GPIS: Genetic Programming based Image Segmentation with Applications to Biomedical Object Detection

Tarundeep Singh Dhot

A Thesis in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Masters of Applied Science (Electrical and Computer Engineering) at Concordia University

Montreal, Quebec, Canada

January 2009

Your file Votre référence
ISBN: 978-0-494-63236-9
Our file Notre référence
ISBN: 978-0-494-63236-9

ABSTRACT

GPIS: Genetic Programming based Image Segmentation with Applications to Biomedical Object Detection

Tarundeep Singh Dhot

Image segmentation plays a critical role in many image analysis applications. However, it is ill-defined in nature and remains one of the most intractable problems in image processing. In this thesis, we propose a genetic programming based algorithm for image segmentation (GPIS). Typically, genetic programming is a Darwinian-evolution inspired program discovery method and in the past it has been successfully used as an automatic programming tool. We make use of this property of GP to evolve efficient and accurate image segmentation programs from a pool of basic image analysis operators. In addition, we provide no *a priori* information about that nature of the images to the GP.

The algorithm was tested on two separate medical image databases and results show the proposed GP's ability to adapt and produce short and accurate segmentation algorithms, irrespective of the database in use. We compared our results with a popular GA based image segmentation/classification system, GENIE Pro. We found that our proposed algorithm produced accurate image segmentations performed consistently on both databases and could possibly be extended to other image databases as a general-purpose image segmentation tool.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the unstinted support of the following people. I will always remain ever so grateful for your contributions.

I would firstly like to thank Dr Nawwaf Kharma, for his invaluable guidance and support has been the cornerstone throughout the course of this thesis. Words cannot match what he has meant not only towards this thesis, but also as a thoughtful mentor to me. In addition, I would also like to thank Ms Aida Abu-Baker and Ms Janet Laganiere from CHUM Research Centre, Notre-Dame Hospital for helping us to prepare our first cell database. They were ever so encouraging and welcoming and provided me with the best start for my research work.

I would also like to thank Mr Mohammad Daoud from UWO, London for providing us with our second image database. He has been keen supporter of this work and his timely help with technical issues is deeply appreciated.

I would also like to especially thank my good friends Mohammad Farooq Shaikh, Aman Sandhu, Rajeev Yadav and François Mollicone for their unyielding support.

I would like to convey a special thanks to Ms Marlene Gross for always being there for me and being a constant source of inspiration and motivation to me.

Finally, I would like to thank my family and my ten-month old niece Sia, for if it was not for them, I could have never reached so far.

To my parents

~ In a world of strife you offered the world to me ~

TABLE OF CONTENTS

Table of Figures ix

Table of Tables xii

Acronyms xiv

Chapter 1: Introduction 1

1.1 Motivation 3
1.2 Problem Statement 6
1.3 Objectives 7
1.4 Thesis Contributions 9
1.5 A Reader's Guide to the Thesis 10

Chapter 2: Overview of Relevant Concepts 12

2.1 Image Segmentation 13
2.2 Evolutionary Computation 15
2.3 Genetic Programming 18
2.4 GENIE Pro 20

Chapter 3: Related Work 22

3.1 Introduction 23
3.2 GP based approaches for Image Segmentation 24
3.3 EC based approaches for Medical Image Segmentation 34
3.4 A word about Cell Segmentation 37

Chapter 4: The Proposed Algorithm: GPIS 38

PART A: THE APPROACH 39

4.1 Overview of the Approach 39
4.2 Software Architectural Overview 41
4.2.1 Training Window Operation 43

4.3 Design Overview 45
4.3.1 Flowchart - Operational Working of GPIS 45
4.3.2 Representation 48
4.3.2.1 Genes 48
4.3.2.2 Chromosomes 49
4.3.2.3 Operator Pool 50
4.3.3 Initialization 50
4.3.4 Fitness Evaluation 52
4.3.5 Selection and Elitism 55
4.3.6 Diversification - Genetic Operators 57
4.3.6.1 Crossover 57
4.3.6.2 Mutation 58
4.3.7 Injection 61
4.3.8 Termination 61
4.4 Summary 62

PART B: EXPERIMENTAL SETUP 64

4.5 Image Databases 64
4.6 Procedure for Training and Validation 65
4.6.1 Procedure for Obtaining Results for GPIS. 65
4.6.2 Procedure for Obtaining Results for GENIE Pro 66
4.7 Parameter Settings for GPIS 67

