An Analysis of Evolutionary Computation used in Image Processing Techniques

John P. Cartlidge
BSc Artificial Intelligence and Mathematics 1999/2000

Summary

Upon considering the title, '*An Analysis of Evolutionary Computation in Image Processing Techniques'*, the following objectives were chosen :

- To research and review the techniques encompassed within image processing.
- To gain a thorough understanding of the principles involved in evolutionary computation and how they may be used.
- To compare evolutionary computation with competing image processing techniques, in the context of current research.
- To consider the reasons behind current trends in research and thus speculate how the field may develop.

The first objective was met through studying general texts about computer imaging, image processing, and machine vision. In order to research the techniques encompassed within image processing, it was of primary importance to define what is actually meant by the term image processing. It quickly become apparent that there is no 'strict' definition of image processing used throughout the field of computer imaging. Therefore, in order to fulfil this primary goal a boundary was proposed, for the purpose of this project, that included the techniques of compression, restoration, enhancement, and analysis. These techniques were then reviewed.

Objective two was completed by categorising evolutionary computation into its constituent parts; artificial life, evolution strategies, evolutionary programming, and genetic algorithms. The historical development of each was considered with comparisons made between the different techniques in terms of representation and fitness analysis, mutation, recombination, and selection. Evolutionary computation is shown to be a very powerful and versatile search optimisation tool.

Image processing techniques using evolutionary computation were reviewed for objective three. Unfortunately, problems arose when comparing evolutionary computation techniques with competing methods due to the lack of data in this area. Most techniques using evolutionary computation are either in the early stages of development, or are very specific. Although some comparisons can be made with contemporary techniques, a substantial analysis could not be performed without the necessary supporting data.

To complete objective four, the emerging trends appearing within image processing techniques using evolutionary computation were considered. Noticeably, most areas of research in this field gravitate towards genetic algorithms and image analysis, although other, 'more innovative' artificial life projects are in existence. I predict that artificial life is a tool that will be progressively used to yield great results in the future.

Acknowledgements

I would like to thank both Dr Nick Efford, for providing much welcomed supervision throughout the duration of this project, and Helen Pickup, for use of her computer during those long, lonely nights.

Contents

1. Introduction.

Artificial intelligence (AI) is an exiting and challenging subdivision of computer science, dedicated to tackling the problem of creating 'intelligent' behaviour within an information system. Although this problem is proving incredibly difficult to solve, the ultimate reward of 'imitating' human intelligence is so awe-inspiring that many, like myself, are driven in this pursuit.

It is usual for artificial intelligence projects to concentrate in specific areas. The field has grown so large as to make it unfeasible to use all AI techniques when attempting to solve a problem. For this reason, different areas of AI have become withdrawn from one another.

Image processing is a section of artificial intelligence concerned with the acquisition, restoration, enhancement, and analysis of images performed by a computer. Although the ultimate output would ideally be used as a visual tool for an autonomous robot, the techniques today are developed only so far as to aid much more mundane, though very useful, tasks.

Evolutionary computation (EC) is a relative of AI. EC differs from most AI techniques due to its bottom-up approach to solving problems. Inspired by biological evolution, this bottom-up technique works towards a complex solution from a simple starting point. In contrast, AI's top-down approach looks at the complex aim, before attempting to break this down into a set of simpler 'rules'.

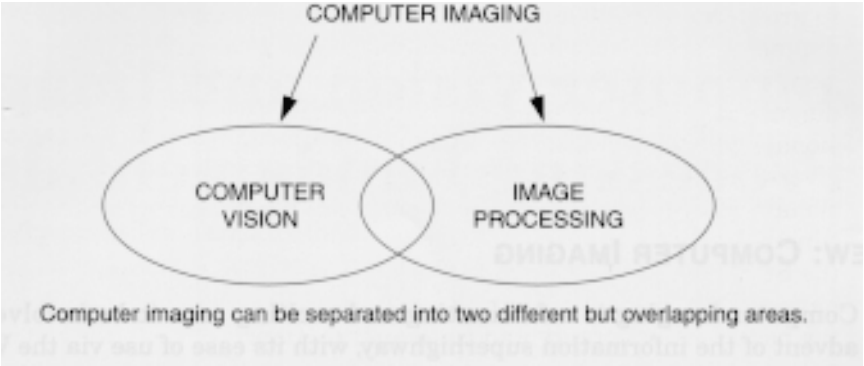
The aim of this project is to discover how evolutionary computation and image processing can be combined. This fascinating adventure shall firstly attempt to define what is understood by the term image processing, before studying the techniques that constitute evolutionary computation, and how they operate. A selection of projects incorporating both evolutionary computation and image processing shall then be examined, finally proceeded by a discussion of the general trends and developments.

2. Introduction to Image Processing.

2.1. Computer Imaging : An Overview.

Russ (1998) suggests that humans are primarily visual creatures. Not all animals depend on their eyes, as we do, for ninety-nine per cent or more of the information received about their surroundings. The exiting world of computer imaging derives its importance from the reliance upon the visual world that human beings have. A visual image conveys an extraordinarily large amount of information. It could be considered an understatement to say that *a picture is worth a thousand words*. Most images contain many millions of bits of data. It is for this reason that humans consider images as a vital and integral part of everyday life. Umbaugh (1998) proposes that computer imaging can be defined as the acquisition and processing of visual information by computer. By regarding the ultimate 'receiver' of the image, one can separate computer imaging into two fields; 1) computer vision and 2) image processing. *See Figure 2.1.*

Figure 2.1. Computer Imaging. (Umbaugh 1998).



Low-level image processing techniques require very little *a priori* knowledge about the content of images as they are to be examined and acted upon by people. In contrast, hi-level computer vision/image understanding applications are based on knowledge, goals and plans of how to achieve those goals, so that a computer can use the visual information directly. This is shown in Sonka et al. (1993, 1999).

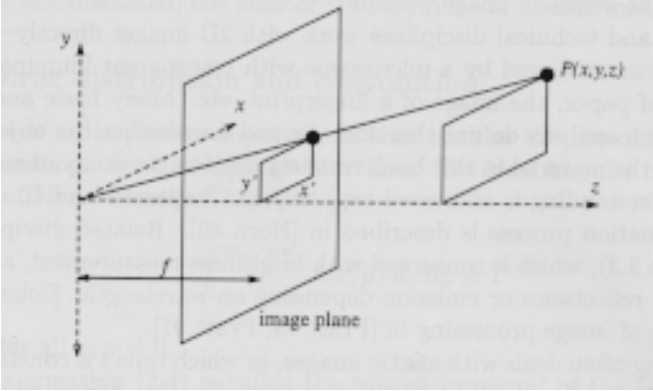
Rather than a definitive boundary between these high and low level processes, one must be aware of a region of overlap as shown in *Figure 2.1*. Application classification as exclusively image processing or computer vision can thus be a problem if in the region of low-level computer vision.

2.2. What is Image Processing?

Pearson (1991) states that image processing is a term used to describe operations carried out on images, with the aim of accomplishing some purpose. This may be a very general 'definition' and certainly too vague to answer the question 'what is image processing?', but the lack of precision helps to highlight the difficulty of rigorously defining a set of applications with 'fuzzy' boundaries. As already shown in section 2.1, image processing, like many disciplines, is not an exclusive set. Awcock and Thomas (1995) argue that the term image processing itself has become firmly associated with the limited objective of modifying images such that they are either: (a) corrected for errors introduced during acquisition or transmission ('restoration'); or (b) enhanced to overcome the weakness of the human visual system ('enhancement'). As such, the discipline of 'pure' image processing may be succinctly summarised as being concerned with "*a process which takes an image input and generates a modified image output.*" For the purpose of this chapter, we shall be concerned with the essential topics of image processing upon individual pixels. These are *Image Compression* (section 2.4), *Image Restoration* (Section 2.5) and *Image Enhancement* (section 2.6). Due to the overlap between computer vision and image processing, certain low-level computer vision applications should be considered when discussing image processing. For this reason, a section of this chapter shall consider *Image Analysis* (section 2.7), with its constituent parts, *Feature Extraction* and *Pattern Classification.*

2.3. What is an Image?

Figure 2.2. Perspective projection geometry. (Sonka et al. 1993).



