[123] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[124] J. Serra. *Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances*. Academic Press, 1988.

[125] J. Shanahan, B. Thomas, M. Mirmehdi, T. Martin, N. Campbell, and J. Baldwin. A soft computing approach to road classification. *Journal of Intelligent and Robotic Systems*, 29(4):349–387, December 2000.

[126] J. Sherrah. Automatic feature extraction using genetic programming. In J. R. Koza, editor, *Late Breaking Papers at the 1997 Genetic Programming Conference*, page 298, Stanford University, CA, USA, 13–16 July 1997. Stanford Bookstore.

[127] J. Sherrah. *Automatic Feature Extraction for Pattern Recognition*. PhD thesis, University of Adelaide, South Australia, July 1998.

[128] J. Sherrah, R. E. Bogner, and B. Bouzerdoum. Automatic selection of features for classification using genetic programming. In *Proceedings of the 1996 Australian New Zealand Conference on Intelligent Information Systems*, pages 284–287, New York, NY. IEEE.

[129] J. R. Sherrah, R. E. Bogner, and A. Bouzerdoum. The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. 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 304–312, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[130] F. Y. Shih and J. Moh. Implementing morphological operations using programmable neural networks. *Pattern Recognition*, 1:89–99, 1992.

[131] N. T. Siebel and S. Maybank. Fusion of multiple tracking algorithms for robust people tracking. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Proceedings of the 7th European Conference on Computer Vision (ECCV 2002), København, Denmark*, volume IV, pages 373–387, May 2002.

[132] K. Sims. Artificial evolution for computer graphics. Technical Report TR-185, Thinking Machines Corporation, 1991.

[133] K. Sims. Evolving images. Lecture, March 1993. Lecture presented at Centre George Pompidou, Paris on March 4, 1993.

[134] K. Sims. Evolving 3D morphologgy and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 28–39, Cambridge, MA, 1994. MIT Press.

[135] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Chapman and Hall, London, UK, 1993.

[136] L. Spector and A. Alpern. Induction and recapitulation of deep musical structure. In *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI'95 Workshop on Music and AI*, pages 741–747, Montreal, Quebec, Canada, 20-25 August 1995.

[137] S. A. Stanhope and J. M. Daida. Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery. In V. William 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 March 1998. Springer-Verlag.

[138] C. D. Stefano, A. D. Cioppa, and A. Marcelli. Character preclassification based on genetic programming. *Pattern Recognition Letters*, 23(12):1439–1448, 2002.

[139] C. D. Stefano, A. D. Cioppa, A. Marcelli, and F. Matarazzo. Grouping character shapes by means of genetic programming. *Lecture Notes in Computer Science*, 2059:504–510, 2001.

[140] W. A. Tackett. Genetic generation of "dendritic" trees for image classification. In *Proceedings of WCNN93*, pages IV 646–649. IEEE Press, July 1993.

[141] 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 July 1993. Morgan Kaufmann.

[142] A. Teller. *Algorithm Evolution with Reinforcement for Signal Understanding*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1998.

[143] 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, 1–3 December 1995. IEEE Press.

[144] A. Teller and M. Veloso. A controlled experiment: Evolution for learning difficult image classification. In *Seventh Portuguese Conference On Artificial Intelligence*,

volume 990 of *Lecture Notes in Computer Science*, pages 165–176, Funchal, Madeira Island, Portugal, October 3-6 1995. Springer-Verlag.

[145] 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.

[146] A. L. Wiens and B. J. Ross. Gentropy: Evolutionary 2D texture generation. In D. Whitley, editor, *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pages 418–424, Las Vegas, Nevada, USA, 8 July 2000.

[147] 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.

[148] M. L. Wong and K. S. Leung. *Data Mining Using Grammar Based Genetic Programming and Applications*, volume 3 of *Genetic Programming*. Kluwer Academic Publishers, January 2000.

[149] X. Yao. *Evolutionary Computation: Theory and Applications*. World Scientific Publ., 1999.

[150] X. Yao, editor. *IEEE Transactions on Evolutionary Computation*. IEEE, 2004.

[151] 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.

[152] 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.

[153] P. Yu, V. Anastassopoulos, and A. N. Venetsanopoulos. Pattern recognition based on morphological shape analysis and neural networks. *Mathematics and Computers in Simulation*, 5:577–595, 1996.

[154] M. Zhang and U. Bhowan. Program size and pixel statistics in genetic programming for object detection. In G. R. Raidl, S. Cagnoni, J. Branke, D. W. Corne, R. Drechsler, Y. Jin, C. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. D. Smith, and G. Squillero, editors, *Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*, volume 3005 of *LNCS*, pages 377–386, Coimbra, Portugal, 5-7 April 2004. Springer Verlag.

[155] M. Zhang and V. Ciesielski. Genetic programming for multiple class object detection. In N. Foo, editor, *12th Australian Joint Conference on Artificial Intelligence*, volume 1747 of *LNAI*, pages 180–192, Sydney, Australia, 6-10 December 1999. Springer-Verlag.

[156] 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.

[157] M. Zhang and W. Smart. Genetic programming with gradient descent search for multiclass object classification. In M. Keijzer, U. O'Reilly, S. M. Lucas, E. Costa, and T. Soule, editors, *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, volume 3003 of *LNCS*, pages 399–408, Coimbra, Portugal, 5-7 April 2004. Springer-Verlag.

[158] M. Zhang and W. Smart. Multiclass object classification using genetic programming. In G. R. Raidl, S. Cagnoni, J. Branke, D. W. Corne, R. Drechsler, Y. Jin, C. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. D. Smith, and G. Squillero, editors,

Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC, volume 3005 of *LNCS*, pages 367–376, Coimbra, Portugal, 5-7 April 2004. Springer Verlag.

[159] P. Ziemeck and H. Ritter. Evolving low-level vision capabilities with the GENCODER genetic programming environment. In G. D. Smith, N. C. Steele, and R. F. Albrecht, editors, *ICANNGA97*, pages 78–82, University of East Anglia, Norwich, UK, 2-4 April 1997. Springer.

[160] D. Zongker, B. Punch, and B. Rand. Lilgp 1.01. user manual. Technical report, Michigan State University, 1996.