Chapter 5: Results and Analysis 69

5.1 Results for Database 1 - HeLa Cells 71
5.1.1 Effectiveness 71
5.1.2 Efficiency 73
5.1.3 Intuitive Reflection: Evolved Programs 74
5.2 Results for Database 2 - Liver Tissue Specimen 80
5.2.1 Effectiveness 80
5.2.2 Efficiency 82
5.2.3 Intuitive Reflection: Evolved Programs 83
5.3 Efficiency: Execution Time Vs Length of Program 91
5.4 Reflective Analysis of Results 93
5.4.1 Analysis of Results for Database 1 94
5.4.2 Analysis of Results for Database 2 95

Chapter 6: Conclusion and Future Work 97

6.1 Conclusions 98

6.2 Future Works 98
6.3 Preliminary Testing of GPIS on Other Images 100

List of References 102

TABLE OF FIGURES

FIGURE 1. TYPICAL LAYOUT FOR IMAGE ANALYSIS 13

FIGURE 2. BASIC CYCLE OF EVOLUTIONARY ALGORITHMS 16

FIGURE 3. AN OVERVIEW OF THE GP APPROACH 40

FIGURE 4. SOFTWARE ARCHITECTURE OF GPIS 42

FIGURE 5. TRAINING WINDOW OPERATION 44

FIGURE 6. FLOWCHART OF THE GP 47

FIGURE 7. SEGMENTATION AS A PIXEL CLASSIFICATION PROBLEM 55

FIGURE 8. CROSSOVER SCHEME: ONE-POINT CROSSOVER 58

FIGURE 9. MUTATION SCHEMES 60

FIGURE 10. SAMPLES FROM IMAGE DATABASES 64

FIGURE 11. CHROMOSOMAL AND GENE STRUCTURE OF PROGRAM 1 74
(DATABASE 1)

FIGURE 12. SUPERIMPOSED INPUT-EVOLVED IMAGE (EVOLVED 75
PROGRAM 1 - DATABASE 1)

FIGURE 13. STEP-BY-STEP IMAGE EVOLUTION OF EVOLVED PROGRAM 1 76
(DATABASE 1)

FIGURE 14. CHROMOSOMAL AND GENE STRUCTURE OF PROGRAM 2 77
(DATABASE 1)

FIGURE 15. SUPERIMPOSED INPUT-EVOLVED IMAGE (EVOLVED 78
PROGRAM 2 - DATABASE 1)

FIGURE 16. STEP-BY-STEP IMAGE EVOLUTION OF EVOLVED PROGRAM 2 - ... 79
VALIDATION IMAGE 2 (DATABASE 1)

FIGURE 17. CHROMOSOMAL AND GENE STRUCTURE OF PROGRAM 1 83
(DATABASE 2)

FIGURE 18. STEP-BY-STEP IMAGE EVOLUTION OF EVOLVED PROGRAM 1 85
(DATABASE 2)

FIGURE 19. SUPERIMPOSED INPUT-EVOLVED IMAGE (EVOLVED 86
PROGRAM 1 –DATABASE 2)

FIGURE 20. CHROMOSOMAL AND GENE STRUCTURE OF PROGRAM 2 88
(DATABASE 2)

FIGURE 21. STEP-BY-STEP IMAGE EVOLUTION OF EVOLVED PROGRAM 2 - ... 88
VALIDATION IMAGE 2 (DATABASE 2)

FIGURE 22. SUPERIMPOSED INPUT-EVOLVED IMAGE (EVOLVED 90
PROGRAM 2 - DATABASE 2)

FIGURE 23. GRAPH 1 - EXECUTION TIME Vs LENGTH OF PROGRAM 92
(FITTEST PROGRAMS)

FIGURE 24. GRAPH 2 - EXECUTION TIME Vs LENGTH OF PROGRAM 92
(PROGRAMS WITH FITNESS > 70%)