Sonka et al. (1993) gives a brief insight into what we shall understand to be an image. He states that the image "can be modelled by a continuous function of two or three variables; in the simple case arguments are co-ordinates (x,y) in the plane, while if images change in time a third variable t may be added. The image function values correspond to the brightness at image points; this allows us to avoid the description of the very complicated process of image formation." *See Figure 2.2.*

2.4. Image Compression.

Image processing is often difficult due to the large amounts of data used to represent an image. One way of reducing the amount of data, by removing that which is inherently redundant, is to work with compressed data. The aim is to represent an image using a lower number of bits per pixel, without losing the ability to reconstruct the image. Methods are divided into two principal groups. *Information preserving* compressions permit error free data recovery, while compression methods *with loss of information* do not preserve the information completely. Although it may appear that information preserving methods are a necessity, in practice this is not the case. In

image processing a faithful reconstruction is not always required so long as data compression does not cause *significant* changes in an image. (see Sonka et al. 1993, 1999)

2.5. Image Restoration.

Image restoration techniques attempt to reduce any degradation present within an image. In order for restoration to be successfully achieved, it is important to have previous knowledge as to the nature of the degradation. Hopefully, image degradation can then be suppressed using a process modelled upon this *a priori* knowledge so that the degradation can be reversed. Examples of degradation types include motion blurring, noise reduction from electric sources or geometric distortion due to lens aberrations. The major tools used in image restoration fall into the categories of *spatial filters*, *frequency domain filters* and *geometric transforms.*

For further reading see Awcock and Thomas (1995), Pearson (1991), Russ (1998), Sonka et al. (1993, 1999) and Umbaugh (1998).

2.6. Image Enhancement.

The overall quality of an image may be improved, or selected features enhanced, by virtue of image enhancement techniques. The aim of image enhancement is to use these techniques in order to fulfil a specific objective required for use in a particular image processing problem. Techniques are thus problem specific and are often not universal tools, having to be specifically designed for each new implementation. Image enhancement techniques consist of *point*, *mask* and *global* operations, using both the frequency and spatial domains. The main tools fall under the categories of *grey-scale modification*, *image sharpening* and *image smoothing*. For further reading see Awcock and Thomas (1995), Pearson (1991), Russ (1998), Sonka et al. (1993, 1999) and Umbaugh (1998).

2.7. Image Analysis.

2.7.1. Feature Extraction.

Operating upon two-dimensional image arrays, feature extraction seeks to identify features, or characteristics, of objects contained within an image. These characteristics can be used to describe the object, or attributes of the object, such as shape or colour. A descriptive list of the object is produced, known as a *feature vector,* which is then used in the subsequent task of pattern classification.

2.7.2. Pattern Classification.

The 'final' task of image processing is that of pattern classification. This problem consists of taking an object within an image before attempting to classify it by deciding which specific group of objects it belongs to. As there are a number of possible choices of groups, the problem of which to select arises. Awcock and Thomas (1995) state that

> "there are three main approaches to group classification; (a) Statistical based classification relies on defining a set of decision rules based on standard statistical theory. (b) Syntactic pattern classification utilises the underlying structure of the patterns themselves. (c) Alternative approaches use architectures such as neural nets which can be trained to correctly associate input patterns."

For further reading see Awcock and Thomas (1995), Pearson (1991), Russ (1998), Sonka et al.(1993, 1999) and Umbaugh (1998).

2.8. Chapter Summary.

Chapter two has introduced some of the basic concepts incorporated within image processing, attempting to segment the field into its more recognised constituents. As this brief foundation is by no means comprehensive, one should refer to any references for further reading around the subject. Image compression, restoration, enhancement and analysis have each briefly been described. Unfortunately, for the purpose of this project it would be impossible to go into detail about these topics as each in its own right could easily contain enough information to encompass an entire textbook. Although detailed knowledge of these topics is not necessary, it is now assumed that the reader will be at least familiar with these concepts.

3. Introduction to Evolutionary Computation.

3.1. Why Evolution?

Since Darwin changed science forever with his theories of natural selection, evolution has been an endless source of amazement. Every habitable environment known to man has a diverse collection of very specialist and complex species. Survival of the fittest appears to direct the evolution of species towards solutions of environmental problems. Evolution does not have 'knowledge' of a pre-determined goal, rather, endless possibilities are tested before acceptable solutions are 'discovered'. Gould (1991) suggests that

> "wings were probably not originally designed to let animals fly. Insect wings probably were at first heat-regulating structures, which happened to give some gliding abilities as a side-effect of increasing the global size of the animal."

It was only through the selective advantage of reproduction that the emergent benefit of flight slowly became the main function of wings. Evolution is equivalent to a process of trial and error, with many millions of parallel trials constantly undertaken. Although it may appear so at first, evolution is not optimisation.

> "Consider the human eye, where light sensors are 'backwards'. Or consider the way we are eating and breathing through our mouths. These non-optimal solutions are due to the genes which our distant ancestors happened to have." (Haataja 1999).

One can easily overlook the fact that the bewildering 'discoveries' made by evolution are often not optimal.

3.2. Evolutionary Computation : A History.

Since the birth of the computer, there have been attempts by many to try to harness the power of evolution through information processes known as evolutionary computation (EC). Although these attempts may have varied significantly both in objective and technique, all still have the underlying connection of attempting to use biological evolution as a guideline for problem

solving, or research, in information technology, for optimisation. Historically, four main paradigms have evolved, quite separately, throughout EC research. These include *artificial life* (Alife), *evolution strategies* (ES), *evolutionary programming* (EP), and *genetic algorithms* (GA). Only recently have these early characterisations become less useful in describing the enormous variety of current activities in the field. In some cases the boundaries have become so 'fuzzy' that categorisation becomes worthless. It is for this reason that, although the reader should have a historical understanding of each paradigm, the techniques shall be discussed with reference to EC as a whole in sections 3.3.1 to 3.3.4. Considering the techniques in depth for each paradigm separately could evoke a misunderstanding in the reader with regards to present research.

3.2.1. Artificial Life.

> "By synthesising the mechanisms underlying the evolutionary process in computers and other 'non biological' media, we can discover solutions to engineering problems that have long resisted our traditional approaches", Langton (1997).

Upon introduction to Alife, it can appear almost a contradictory concept that to achieve complex objectives one should approach problems by waiting for emergent behaviour from a simple set of localised rules. With Alife, however, this is just the case. Throughout the literature of Alife, Von Neumann is often cited as the pioneer with his early works on cellular automata. (see Emmeche 1994, and Levy 1992). These 'organisms of pure logic' were essentially self-reproducing automata living within a grid space much like a chess board. A major development in the evolution of cellular automata came with Horton Conway's game of 'Life' in which each cell had only two possible states. Fundamentally, the aim of 'Life' was to observe the emergent behaviour from organisms capable of mutual interaction in conjunction with genetic modification and recombination of genomes. Rather than try to dictate the direction of evolution towards a predetermined goal, Conway was more interested in behavioural patterns within the population and so no fitness function was attributed to the cells (Fogel 2000). It was observed that complexity can quickly arise from simulations of initially very simple cell patterns.

The Tierra project of Ray (1991) consisted of a simulation of assembly code programs 'living' within the CPU and operating system, competing for CPU time. Ray (1991) states

> "sets of machine instructions similar to those used in the Tierra simulation have been shown to be capable of universal computation. This suggests that evolving machine codes should be able to generate any level of complexity."

Bach (1996) states that, "[throughout Alife] in many cases the agents are equipped with internal rules or strategies determining their behaviour, and an evolutionary algorithm is used for evolving

these strategies." These evolutionary algorithms cover evolution strategies, evolutionary programming, and genetic algorithms. A discussion of each algorithm follows. For further reading on Alife, see Brooks and Maes (1994), Langton et al. (1991), and Langton (1994).

3.2.2. Evolution Strategies.

"Evolution Strategies are a joint development of Bienot, Rechenberg and Schwefel, who did preliminary work in this area in the 1960s at the Technical University of Berlin (TUB)", Bach (1996). All work used vectors as a framework for optimisation algorithms. Fogel (2000) described their "evolution strategy" as follows:

1) The problem is defined as finding the real valued n-dimensional vector **x** associated with the functional $F(\mathbf{x}):R^n \rightarrow R$, presented as a minimisation process.
2) An initial population of parent vectors $\mathbf{x}_i$, i = 1,…,P is selected at random.
3) Offspring vector, $\mathbf{y}_i$ created from each parent $\mathbf{x}_i$, i = 1,…,P by adding a Gaussian random variable with zero mean and preselected standard deviation to each component of **x**.
4) Selection determines which vectors to maintain by ranking errors $F(\mathbf{x}_i)$ and $F(\mathbf{y}_i)$, i = 1,…,P. P vectors with least errors become new parents for next generation.
5) Continue generating trials until sufficient solution reached or available computation exhausted.

For a discussion on evolutionary strategies, refer to Bach (1996). Although initial efforts examined the proceeding algorithm with a *single parent single offspring* approach, termed a (1+1)-ES[1], recent approaches explored are much more efficient, denoted (μ+λ)-ES and (μ,λ)-ES. These incorporate multi-population parents and offspring.

Bach (1996) states;

> "currently the (μ,λ)-ES characterises the state-of-the-art in ES research… The main quality of this strategy is seen in its ability to incorporate the most important parameters of the strategy, [i.e. standard deviations and correlation coefficients of normally distributed mutation], into the search process, such that optimisation not only takes place on object variables, but also on strategy parameters according to the actual local topology of the objective function".

3.2.3. Evolutionary Programming.

Evolutionary programming (EP) essentially began with the works of Fogel et al. (1966).

> "To simulate evolution, it is necessary to choose a mathematical representation for the organism. Let the organism take the form of a finite-state machine…the succession of finite-state machines considered is dependent partly upon the statistical process of mutation and partly upon the environment", Fogel et al. (1966).

[1] See appendix B for ES terminology and definitions.

The environment was described as a sequence of symbols taken from a finite alphabet, with an evolutionary problem of evolving an algorithm that would operate on the symbols to maximise an output. Finite-state machines provided the necessary representation of the required behaviour.

Although evolution strategies and evolutionary programming are surprisingly similar, development proceeded independently of each other until 1992. EP operated by exposing a population of finite machines to the environment. A payoff function was used as a fitness function, based upon output error. Each parent had one offspring, created by random mutation, before the whole population became subject to *non-regressive*[2] evolution, i.e. a machine must rank in the top half of the population in order to survive. Evolutionary programming does not have genetic processes such as crossover and inversion (for further details refer to Bach 1996).

EP became a very successful tool for sequence prediction and game playing strategies, but unfortunately had to cope with a massive state-space of machines, (see Fogel 2000). A complex problem can quickly result in a state space that makes searching infeasible. Atmar (1976) calculated that the number of possible configurations for a finite-state machine given **n** states, **a** input symbols, and **b** output symbols is given by

$$\mathbf{N = (n^a b^a)^n}$$

The 1980s saw the use of Moore machines used in EP (Fogel 2000) and more recently diverse combinatorial optimisation problems.

3.2.4. Genetic Algorithms.

In the 1960s and 1970s Holland (1975) invented the GA along with his students and colleagues whilst teaching at the University of Michigan. In contrast to EP and ES, Holland's original goal was not to solve specific problems, but rather to formally study natural adaptation synthesised within computer systems.

Holland's GA is a method of moving from one population of 'chromosomes' to a new population with the genetics-inspired operators of crossover, mutation, and inversion. Each chromosome consists of binary strings, representing the information stored as the bases in a strand of DNA. The GA of Holland was a major innovation. Evolutionary programming used only mutation to

[2] In non-regressive evolution, the score of the offspring must be equal to or exceed that of the parent in order to survive.

provide variation, and Evolution Strategies did not incorporate many-individual populations and crossover until much later. Mitchell (1996) suggests "Holland was the first to attempt to put computational evolution on a firm theoretical footing".

Crossover in genetic algorithms is always a sexual operator. Two parents from the population are chosen, recombined to form two new individuals, before one of the offspring is discarded at random. Traditionally, *one-point crossover* chooses one position within the bit-string at random, before exchanging the bits to the right of that between both individuals. This is clearly inferior to other crossover operations, with respect to performance results. *Multi-point crossover* chooses **z** points along the bit-string where crossover takes place. Although it has been indicated that **z** = 8 is optimal, the standard used in implementations is two-point crossover. *Uniform crossover* exchanges segments of length one bit, with each bit exchanged or not by the toss of a coin, this has also been shown to improve performance, relative to one-point crossover.

Punctuated crossover is the only known attempt to incorporate self-adaptation of some strategy parameters into GAs, copying the self-adaptive ability of ES and EP. Punctuated crossover self-adapts both number and position of crossover points for a multi-point crossover. Today, researches often use the term "genetic algorithm" to describe something far from the original conception of Holland. De Jong (1985).

3.3. Evolutionary Algorithms: A Comparison.

> "Although all [ES, GA and EP] algorithms share an identical meta-model, each one emphasises different features as being the most important to a successfully modelled evolutionary process." Bach (1996).

In this section the main components of evolutionary algorithms are discussed, taking into account the differing emphasis each algorithms takes.

3.3.1. Representation and Fitness Analysis.

As already seen, in sections 3.2.2 - 3.2.4, evolutionary algorithms have a range of differing representations. The real-valued representations of ES vectors and EP finite-machines have a strong contrast with the binary-valued bit strings used for GA. Fitness analysis also varies across algorithms, for example ES uses the objective function value as a fitness value, yet EP and GA tend to use a scaled objective function value. See Bach 1996 for a detailed discussion.

Different algorithm representations tend to result from the initial objective, ranging from universal binary encodings to problem-specific encodings for real-valued parameter optimisation problems. The EC community

differs widely on opinions and strategies for selecting appropriate representations. De Jong (1999) concludes;

> "Although there are strong historical associations between GA and binary string representations, between ES and vectors of real numbers, and between EP and finite-state machines, it is now quite common to use representations other than the traditional ones in order to effectively evolve more complex objects such as symbolic rules, LISP codes, or neural networks. Claiming one EA approach is better than another on a particular class of problems is not meaningful anymore without motivating and specifying the representations chosen."

3.3.2. Mutation.

"Apart from representational issues, the most conspicuous difference [between algorithms] is given by the interpretation of the role of genetic operators", Back (1996). As in biological systems, mutation is a key concept throughout evolutionary algorithms. There is however a great differential in the algorithms' emphasis placed upon the mutation operator. As an analogy to the natural model of evolution, genetic algorithms usually have a very small mutation probability, often in the region of 0.01 or 0.001. In some cases GAs are implemented with no mutation operator at all. In complete contrast, evolution strategies use Gaussian mutation as their main operator, whilst evolutionary programming relies upon mutation entirely. Using no recombination (section 3.3.3), asexual mutation is the only reproductive operator that EP uses

during offspring inheritance. ES and EP, in similar ways, can evolve their own strategy parameters during the search, exploiting an implicit link between appropriate internal model and good fitness values. This is known as *self-adaptation*. Bach (1996) states;

> "Both evolution strategies and evolutionary programming rely on these self-adaptation processes for step-size control and correlation of mutations. This concept was tested in the context of genetic algorithms only in the form of the punctuated crossover operator but it never gained acceptance due to a combination of a lack of success and acknowledgement."

3.3.3. Recombination.

Recombination, the second genetic operator, is completely ignored throughout evolutionary programming and appears only as a background operator in evolution strategies. ES indicate a necessity to use recombination on strategy parameters only, in order to facilitate self-adaptation. Evolutionary programming indicates that, with mutation alone, self-adaptation can be achieved, although combining operators in ES can be shown to improve efficiency. In complete contrast, recombination plays the dominating role for genetic algorithms, with mutation almost completely neglected. Recombination in GAs consists of z-point and uniform crossover and as such is always sexual. In evolution strategies, recombination is discrete and intermediate and can be sexual or panmitic[3]. For a thorough review of genetic operators throughout evolutionary algorithms, see Back (1996).

3.3.4. Selection.

Selection refers to the method of choosing members from a population that will 'survive' until the next generation. Although differently implemented, both evolutionary programming and genetic algorithms insist on probabilistic selection. In contrast, evolution strategies use strictly deterministic *extinctive*[4] selection. Due to the asexual property of mutation, the offspring population in EP is the same size as the parent population. The selection mechanism can thus be interpreted as a kind of probabilistic $(\mu+\mu)$-selection. While in evolutionary programming some

[3] Panmitic selection has no biological basis. One parent is chosen and held fixed, while for each component of its vectors, the second parent is randomly chosen anew from complete population.
[4] Extinctive selection implies that the worst individuals are definitely excluded from selection.

individuals are excluded from selection, genetic algorithms make use of *preservation*.[5] An alternative to using absolute fitness values is *rank-based selection*. This can be used to utilise the indexes of individuals when ordered according to fitness values, to calculate the corresponding selection probabilities. This helps evolution continue, even with a population of very fit individuals. An example of this can be seen in Husbands et al. (1998).

Finally, selection also varies across evolutionary algorithms due to *elitism*. This is where the best individual is assigned a maximum score so that survival is guaranteed. In evolutionary programming the elitist property is implicit to selection, while it occurs in evolution strategies $(\mu+\lambda)$-ES and GA only if requested explicitly. For further reading see Back (1996).