FIGURE 25. PRELIMINARY RESULTS FOR: (a) LANE DETECTION 100
(b) TREE DETECTION

FIGURE 26. PRELIMINARY RESULTS FOR EXTRACTING INTRA-CELLULAR ... 101
CONTENT OF WRIGHT STAINED WHITE BLOOD CELL IMAGES

TABLE OF TABLES

TABLE 1. GP BASED APPROACHES FOR IMAGE SEGMENTATION 31

TABLE 2. EC BASED APPROACHES FOR MEDICAL IMAGE SEGMENTATION .. 36

TABLE 3. LIST OF PRIMITIVE IMAGE ANALYSIS OPERATORS 51

TABLE 4. LIST OF MORPHOLOGICAL STRUCTURING ELEMENTS 51

TABLE 5. SUMMARY OF GPIS APPROACH 63

TABLE 6. FINAL VALUES OF G, T, V, R FOR GPIS 65

TABLE 7. SYSTEM PARAMETERS SETTINGS FOR DATABASE 1 67

TABLE 8. SYSTEM PARAMETERS SETTINGS FOR DATABASE 2 68

TABLE 9. RECOMMENDED PARAMETER VALUES FOR GPIS 68

TABLE 10. SEGMENTATION ACCURACY: GPIS Vs GENIE PRO 71
(DATABASE 1)

TABLE 11. STATISTICAL RESULTS FOR GPIS: SEGMENTATION ACCURACY ... 72
(DATABASE 1)

TABLE 12. CELL COUNT: GPIS Vs GENIE PRO (DATABASE 1) 72

TABLE 13. NUMBER OF GENERATIONS AND FITNESS EVALUATIONS 73

FOR GPIS (DATABASE 1)

TABLE 14. STATISTICAL RESULTS FOR GPIS: NUMBER OF GENERATIONS ... 73
AND FITNESS EVALUATIONS (DATABASE 1)

TABLE 15. SEGMENTATION ACCURACY: GPIS Vs GENIE PRO 80
(DATABASE 2)

TABLE 16. STATISTICAL RESULTS FOR GPIS: SEGMENTATION ACCURACY ... 81
(DATABASE 2)

TABLE 17. CELL (NUCLEI) COUNT: GPIS Vs GENIE PRO (DATABASE 2) 82

TABLE 18. NUMBER OF GENERATIONS AND FITNESS EVALUATIONS 82
FOR GPIS (DATABASE 2)

TABLE 19. STATISTICAL RESULTS FOR GPIS: NUMBER OF GENERATIONS .. 83
GENERATIONS AND FITNESS EVALUATIONS (DATABASE 2)

Acronyms

AVIRIS	Airborne Visible/Infrared Imaging Spectrometer
CT	Computed Tomography
DOQ	Digital Orthophoto Quadrangle
EA	Evolutionary Algorithms
EC	Evolutionary Computing
EP	Evolutionary Programming
ES	Evolutionary Strategy
fMRI	Functional Magnetic Resonance Imaging
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
GA	Genetic Algorithms
GENIE Pro	Genetic Imagery Exploitation Pro
GP	Genetic Programming
GPIS	Genetic Programming based Image Segmentation
GT	Ground Truth
IA	Image Analysis
IRLS	Infrared Linescan
IS	Image Segmentation

MEG	Magnetoencephalography
MRI	Magnetic Resonance Imaging
MODIS	Moderate Resolution Imaging Spectroradiometer
MTI	Multispectral Thermal Imager
OP	Operator
SAR	Synthetic-aperture Radar
SE	Structuring Element
SPOT	Satellite Pour l'Observation de la Terre
TN	True Negative
TP	True Positive

CHAPTER 1: INTRODUCTION

It is not the strongest of the species that survives,

nor the most intelligent that survives.

It is the one that is the most adaptable to change.

~ Charles Darwin ~

Darwinian evolution is one of nature's most unique and significant phenomena. Yet as simple and random at times it might seem, it is an incredibly powerful and functional phenomenon. The level of adaptability, depth of detail and intricate complexity observed in nature is a mere reflection of the power of evolution. It is therefore no surprise that these principles have been applied to solve complex engineering problems. In computer science, evolution based problem solving approaches are collectively termed as *Evolutionary Computation* (EC) or *Genetic and Evolutionary Computation* (GEC). These EC approaches are typically used for combinatorial optimization problems such as the travelling salesman problem, the knapsack problem, etc. But the general applicability of these approaches makes it possible to use them in a wide range of real-world applications also. Of the many application domains that EC has been used in, recently keen interest has been seen in applying these techniques to the field of computer vision and image analysis.