3.4. Chapter Summary.

Chapter three has introduced the reader to the basic concepts of evolution and how these have been adopted in various ways to simulate evolution within an artificial system. Evolutionary computation's historic characterisations of artificial life, evolution strategies, evolutionary programming, and genetic algorithms have been introduced. In conjunction, the main components of any evolutionary algorithm, representation and fitness analysis, mutation, recombination, and selection, have also been discussed. For a more detailed understanding of evolutionary computation, the reader is referred throughout the chapter to recommended texts. Fogel (1999) summarises well;

> "Evolutionary computation, the term now used to describe the field of investigation that concerns all evolutionary algorithms, offers practical advantages to the researcher facing difficult optimisation problems. These advantages are multi-fold, including the simplicity of the approach, its robust response to changing circumstance, its flexibility, and many other facets."

[5] Preservation implies that *each* individual receives non-zero selection probability.

4. Image Processing Techniques Using Evolutionary Computation.

4.1. The Overlap Between Evolutionary Computation And Image Processing.

When embarking upon research into image processing techniques using evolutionary computation, it quickly becomes apparent that the field is not extensive. *Table 4.1*, below, displays the areas where work is currently taking place. It is overwhelmingly obvious that many areas are ignored, with most projects clustered into a small proportion of the possible regions.

Table 4.1. How Evolutionary Computation and Image Processing merge.
The shaded regions indicate in which areas work is undertaken.

Image Processing	Evolutionary Computation: A-life	EP	ES	GA	GP
Compression				■	
Restoration					
Enhancement					■
Feature Extraction	■			■	■
Pattern Classification	■			■	■

4.1.1. Current Research : An Overview.

The majority of work concerning image processing using evolutionary computation is confined to the area of image analysis. In conjunction with this, little work is focused upon evolutionary programming and evolution strategies. These trends can be observed in *Table 4.1*, and are discussed fully in section 5. A *new* member of evolutionary computation that has yet to be

discussed is genetic programming (GP), an extension of the GA. GP is a later development in which:

> "the structures that make up the population under optimisation are not fixed-length character strings that encode possible solutions, but *programs* that, when executed*, are* the candidate solutions to the problem", Poli (1996).

Programs are expressed as parse trees rather than as lines of code, with the basic GP search algorithm a classical GA with mutation and crossover specifically designed to handle parse trees. (See section 4.4). The aim of this chapter is to introduce the reader to the wide variety of work currently underway, with particular attention paid to the more *unusual* and *interesting* projects. For this reason, the fascinating work of image analysis using autonomous agents shall be discussed in section 4.2, a selection of more *orthodox* work using GAs is shown in section 4.3, and finally genetic programming is introduced in section 4.4.

4.2. Artificial Life in Image Analysis.

Table 4.1 in the previous section displayed the use of artificial life in the area of image analysis. Although this work is very rare, the individual approaches are fascinatingly innovative and could hold some clues as to the future of image analysis techniques.

4.2.1. Using Artificial Life for Feature Extraction.

Papers by Liu et al. (1997a, 1997b) introduce a new class of evolutionary autonomous agents that "evolve in a digital image environment and emerge salient image features".

> "The approach introduced here utilises evolutionary autonomous agents that can self-reproduce, diffuse, and cease to exist during the course of interacting with a digital image environment". (Liu et al. 1997a).

Evolutionary autonomous agent (EAA) systems as applied to image processing techniques is a newly explored area of research that studies emergent behaviour from the interaction of agents and the digital image according to a set of behavioural rules. The EAA approach offers a new alternative technique to image feature detection that is robust and adaptive. Conventional techniques are usually composed of grid template-like look-up tables. These have several problems; a) they are noise sensitive, b) all possibilities must be analysed and searched, and c) the complexity of image feature extraction depends upon the complexity of the image. In the proposed agent model, these problems can be somewhat overcome. Due to the reproduction and diffusion into their adjacent search space, agents can encounter local features simultaneously.

During the course of evolution, each of the agents in the lattice environment will exhibit several important behavioural responses including self-reproduction, randomised/non randomised search with varying step sizes, decay, and cease to exist. These behavioural responses can be triggered by the external stimuli present within the environment. As a result of the agent interaction, certain patterns, i.e. the *phenotypes* of the agents, will emerge, which in turn characterise the features in the digital image environment. An example of feature detection is border tracing. Here, an agent on a border will stay and reproduce asexually within a locus of specific radius, giving the offspring similar characteristics. Some offspring will wander into a border and the reproduction cycle will continue. After a specified number of generations, if a border is not detected, the agent will cease to exist. In general, features are detected by an agent through the concentrations of grey-level pixel values in the neighbouring region. If this concentration is within a certain range then the agent will reproduce. The closer an agent is to a *sensitive location*, the higher its fitness value.

Image features are evolved through mutations in spatial displacement, i.e. the location an agent is generated with respect to a parent, and randomised changes on the radii of offspring, where an evolution strategy based search process finds a set of values that best suits local conditions.

Figure 4.2. The extraction of edges from a digital image based on the proposed autonomous agent-based computation model. *The original input image is the one labelled as t = 0. The following images show the evolution of the agent population over the 2-D lattice. At t = 26, all image features (i.e. region borders in this case) are found and labelled with markers. (Taken from Liu et al. 1997a).*

(a) t = 0. (b) t = 4. (c) t = 16. (d) t= 26.

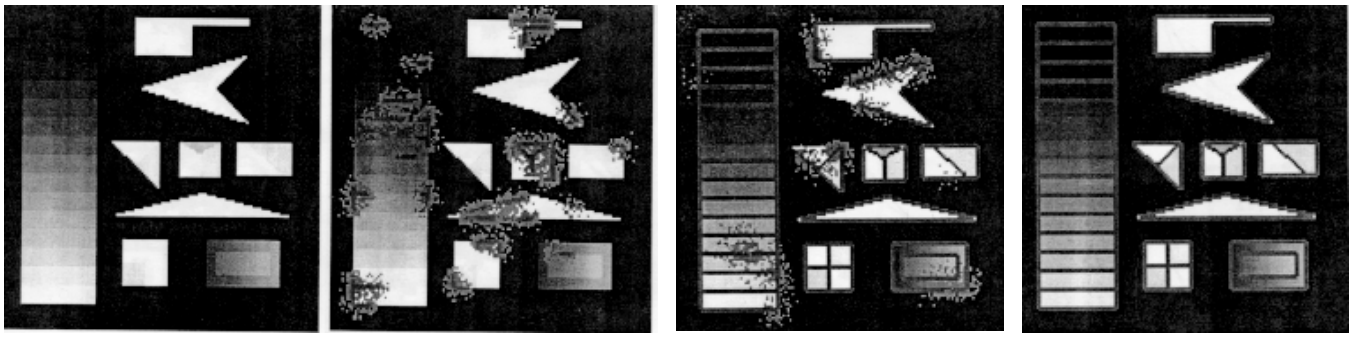


Figure 4.2 above, shows the excellent edge detection results that can be achieved in only 26 time steps. Border extraction, multiple feature extraction, and image feature tracking also display impressive results using this method.

Advantages of approach.

1) Image feature extraction process is entirely determined by the locality and parallelism of the individual agents.

2) Directions for the diffusion and self-reproduction of agents are dynamically selected and evolved.

Liu et al. (1997a) suggest

> "the proposed approach could have significant impact on difficult image processing problems, i.e., problems in which conventional edge and contrast enhancement have failed to extract important features. Examples are:
>
> * identification of pathological foci of early stage cancer and important anatomical features from ultrasound images of a prostate.
> * identification of speculated lesions, microcalcifications, and circumscribed lesions in scanning mammograms for breast cancer."

4.2.2. Character Recognition using Autonomous Agents.

In order to get to the stage of pattern recognition, a large amount of pre-processing often has to take place. The aim of Zhou and Franklin (1994) is to significantly reduce the amount of pre-processing, by almost eradicating the need for feature extraction, when performing character recognition. With standard feature extraction techniques it is hard to find a single set of features that can define the whole recognition space. A *featureless* method of pattern classification would thus be of great benefit. Zhou's and Franklin's motivation is to see if such a *featureless* method can be found by virtue of artificial life methods.

Zhou and Franklin (1994) propose a set of character recognition agents that "must find food in a predefined artificial environment". This environment consists of a 24x24 pixel matrix character image, within which an agent must learn to find food by following a path along a character. Zhou and Franklin realise that "there are two problems to be solved:

(1) If eating the food means recognition, how do agents show recognition results?
(2) How can an agent find and eat food fast. "

These problems are solved as follows:

(1) A *set* of agents is created, each trained to eat a particular character. "They begin eating simultaneously. An agent trained to eat this particular character will eat faster than the others. The character is recognised by the index of the winning agent, the fastest eater. (Zhou and Franklin 1994).
(2) An agent must be able to memorise the paths of its given character in order to be able to find food quickly. It does so by neural network learning.

Several learning strategies are proposed including *reinforcement learning* and *rule learning*. For this project, character recognition agents employ a variant of reinforcement learning called *semi-reinforcement learning*.
Since *fast eating* is the key to a successful agent, the fitness function can be evaluated as

$$\mathbf{E(i) = Nf / Nm}$$

where **i** indexes a particular agent, **Nf** is the number of food pixels eaten, and **Nm** is the number of moves made. Range of fitness value for each agent is $\mathbf{1 \geq E(i) > 0}$.

There is a question as to where in the environment the agent should begin its life. Zhou and Franklin tackle this by choosing four character agents for each character, one starting from each corner. A search problem of choosing a set of agents that produces good recognition is performed by a genetic algorithm. See Zhou and Franklin 1994 for further details of the GA used.

In order to make the evolutionary search more effective, Zhou and Franklin test a *co-evolutionary* strategy. After several generations, the most fit individuals are tested with each of the set of characters. The characters are ranked according to the error these fit individuals make. The greater the error, the higher the rank. The fittest characters are then added to the training set and the process is repeated.

Testing.

The initial prototype for testing consists of 7x4 agents meant to recognise the characters A, B, C, D, E, F, G. 7x10 characters are used as a fitness test set. The intent is to create one individual (agent-string) to describe a set of agents which can perform an adequate job of recognition. The

results for co-evolution and learning (though eventually tending to the same fitness value) reaches the maximum value in much fewer generations than using evolution and learning, proving the benefit of a co-evolutionary strategy. Results also show that testing using a majority vote of the best three individuals out performs that of using the single best individual. Zhou and Franklin (1994) state that:

> "[the] recognition rate achieved, while not high in an absolute sense, is promising in a small prototype with a small training set. On the other hand, testing is computationally intensive, since the training set increases over time and backpropagation learns slowly."

Although the tests are as yet partial, small and incomplete, results from this project are exiting. Zhou's and Franklin's next task is to build a complete system. Until then, a complete analysis of this work is infeasible although one cannot fail to notice that such an innovative technique must be beneficial to pattern recognition as a whole, even if not developed fully. To operate directly within an image, without the need for feature extraction, is definitely an area that future research should try to incorporate.

4.3. Genetic Algorithms used in Image Processing.

Unlike artificial life techniques, GAs are extensively used throughout image processing. Due to their *versatility* and *robustness*, the popular GA appears as a search space optimiser in image compression (section 4.3.1), feature extraction (section 4.3.2), and pattern recognition (section 4.3.3). In this section, the reader shall be introduced to a selection of projects, with attention paid to the improvement, if any, the GA can make upon contemporary techniques.

4.3.1. Using a GA for Fractal Image Compression.

Genetic algorithms can be used in image compression techniques to reduce the search space for self-similarities within an image. Rather than constraining a method to an exhaustive and computationally inefficient search, a genetic algorithm can use multiple search points in order to

find a near optimal solution. A heuristic search of this nature can reduce a search space by 'throwing away' many solutions whose fitness values are far away from the optimal.

Mitra et al. (1998) proposes a "new method for fractal image compression [using a] genetic algorithm with elitist model". This work uses an iterative function system (IFS).
A fully automated fractal based encoding process that approximates small image (range) blocks from larger domain blocks was first proposed by Jacquin (1992). Jacquin's "fractal block coding" obtained a set of separate transformations for each range block. When this set is iterated upon using any arbitrary starting image, an attractor approximating the target image will result. This is known as a partitioned iterative function system (PIFS). In Mitra et al. (1998);

> "a new method for image compression using PIFS is proposed. The proposed method uses a simpler classification system for range blocks. Genetic algorithms with elitist model are used in finding the appropriate domain block as well as the appropriate transformation for each range block."

The fitness function of the genetic algorithm is equal to the distance function. That is the mean square error of the original set of grey level values and the obtained set of grey values. Probability of selection is inversely proportional to fitness function. The elitist model keeps knowledge about best string so far preserved in population. This provides a track of the best strings over all iterations.

Mitra et al. (1998) use block classification to reduce encoding time and increase compression ratio. A range block is classified as 'smooth' if the variance of pixel values of the block is below a threshold, otherwise block is 'rough'. The genetic algorithm is used to find the appropriate values of rough blocks. 8x8 (parent), and 4x4 (children) range blocks are computed and corresponding thresholds are selected from respective histograms of variances.

The purpose of block classification is to store fewer bits and thus increase the compression ratio, whilst reducing the search space of the domain blocks and thus reducing the encoding time. The peak signal-to-noise ratio (PSNR) can be used to measure the quality of the image. Below is a comparison between Mitra et al.'s GA and the exhaustive search.

Table 4.3. A comparison between Mitra et al.'s GA and an exhaustive search.

	Number Of Encoding Levels	
	One	Two
Compression Ratio	Same	Better for exhaustive search
PSNR	Very close	Better for GA
Domain Blocks Searched	20 times smaller for GA	20 times smaller for GA

Table 4.3, above, highlights that an exhaustive search gives a better compression ratio than the GA for a two level encoding scheme. This is because more rough-type parent blocks are encoded correctly with an exhaustive search in the first level. These are not divided into children in the second level and so a greater compression ratio is achieved.

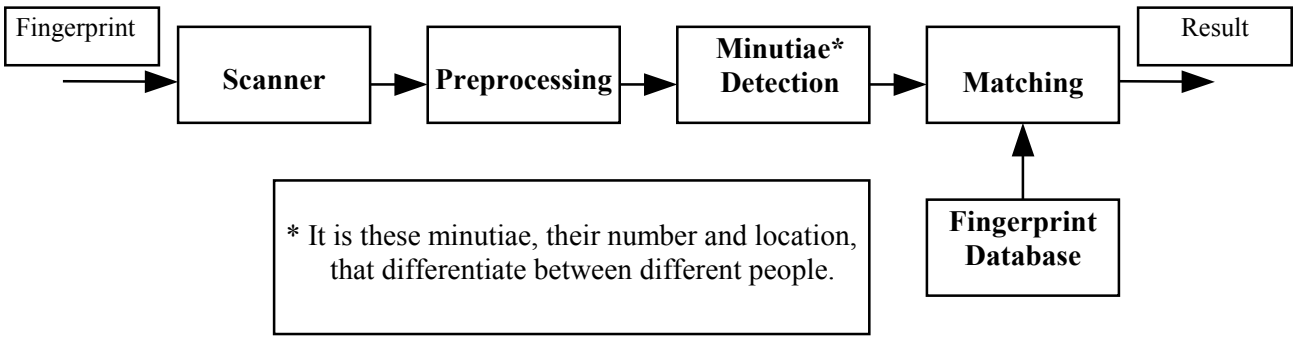
Jacquin (1992) describes a trimming of the search space that differs from Mitra et al.'s. The reduction ratios are 16 and 4 for 8x8 and 4x4 range blocks respectively, for a 256x256 image. In comparison, Mitra et al.'s (1998) GA based method has search space reduction ratios for a parent and child rough type range blocks of approximately 18 and 21 respectively. Mitra et al.'s GA based fractal scheme is also comparable to the algorithm described by Fisher et al. (1992), where the compression ratio depends upon the number of range blocks. Using a quadtree method, the compression ratio (9.97) and PSNR (31.53) reported by Fisher et al. is almost equal to that by Mitra et al.(1998), although complex images would require the exhaustive search to be very extensive. In comparison then, the GA based method is better for reducing the search space.

4.3.2. Feature Extraction using GAs.

Feature extraction methods can be improved using a GA in several different ways. This section shall concentrate upon the "estimation of ridges in fingerprints" (Abutaleb and Kamel 1999), and "automatic facial feature extraction" (Chun-Hung and Ja-Ling 1999). Abutaleb and Kamel (1999) developed a GA to find the ridges in paper fingerprints as part of an automatic fingerprint identification system. See *Figure 4.4*. Based upon the fact that the ridges in a fingerprint are

parallel, the system can be modelled as a parameter optimisation problem with values the (pixel) width of ridges in the fingerprint.

Figure 4.4. Automatic fingerprint identification system.



> "The estimation of the of the line widths is a non-linear estimation problem that is not easy to solve. In this correspondence, we propose to use an adaptive GA to solve the non-linear estimation problem, and thus obtain the ridges in the fingerprint. This approach is *guaranteed* to converge to the global minimum". (Abutaleb and Kamel 1999).

Each scanned line in the image is made up of black or white pixels. The width of each segment, and its value, black or white, are uniform parameters. These represent a gene, with each parameter independently treated by crossover and mutation, whose probabilities change with each iteration. After each iteration, the solution yielding the maximum correlation value (fitness function) has as its members the values of the probability of crossover and mutation. These values are then used in the mutation to produce new offspring. Iterations stop if the number exceeds a limit or correlation coefficient does not change.