One of the most challenging sub-fields of computer vision and image analysis is image segmentation. Image segmentation is a fundamental part of image analysis and is extensively used in a variety of image processing, video processing and computer vision applications. Its sole purpose is to reduce an image into useful information by identifying and isolating the objects of interest in the image. It is an essential first step in the imaging process, as the accuracy of any subsequent imaging task is dependent on the quality of the segmentation process [52]. It is thus an integral part of any expert imaging and automatic object detection system.

However, image segmentation can be a deceptively difficult problem. Traditionally, the task of segmentation (and further classification) is assigned to trained human experts. While there is no substitute for trained experts, relying entirely on human expertise can be labor intensive, time consuming and simply not practical for a large scale deployment. In recent years, there has been a tremendous improvement in image acquisition techniques which has led to the availability of high quality images in greater numbers. With larger computational capacities at hand, it's highly desirable to develop automated segmentation methods which require minimal human supervision.

This thesis proposes a new image segmentation algorithm; Genetic Programming based Image Segmentation or *GPIS*. It uses Darwinian evolution to evolve complete, efficient and accurate image segmentation algorithms for medical images.

1.1 MOTIVATION

The tenets of present day medical imaging can be traced back to late 1895 and the discovery of X-rays by Wilhelm Conrad Röntgen. This unintended scientific discovery drastically changed the face of medicine as it gave doctors the chance to decisively diagnose internal medical conditions without having to surgically operate on their patients. Today, X-rays are one of the many medical imaging modalities available to the medical profession. The challenge now is to effectively analyze the vast quantities of data produced by these imaging modalities with minimal human intervention.

One of the most important application domains of image segmentation is medical imaging. Medical imaging has been an invaluable tool to doctors and physicians. It not only simplifies the examination process but also assists in speeding up disease diagnosis and treatment planning. The role of medical image segmentation is crucial due to its inherent presence in the tools that aid the above.

Medical image segmentation can be defined as the extraction of known anatomical or cellular structures from the acquired medical images. The amount of medical data requiring expert analysis is often too great for a physician to handle. In addition, a significant amount of variance is possible among manual segmentations from different sources. This further increase the risks related to intra and inter-observer reliability. A quality medical imaging system requires segmentations to be accurate, robust, efficient and reproducible. The diagnostic data possible from non-invasive techniques like Computed Tomography (CT), Magnetic Resonance Imaging

(MRI), digital mammography, ultrasound and pathological imaging modalities can offer deep understanding of normal and diseased anatomy of a subject [34] and is critical to medical research. Thus, there is a need to build automated segmentation systems for medical imaging.

So far, various methods for automated image analysis have been developed to process the acquired images from the aforementioned imaging modalities. Though these approaches can be categorized in numerous ways [13, 32, 34], evolutionary based methods offer a lot of promise. One such evolutionary technique is genetic programming.

Genetic programming (GP) comes from a class of biologically inspired artificial intelligence algorithms that mimic Darwinian-evolution principles of natural selection and recombination. GP uses these principles to evolve a population of programs that are themselves effective solutions to a specific problem. These computer programs are evaluated based on their effectiveness in solving a given problem. An extremely desirable feature of GP is the operational nature of the solutions, typically expressed as executable computer programs [1]. This readily executable format of evolved solutions makes GP quite suited for an automatic image segmentation system.

Why use GP for segmentation?

At this point, it is quite clear that medical image segmentation is a crucial process and the methods used to solve it can greatly affect the final outcome. In the past, evolutionary approaches, GP in particular, have proved to be good problem-solving tools and could be applied to image segmentation also. The following reflect why we chose an EC based (in particular GP) approach for image segmentation.

1. From a strictly image processing point of view, every image segmentation and feature extraction program can be broken down to a set of basic low-level image processing operations, or *primitive* operations. Due to the high dimensionality and large cardinality of the search space of all possible solutions, it is simply not possible to cover every effective combination of primitive operators. A human expert is limited to exploring certain operator combinations which are often guided by conventional wisdom. GP on the other hand, is not biased or limited to exploring only certain operator combinations. It often evolves very effective solutions which can be rather unconventional in nature. This abstruse nature of GP can be extremely useful while searching a large space for possible solutions. This complements its usefulness for evolving image segmentation algorithms that use a primitive operator based approach.
2. Evolutionary approaches like genetic programming are guided by the fitness of individuals in the population. Therefore, it is not a random search to find an optimal solution, rather a steady refinement towards the search space of more fit individuals [4, 5]. In addition, evolutionary approaches offer a certain black box character as compared to other optimization methods. They make fewer assumptions about the underlying objective functions reflecting optimality of individuals. These functions often don't require deep

insight into the structure of the problem space and are easier to build as compared to building an admissible heuristic for the given problem [48].