The results for this adaptive GA technique are very promising, easily out performing the contemporary techniques of Sobel mask operators and the phase-based algorithm, see Abutaleb and Kamel 1999, particularly in image segments that contain over or under inking. Currently, Abutaleb and Kamel are investigating the problem of removing the writings from the fingerprints.

Chun-Hung and Ja-Ling (1999) attempt automatic facial feature extraction by using GAs. The approach is partitioned into two stages. Firstly, the *face region estimation* stage implements a region growing method used to estimate the face region of the target image, before a GA extracts the facial feature points within the face region in the *feature extraction* stage.

> "In the first stage feature extracting, the main features (eyes and mouth) are extracted from the face region. When the main features have been found, in the second stage feature extracting, the other features (eyebrows and nose) are also extracted based on the intrinsic information of the main features".

For details on how the GA operates see Chun-Hung and Ja-ling 1999.

Results from trials show impressive results. A comparison between applying a GA to the second stage of the algorithm and not applying a GA shows that using a GA greatly reduces the computational cost if the search range is large. Only if the search range is small does the GA reduce algorithm efficiency.

> "It is shown by computer simulations that the facial features can *always* be extracted *exactly* by applying the proposed algorithm". (Chun-Hung and Ja-Ling 1999, not original emphasis).

4.3.3. Pattern Classification using GAs.

In similar fashion to the GA, neural network architectures gained widespread enthusiasm throughout the 1980s. Nowhere is this prevalence more obvious than in the area of GAs used for pattern classification. A very popular tool for pattern recognition, the artificial neural network can benefit from the use of genetic algorithms to help to tune network parameters effectively. Although this *is* a practical use of the genetic algorithm being used for pattern recognition, the actual input of the GA as a ratio of the entire project is very small. When considering the fact that it is essentially a parameter optimisation problem, this work is not the most innovative around and so shall not be discussed in great detail in this section. Rather the reader shall merely be introduced to several varied problems to get a feel for the work currently undertaken.

Wang et al. (1997) use a GA to tune densities for fuzzy integral multiple neural networks for a hand-written digits recognition program. The fuzzy integral, using densities calculated from the

proposed genetic algorithm, outperformed that using predetermined fixed densities, and that which averages outputs.

Fillipidis et al. (1999) attempt surface land mine detection using data fusion and fuzzy reasoning. A GA performs a combinatorial search across all provided input variables and neural network configurations of a multi-layered perceptron. The *false alarm rate* for land mines was greatly reduced through the implementation of the GA.

Finally, Lakshmanan (1999) uses a GA to tune fuzzy sets to differentiate between *Bounded Weak Echo Regions* (BWER) and non-BWERs for weather forecast systems. Confidence estimations for detection can then be output directly to a forecaster, or passed to a neural network.

4.3.4. Using Alife and GAs for Surface Approximation : A Comparison.

An interesting area of research related to pattern recognition is proposed by Fujiwara and Sawai (1997, 1999) based upon a surface approximation problem. Continued from earlier work (Fujiwara and Sawai 1997) that is based upon Alife techniques only, involving a *selection-reproduction algorithm* (SR), Fujiwara and Sawai (1999) implement and compare two evolutionary algorithms applied to the approximation of a three-dimensional image of human face. The first algorithm comprises the selection and reproduction of node points in a single triangulation, whilst the second algorithm comprises a GA in which a set of different triangulations is regarded as a population. Even though there are very few studies that use EC techniques to solve this kind of problem, Fujiwara and Sawai (1999) insist that it is an effective method. The target of the project is to approximate a human facial surface by constructing a triangular mesh with a limited number of sample points. Errors are calculated by the total Euclidean ($\mathbf{R}^3$) difference in volume between the approximation and the original image. This has possible connections with face recognition problems, but with the added complexity of three dimensional facial data. For details of the EC algorithms and results see Fujiwara and Sawai (1999).

Although both algorithms give good results, there are differences in their efficiency. The SR algorithm appears to be a much better run time algorithm for time-varying surfaces, converging to a stabilised total error after much fewer generations than the GA, yet the GA tends to a lower

overall error value. As such there are benefits gained from both algorithms, making it difficult to ascertain which is the most *useful*. Obviously, one important fact that can be taken from this research is that EC in general can be a very helpful tool, not only in this specific area but in other related fields.

4.4. Genetic Programming for Image Analysis.

As briefly mentioned in section 4.1.1, genetic programming is an extension of the genetic algorithm that uses programs expressed as parse trees as a genotype, rather than binary strings.

> "GP has been applied successfully to a large number of difficult problems like automatic design, pattern recognition, robotic control, synthesis of neural networks, symbolic regression, music and picture generation", Poli (1996).

Unfortunately, only a small number of GP applications are reported in low-level image analysis. This section aims to introduce work in this area.

4.4.1. GP Applied to Low-Level Image Analysis.

Poli's approach to using GP for image analysis (Poli 1996):

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

Of these 'low-level' tasks, image segmentation is by far the most difficult problem, with standard filtering techniques usually providing insufficient results for reliable classification. Only methods considering all sources of information, such as artificial neural networks, seem to give relatively good results. It is arguable though, that the good performance of neural networks derives from the presence of certain non-linearities. An unbiased learning technique, such as GP, that is not dependant upon specific non-linearities or fixed-shape neighbourhoods should produce better results.

The fitness function for image segmentation is instrumented to overcome the sensitivity/specificity dilemma. Any segmentation or detection algorithm dealing with a real-world image has to face the fact that when detecting points of interest, some will be missed, and other uninteresting points will be detected. The optimum setting is one where *sensitivity* and *specificity* are as great as possible. A problem occurs within GP, as results often give a very sensitive algorithm with poor specificity, or vice versa. The ideal situation is one with a good trade off.

4.4.2. Experimental Results of GP in Medical Imaging.

Poli (1996) attempts to obtain filters using GP that can perform optimal segmentation and detection in complex real-world grey scale images. In order to sufficiently test the filters, the domain of medical imaging, one of the most difficult, was chosen.

Segmentation of the brain in magnetic resonance images was attempted using GP, with experiments using neural-networks also undertaken for comparison. Although the understandability of the optimal program evolved through GP, was minimal, its performance is much better than that of the neural network, and much closer to the desired results.

Detection of blood vessels in X-ray coronagrams was also attempted by Poli, with GP again easily outperforming neural-networks. Poil (1996) states:

> "The research on GP for image analysis is hampered by the tremendous demand on computational resources involved in fitness evaluation... However, the impressive performances shown (after learning) by GP compared with NNs [neural-networks] in the experiments reported [here] seem to suggest that there is a huge space of image processing tools much more powerful than those used in image processing nowadays and that GP can be used to start exploring this space".

As the results from genetic programming are truly outstanding, putting research efforts into this area could be a very good investment for the future.

4.5. Chapter Summary.

Chapter four has introduced a selection of projects using evolutionary computation for image processing. The aim has been to give an oversight into the current field, concentrating upon the more 'interesting' projects. Although this has not been a comprehensive survey of the field, the collection of projects chosen are diverse, covering most areas. The results from all are very promising.

Other related work that is not strictly image processing is currently being undertaken in Sussex university. Husbands et al. (1998) and Jakobi et al. (1998) have been evolving GASnets using GAs for robot vision. Their work is showing exiting results and is recommended as a reference to any reader attempting research into evolutionary computation used in robot vision.

Although it is currently a small field, using EC in image processing, as shown in this chapter, can be very effective. Surely this suggests that interest and thus research, should and hopefully will, grow in the future. Presently most work is in the very early stages of development, and so it is difficult to strictly determine how they compare to contemporary techniques, yet it can clearly be seen that even at this 'early' stage a lot is being achieved.

5. Trends and Developments.

As briefly discussed in section 4.1, and displayed in *Table 4.1*, there are several trends developing in the field of evolutionary computation used in image processing. In short, there appears to be a strong attraction towards 'favoured' EC methods, with others being almost completely ignored. (See section 5.1). In conjunction with this, the majority of work is confined to the area of image analysis, (Section 5.2). This chapter aims to discuss these trends in full, speculating upon the possible causes of these developments, and how they may affect the future of image processing.

5.1. Favouritism Amongst Evolutionary Computation Techniques.

When researching the use of EC techniques in image processing, it is strikingly obvious (*Table 4.1*) that evolution strategies and evolutionary programming are largely ignored. Developed in Berlin in the early 1960s, evolution strategies had a very different beginning to the other EC techniques. Very much an 'outsider' in the world of EC, evolution strategies had a very independent evolution, increasingly falling out of favour as time progressed. This resulted in the near-extinction of ES observed today. Though it *is* worth recognising the existence of ES, it is very doubtful that there is a significant amount of work utilising ES today. Realistically, ES can almost be discounted when discussing present EC technology.