3. GP allows symbolic knowledge representation of its solution. This allows considerable ease in readability and visual inspection of the solutions as compared to other techniques like supported vector machines and artificial neural networks which use sub-symbolic knowledge representation. In addition, it also improves portability and reusability of evolved code [26].
4. Segmentation typically represents a class of problems having not necessarily one perfect solution. GP has been successful in dealing with such problems due to its ability to generate multiple equivalent solutions [10, 11, 46].

Finally, we were highly motivated by the promising works of Tackett [47] and Brumby *et al.* [6, 7, 8, 17, 18, 35]. The potential of their impressive approach and reported results were a driving factor during the course of this research.

1.2 PROBLEM STATEMENT

Image segmentation is typically used to extract a region or object of interest from a given image. It is a crucial first and also the most difficult step of any automated image analysis process and is widely used in many computer vision and image processing applications like medical imaging and remote sensing. Even though numerous approaches have been proposed in the past [13, 32, 33, 34], there is still no general segmentation framework that can perform adequately across a diverse set of images. In addition, most image segmentation techniques exhibit a strong domain or application dependence [13, 32, 52]. Automated segmentation algorithms often include *a priori* information of its subjects [14]. This makes use of well-designed, proven segmentation techniques restricted to a small set of imagery. Thus, image segmentation still remains an open and ill-defined problem and a general, automatic image segmentation tool is desirable.

We propose a simple image segmentation algorithm, *GPIS*, which uses genetic programming to evolve segmentation algorithms from a pool of basic low-level image analysis operators. The evolved algorithms are MATLAB programs that readily executable to perform the image segmentation task. In addition, we provide no *a priori* information of images to the algorithm apart from a small set of training images. To check generality of the algorithm, we have tested it on two separate medical image databases and report the results in this thesis.

1.3 Objectives

The objectives of this thesis are manifold and can be described as a combination of factors as given below:

1. **Effectiveness:** The first and foremost objective of this thesis is to develop a GP based algorithm that can produce accurate image segmentation programs for images of varying complexities, where accuracy is defined as the algorithm's ability to correctly classify a pixel as object or non-object (pixel-level classiffication).

2. **Simplicity and Transparency:** The algorithm should produce segmentation programs that are simple in nature and easy to understand. Every image segmentation algorithm, however complex it may be, can be broken down to a set of primitives. Therefore, primitive operator representation should be simple, standardized and easy to read.
3. **Segmentation sans *a priori* information:** The algorithm should require no prior information about the nature of images used or objects to be segmented. The only such requirement is availability of training images, like any other EC based approach. Presently, many image segmenting systems especially for medical imaging require moderate to excessive amounts of *a priori* information which make them application or user dependent.
4. **Generality:** The algorithm should be fairly general i.e. it should able to produce accurate results on a wider application domain. A large number of image segmentation algorithms developed in the past are single-application segmenting tools.
5. **Minimum Human Intervention:** The algorithm should require minimal to no human assistance to derive the required solution.
6. **Ease of Use:** Users without any background in computer vision or image analysis should be able to use the image analysis tool with ease. No formal training should be needed for them to produce desired segmentations. On the other hand, it should allow an expert to "peek" in to the system if he so desires.

We realize that if these objectives are successfully satisfied, it is possible to build an automatic image segmentation engine. This thesis is our attempt to build one.

1.4 Thesis Contributions

The contributions of this thesis are as follows:

1. **Simple approach and anyone with MATLAB can use it:** The proposed algorithm *GPIS* follows a simple evolutionary approach to perform complex image segmentation tasks. The algorithm is implemented on MATLAB. This allows ease in usability to anyone with access to MATLAB.
2. **Open sourced code:** All image operators used in *GPIS* are MATLAB functions. In addition, the GP implementation of GPIS is also done on MATLAB; therefore, all evolved programs are open sourced.
3. **Requires no *a priori* information other than training images:** GPIS performs the required segmentation task without any prior spatial or textural information of objects to be segmented from the given images. It only requires a set of training images like any other EC based approach.
4. **Relatively general approach based on results on the two databases:** GPIS has been tested on two medical image databases for image segmentation tasks. The nature of the images in the two databases was considerably different. The results show that GPIS was able to perform the

given tasks on both databases effectively. This shows that the algorithm is relatively general in nature.