Despite a very different development, the similarity of evolutionary programming to evolution strategies has led both to a similar fate. Throughout current research, it is rare to see 'pure' EP that has not been subject to hybridisation. Perhaps the reason for falling so 'out of favour' was, like ES, EPs initial 'problem specific' development.

Shortly after the introduction of the GA, equipped with a firm theoretical framework laid down by Holland, it became evident that this non problem-specific tool was much more versatile than other evolutionary algorithms. Rather than endure complex initialising of algorithms for each new problem, the GA has the 'freedom' to allow one algorithm to be used for many different

applications with the same operators. Without being constrained to one set of problems, the GA offers great portability and ease of use. This caused a great increase in popularity. In the 1980s particularly, the meteoric rise in use of the GA led to a situation where everybody wanted to utilise this 'fashionable' tool. In turn, the demise of the other EAs exacerbated the situation. Presently, nearly all EAs used are GAs, or a GA hybrid.

It can be observed that, albeit rarely, Alife techniques are in use throughout image processing. This is probably due to the fact that Alife directly interacts within an image. This novel approach to image processing, given some success, could possibly change the direction of the whole field. It is thus unsurprising that projects have been attempted, although this 'unorthodox' approach is always unlikely to gather a huge following until solid evidence supporting improvements have been produced.

5.2. The Domination of Image Analysis.

Table 4.1 shows that there is very little work using EC outside of image analysis. In particular there is no image restoration work at all using EC. Restoration is usually a specific process, dependent upon the degradation within the image. In order to successfully restore, *a priori* knowledge of the type of degradation is needed. The problem of image restoration is thus one of finding the correct pattern of degradation. Therefore, it can be assumed that using a search space optimiser such as evolutionary computation would be unnecessary and unhelpful. Not just presently, but for the foreseeable future, image restoration is unlikely to utilise EC whilst there is no real search space to optimise.

In contrast to image restoration, image analysis is very heavily dependent upon searching for optimum solutions. Feature extraction has to search pixel-spaces in order to identify attributes as optimally as possible, whilst pattern classification has to search object-spaces in order to successfully classify a pattern. It is thus unsurprising that work involving evolutionary computation is focused in this area.

Despite the fact that there is little evidence to support wide use of evolutionary computation in image compression, Section 4.3.1 shows how EC can be utilised in this area. The promising results gained from using a GA for fractal image compression rely on the specific method used,

where a search space results from finding optimal transforms for range blocks. Small projects are also underway in the area of image enhancement, using genetic programming to find optimal filters which, when applied to an image, transforms it into another image with the desired characteristics. See Poli 1996.

5.3. Possible Developments.

After discussing the trends appearing within image processing using evolutionary computation, there appears to be a set of probable developments for the foreseeable future. These are shown overleaf:

- Due to the great momentum of research surrounding GAs, it seems reasonable to assume that this will be a continued pattern for the foreseeable future. The GA, and hybrid tools such as genetic programming, already have a strong following of research, producing excellent results across a wide range of image processing techniques. For this reason there appears no logical reason why this trend would change. It may not be long before the genetic algorithm is accredited as being a fundamental image processing tool, appearing on computer imaging syllabuses throughout the undergraduate world.

- The impressive, though preliminary, results of Alife used for image analysis leads to the possibility of a surge in enthusiasm in this area. Successful pattern recognition without the need for feature extraction is so desirable that there must be more research completed in this area in the near future. Artificial life is unlikely to become as prevalent as genetic algorithms, but there is so much scope available for development of image analysis techniques that utilise artificial life, that surely research in this area will become more common place.

- In all likelihood, the evolutionary computation techniques of evolution strategies and evolutionary programming will be abandoned in image processing. The only time that these algorithms may appear would be as a selection mechanism for artificial life agents, even then this would be rare.

- Unless there is a change in the general approach to image compression, restoration, and enhancement, these sections of image processing are likely to benefit little from evolutionary computation. In the case of image restoration, work will probably be negligible. Although there probably is room for developing techniques in image enhancement and image compression, these will unfortunately fail to be exploited.

- No matter what the outcome of developments in this field, the twenty-first century should be very exciting, as we see evolutionary computation become more common place throughout the whole of computer science.

5.4. Chapter Summary.

Chapter five has highlighted the general trends appearing in the field of image processing techniques that utilise evolutionary computation. These include the widespread use of genetic algorithms and hybrid tools such as genetic programming, the innovative and promising research into artificial life techniques, and the fact that the bulk of work is concentrated throughout image analysis. The redundancy of evolution strategies and evolutionary programming has been highlighted, in conjunction with image restoration's seemingly incompatibility with evolutionary computation in general. Possible developments in the future of the field were also suggested.

6.Conclusion.

Image processing is a subject that could benefit immensely from techniques reducing computation time and output accuracy. Despite the fact that there are many valuable techniques available to a researcher in image processing, few achieve the level of success required for complex problems to be solved effectively. Undoubtedly, there is great room for improvement throughout the field of image processing.

Evolutionary computation, created to tackle search optimisation problems, has been shown to succeed in improving the performance of image processing systems throughout a range of problems. Image analysis, in particular, can benefit from applying artificial life, genetic algorithms, and genetic programming to a variety of tasks. It is not unreasonable to suggest, therefore, that evolutionary computation, in general, can improve the performance of image processing operations.

The field of image processing using evolutionary computation is currently very immature. Unfortunately, this hinders the evaluation of proposed models, due to the lack of test results. Currently there appears to be no standard method of evaluation, resulting in a lack of hard evidence to support arguments for or against proposed algorithms. This makes it difficult to conclude with certainty whether or not evolutionary computation can benefit the world of image processing.

It should be noticed, however, that throughout the duration of this project, lots of new literature has appeared, not only upon the subject of evolutionary computation in general, but also EC used in image processing. This displays the fact that the field is growing at a significant rate. The implication of this growth must be the underlying success of the methods discussed.

In conclusion, therefore, I suggest that image processing not only *can benefit* from evolutionary computation, but *will increasingly continue to benefit* for the foreseeable future.

APPENDIX A. Evaluation.

Once I had overcome the initial apprehension of undertaking such a large assignment in an unknown area, working on this project became a pleasurable experience. Throughout my life, I have always had a fascination for evolutionary computation, despite my scant knowledge. This project has enabled me to vastly increase my understanding of evolutionary computation, and how it can be applied. I also have a been left with a great sense of achievement from the fact that I have, to a great extent, taught myself through research.

Considering the general demands of a final undergraduate year in joint honours, I feel that I have worked to the best of my ability. In similar fashion to most things, though, I have yet again realised that no mater how early one starts a task, there is always an inevitable rush to finish at the end. It is for this reason that the project may not quite be at the standard I would have hoped.

Retrospectively, I believe that I would perhaps have benefited from researching a field that was more 'documented'. Previously unfamiliar with research techniques, it took me several months to collate a sufficient amount of information concerning research using evolutionary computation in image processing. Many hours where wasted whilst vainly searching in the wrong place. Had I been researching a field with more easily available information, I believe that I may have been able to make more productive use of my time during the early months of the project.

Overcoming my initial vexations and frustrations, I soon found myself engaged in a piece of work on a scale far greater than I had ever attempted previously. In conjunction with learning about the content of the project, I also found the process of undertaking such a large task a learning experience, both in the domains of time management, and self discipline.

On the whole, I am extremely pleased with the progress made during the construction of this project. Apart from several minor changes, if I was to repeat the project, the only change that I would make would be to put a lot more time in during the early stages. That said, I sincerely hope that the reader has had as much pleasure in reading the project as I had in writing it. The world of artificial intelligence has never been so exciting.

APPENDIX B. Evolution Strategy Syntax.

Below is a list of characterisation terms for evolution strategy selection.

- **(1+1) ES.** A two-membered evolution strategy. The offspring is created by applying mutation with identical standard deviations to each object variable. The two individuals are compared, with the best surviving to become parent of the next generation, whilst the other is discarded.

- **(μ+1) ES.** Incorporates the idea of a population. **μ** parent individuals recombine to produce one offspring, which after being mutated eventually replaces the worst parent individual.

- **(μ+λ) ES.** Best **μ** individuals out of the union of parents and offspring survive.

- **(μ,λ) ES.** Only the best **μ** offspring individuals form the next parent generation, (consequently, **λ>μ** is necessary).

Both **(μ+λ) ES,** and **(μ,λ) ES** can be interpreted as instances of the general **(μ, K, λ)** strategy, where **∞≥ K ≥1** denotes maximum life span (in generations) of an individual. For **K=1**, selection method yields **(μ,λ)** strategy, while it turns into the **(μ+λ)** strategy for **K=∞**.

References.