5. **Produces better results as compared to GENIE Pro:** Results obtained for GPIS were compared to another GA based image segmentation algorithm, GENIE Pro. In both cases, the same sets of images were used for training and validation and the same segmentation task was given to both the algorithms. The accuracy of each algorithm was also calculated using the same formula and GPIS performed better than GENIE Pro for both databases.

1.5 A Reader's Guide to the Thesis

This thesis is organized in three parts: *Background*, *Genetic Programming based Image Segmentation (GPIS)*, and *Results and Findings*.

The purpose of the first part, *Background*, is two-fold; provide the reader with the background theory and a grasp of related work undertaken in order to achieve a deeper understanding of the core themes discussed in the thesis and contributions of our work. This section consists of Chapters 2 and 3.

Chapter 2 discusses the terms and concepts of image segmentation, genetic programming and evolutionary computation. It serves as a quick memory refresher for these topics. The chapter ends with a section on GENIE Pro, the comparative tool used for comparing results obtained by GPIS.

Chapter 3 provides a review of related work done in the field of evolution based image segmentation (and image analysis). It also briefly discusses relevant work done in the field of medical image segmentation using evolutionary approaches.

The second part, *Genetic Programming based Image Segmentation, GPIS*, consists of Chapter 4. Chapter 4 is divided into two parts, Part A (*The Proposed Algorithm*) and Part B (*Experimental Settings*). Part A explains the proposed segmentation algorithm, GPIS. It provides a detailed overview of the approach and the design. Part B provides details of the Experimental Setup while testing of the algorithm.

The third section, *Results and Analysis*, is crucial as it details the results obtained during experiments using the proposed algorithm GPIS and an analysis based on the results. It consists of Chapter 5 and 6.

Chapter 5 provides detailed results obtained on application of algorithm on two separate cell databases. It also details the comparative results with respect to GENIE Pro.

Finally, Chapter 6 provides conclusions deduced from the work. It also discusses possible future works in the field. The algorithm has also been applied on some other images also. The results obtained are reported in this section as a plausible option for future work.

A list of references used during the course of the thesis is included at the end.

CHAPTER 2: OVERVIEW OF RELEVANT CONCEPTS

The purpose of this chapter is to provide the reader with the necessary background knowledge of the relevant topics discussed in this thesis. The two underlying concepts in this thesis are image segmentation and genetic programming. They are discussed in Sections 2.1 and 2.3. However, in order to have a better understanding of genetic programming, basic knowledge of evolutionary computation is desirable. Therefore, Section 2.2 is written to provide the reader with a broader foundation to the field and appears before the section on genetic programming. Other common variants of evolutionary computation like genetic algorithms, etc are also briefly discussed in this section. The chapter is concluded by Section 2.4 on GENIE Pro [6, 7, 8, 17, 18, 35]. It is a general purpose GA based image segmentation tool at the Los Alamos National Laboratory, New Mexico. It has been used in this thesis for comparison purposes with the proposed algorithm, GPIS, and as such duly explained.

2.1 IMAGE SEGMENTATION

Image segmentation falls under the category of image analysis, which forms the middle layer in the image engineering framework along with image processing (lower layer) and image understanding (higher layer). The role of image analysis in the framework is fundamentally *image in, measurements out,* i.e. the required information is extracted from the given image and returned as data. As shown in Figure 1, object representation and feature measurement typically complete the middle layer along with image segmentation. Image segmentation is the first step in image analysis and is therefore, a critical one. Depending upon the application, the entire process can be done in a supervised or unsupervised manner.

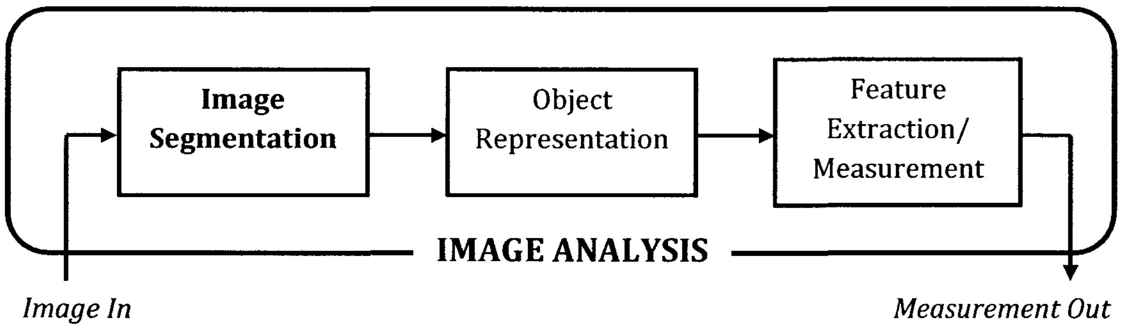


FIGURE 1. TYPICAL LAYOUT FOR IMAGE ANALYSIS

Image segmentation is also one of the most widely used steps in the process of reducing images into information. The main theme is to subdivide the image into its constituent parts and to then extract the objects of interest from it. Based on the objects of interest, the image is intuitively divided into two parts: foreground and background. The foreground consists of the objects of interest while the background is comprised of the remaining image. Segmentation can thus be considered a foreground/background separation process where the selection procedure focuses on one kind of object and ignores the rest [43].

However, it must be remembered that image segmentation is still an ill-defined problem and there is no general solution or defined framework for the segmentation problem. Due to this, techniques developed are often combined with domain knowledge to overcome this problem, thus most segmentation algorithms developed are domain and application dependent.

In order to reflect effectiveness, predictability and reliability of a segmentation scheme, the following criteria [33] can be used as useful pointers. This is relevant especially in our case.

a. *Correctness*: ability to produce segmentation results comparable to human intuition.
b. *Stability with respect to image choice*: ability of the algorithm to produce consistently correct results over a range of images using the same parameter choices

These aforementioned criteria therefore serve as useful indicators when considering reliability of a segmentation technique to serve in a larger system. Over the years, several schemes have been proposed to categorize segmentation algorithms

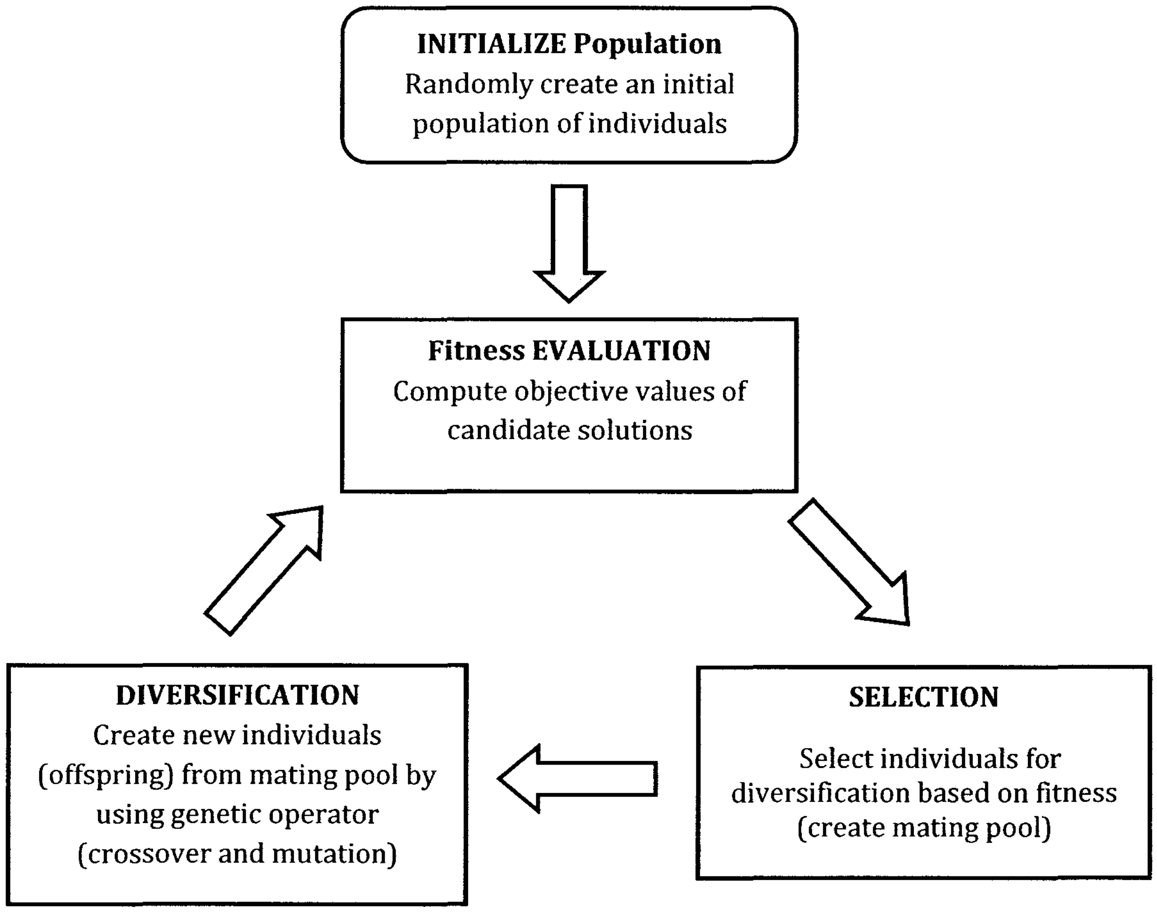
[13, 32]. But due to the nature of the problem, it is difficult to come up with an exhaustive categorization that covers the entire spectrum of segmentation algorithms.