Abutaleb, A.S., and Kamel, M. (1999), *A Genetic Algorithm for the Estimation of Ridges in Fingerprints.* IEEE Transactions on Image Processing, Vol. 8, No. 8, August 1999, pages 1134-1139.

Atmar, J.W. (1976), *Speculation on the Evolution of Intelligence and Its Possible Realization in Machine Form.* Sc. D. Diss., New Mexico State University, Las Cruces.

Awcock, G.J. and Thomas,R. (1995), *Applied Image Processing.* Basingstoke: Macmillan.

Bach, T. (1996), *Evolutionary Algorithms in Theory and Practice: evolution strategies, evolutionary programming, genetic algorithms.* New York Oxford: Oxford University Press.

Bhandarkar, S.M., and Zhang, H. (1999), *Image Segmentation Using Evolutionary Computation.* IEEE Transactions on Evolutionary Computation, Vol. 3, NO. 1, April 1999, pages 1-21.

Brooks, R.A., and Maes, P., eds. (1994), *Artificial Life IV. Proceedings of the Fourth International Workshop on the synthesis and Simulation of Living Systems.* Cambridge, Mass.: MIT Press.

Chun-Hung, L., and Ja-Ling, W. (1999), *Automatic Facial Feature Extraction by Genetic Algorithms.* IEEE Transactions on Image Processing, Vol. 8, No. 6, June 1999, pages 834-845.

De Jong, K.A. (1985), *Genetic algorithms: A 10 year perspective.* In Grefenstette (1985), pages 169-177.

De Jong, K. (1999), *Evolutionary Computation: Recent Developments and Open Issues.* In Miettenan et al. (1999), pages 43-52.

Emmeche, C. (1994), *The garden in the Machine: the emerging science of artificial life.* Translated by Steven Sampson. Princeton, N.J.: Princeton university press.

Filippidis, A., Jain, L.C., and Martin, N.M. (1999), *Using Genetic Algorithms and Neural Networks for Surface Land Mine Detection.* IEEE Transactions on Signal Processing, Vol. 47, No. 1, KJAnuary 1999, pages 176-186.

Fisher, Y., Jacbos, E.W., and Boss, R.D. (1992), *Fractal image compression using iterated transforms*. In Storer (1992), pages 36-61.

Fogarty, T.C., ed. (1996), *Evolutionary Computing AISB Workshop. Brighton, U.K. April 1-2 1996.* Berlin London: Springer.

Fogel, D.B., and Atmar, W. (1990), *Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems.* Biological Cybernetics, 63: pages 111-114, 1990.

Fogel, D.B. (1999), An Introduction to Evolutionary Computation and some Applications. In Miettinen et al. (1999),
pages 23-41.

Fogel, D.B. (2000), *Evolutionary Computation: toward a new philosophy of machine intelligence.* New York: IEEE Press, 2nd ed.

Fogel, L.J., Owens, A.J., and Walsh, M.J. (1966), *Artificial intelligence through simulated evolution.* New York : Wiley.

Fujiwara, Y., and Sawai, H. (1997), *Evolutionary Computation Applied to 3D Image Reconstruction.* Proc. 1997, IEEE International Conference on Evolutionary Computation 1997, pages 465-469.

Fujiwara, Y., and Sawai, H. (1999), *Evolutionary Computation Applied to Mesh Optimization of a 3-D Facial Image.* IEEE Transactions on Evolutionary Computation, Vol. 3, No. 2, July 1999, pages 113-123.

Goldberg, D.E (1989), *Genetic Algorithms in Search, optimization and Machine Learning.* Reading, Mass. Wokingham: Addison-Wesley.

Gould, S.J. (1991) *Bully for brontosaurus.* W.W. Norton.

Grefenstette, J.J., ed. (1985), *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Haataja, J. (1999), *Using Genetic Algorithms for Optimization: technology transfer in action.* In Miettinen et al. (1999),
pages 3-22.

Hao, J.K., Lutton, E., Ronald, E., Schoenauer, M., and Snyers, D., eds. (1998), *Artificial evolution: third European conference, AE '97, Nimes, France, October 22-24, 1997: selected papers.* Berlin New York: Springer.

Holland, J. (1975), *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Husbands, P., Smith, T., Jakobi, N., and O'Shea, M. (1998), *Better Living Through Chemistry: Evolving GasNets for Robot Control*.
http://www.cogs.susx.ac.uk/users/toms/GasNets/CONNSCI98/index.html

Jacquin, A.E. (1992), *Image coding based on a fractal theory of iterated contractive image transformations*. IEEE Transactions on Image Processing, Vol. 1, No. 1, pages 18-30.

Jakobi, N., Husbands, P., and Smith, T. (1998), *Robot Space Exploration by Trial and Error.* http://www.cogs.susx.ac.uk/users/toms/GasNets/GP98/index.html

Lakshmanan, V. (1999), *Using a Genetic Algorithm to Tune a Bounded Weak Echo Region Detection Algorithm.* http://www.nssl.noaa.gov/~lakshman/Papers/ga/node9.html

Langton, C.G., Taylor.C., Farmer, J.D., and Rasmussen, eds. (1991), *Artificial Life II: the proceedings of an interdisciplinary workshop on the synthesis and simulation of living systems held in 1990 in Los Alamos, New Mexico.* Redwood City, Calif.: Addison-Wesley.

Langton, C.G., ed. (1994), *Artificial Life III: proceedings of the Workshop on Artificial Life, held June 1992 in Santa Fe, New Mexico.* Reading, Mass.: Addison-Wesley, Advanced Book program.

Langton, C.G., ed. (1997), *Artificial Life: an overview.* Cambridge, Mass. London: MIT.

Levy, S. (1992), *Artificial Life: the quest for a new creation.* London: J.Cape.

Liu, J., Tang, Y.Y., and Cao, T.C. (1997a), *An Evolutionary Autonomous Agents Approach to Image Feature Extraction.* IEEE Transactions on Evolutionary Computation, Vol. 1, No. 2, July 1997, pages 141-158.

Liu. J., Maluf, D.A., and Tang, Y.Y. (1997b), *Toward Evolutionary Autonomous Agents for Computational Perception.* 1997 Online, IEEE Invited Paper, pages 3096-3101.

Miettinen, K., Makela, M.M., Neittaanmaki, P., and Periaux, J. (1999), *Evolutionary Algorithms in Engineering and Computer Science: recent advances in genetic algorithms, evolution strategies, evolutionary programming, genetic programming, and industrial applications.* Chichester: Wiley.

Mitchell, M. (1996), *An Introduction to Genetic Algorithms.* MIT Press/Bradford Books, Cambridge.

Mitra, S.K., Murthy, C.A., and Kundu, M.K. (1998), *Techniques for Fractal Image Compression Using Genetic Algorithm.* IEEE Transactions on Image Processing, Vol. 7, No. 4, April 1998, pages 586-593.

Pearson, D. (1991), *Image Processing*. London: McGraw-Hill.

Poli, R. (1996), *Genetic Programming for Feature Detection and Image Segmentation.* In Fogarty (1996), pages 110-125.

Ray, T.S. (1991), *An Approach to the Synthesis of Life.* In Langton et al. (1991), pages 371-408.

Russ, John C. (1998), *The Image Processing handbook.* Boca Raton, FL: CRC Press, 3rd Edition.

Sankar, K.P., and Wang, P.P., eds. (1996), *Genetic Algorithms for Pattern Recognition.* Boca Raton London: CRC Press.

Sonka, M., Hlavac, V. and Boyle, R. (1993), *Image Processing, Analysis, and Machine Vision.* London: Chapman and hall.

Sonka, M., Hlavac, V. and Boyle, R. (1999), *Image Processing, Analysis, and Machine Vision.* Pacific Grove, CA: PWS Pub, 2nd Edition.

Storer, J.A.., ed. (1992), *Image and Text Compression.* Boston, MA: Kluwer.

Umbaugh, S.E. (1998), *Computer Vision and Image Processing: a practical approach using CVIP tools.* Eaglewood Cliffs, N.J.: Prentice Hall PTR.

Voigt, H.M., Ebeling, W., Rechenberg, I., and Schewfel, H.P., eds. (1996), *Parallel Problem Solving from Nature IV.* Berlin: Springer.

Vose, M.D. (1998), *The Simple Genetic Algorithm: Foundations and Theory.* Cambridge, Mass.: MIT Press.

Wang, D., Wang, X., and Keller, J.M. (1997), *Determining Fuzzy Integral Densities Using A Genetic Algorithm for Pattern Recognition*. 1997 Annual Meeting of the North American, 1997, pages 263-267.

Zhou, L., and Franklin, S. (1994), *Character Recognition Agents.* In Brooks and Maes (1994), pages 301-306.