2.2 EVOLUTIONARY COMPUTATION

Evolutionary computation is an umbrella term used to define population-based meta-heuristic optimization algorithms that are inspired by Darwinian evolution. Candidate solutions called *chromosomes* are refined iteratively based on principles of Darwinian evolution, such as natural selection, reproduction (diversification), recombination, crossover, mutation and survival of the fittest [48]. These candidate solutions play the role of individuals in a population and their survival is dependent on a cost function or *fitness function*, which is an objective function that quantifies the optimality of the solutions. Individuals are allowed to breed by applying genetic operators like *crossover* and *mutation* hoping to produce a generation of better (more fit) individuals. The entire process is iterated till the time some terminating condition is met.

The evolutionary cycle is both stochastic (for example, choosing evolutionary operators) and deterministic (for example, selection) in parts. This ensures all individuals have a chance of becoming a parent or surviving a generation, even individuals having low fitness values.

Evolutionary computation employs three key ingredients from Darwinian-evolution: *inheritance*, which allows features to be passed on from parents to offspring, *variation*, which avoids duplication of parents and in turn ensures diversity, and *natural selection*, which can simply be described as survival of the fittest [9]. These features are common in all types of evolutionary algorithms and function in accordance to a basic scheme for evolutionary algorithms [48] illustrated in Figure 2. As seen in the figure, the evolutionary cycle consists of four main steps: *Initialize*, *Evaluate*, *Select* and *Diversify*. The cycle repeats iteratively till the time a physical termination condition hasn't been satisfied.



INITIALIZE Population
Randomly create an initial population of individuals
Fitness EVALUATION
Compute objective values of candidate solutions
SELECTION
Select individuals for diversification based on fitness (create mating pool)
DIVERSIFICATION
Create new individuals (offspring) from mating pool by using genetic operator (crossover and mutation)

FIGURE 2. BASIC CYCLE OF EVOLUTIONARY ALGORITHMS

The basis of division amongst evolutionary algorithms is largely due to difference in representation of individuals (chromosomal representation), diversification strategies and genetic operators adopted. Based on these, evolutionary algorithms are divided into four different paradigms. They are as follows:

1. **Genetic Algorithms:** GAs are a subclass of evolutionary algorithms. The representation of candidate solutions is in the form of binary strings or arrays of other elementary type. Diversification is achieved by crossover and mutation. GAs are typically used to solve scheduling and timetabling problems as well as global optimization problems.
2. **Genetic Programming:** GP is a variant of GA. Here representation of candidate solutions is in the form of computer programs; therefore, it evolves computer programs. Traditionally representation is in the form of a tree structure but non-tree representations like linear genetic programming have also been successfully implemented. GP is particularly useful as an automatic programming tool or as an automatic problem-solving engine [38].
3. **Evolutionary Strategies:** ES are parameter optimization techniques and candidate solutions are vectors of real valued parameters. Therefore the search and the problem space both are comprised of fixed-length strings of floating point numbers. ES are primarily mutation based as use of recombination is less common. The most common forms of ESs are: $(1 + 1)$, $(\mu + 1)$, $(\mu + \lambda)$ and (μ, λ) [39].

4. **Evolutionary Programming:** EP is typically a mutation driven evolutionary algorithm i.e. it only uses mutation as its genetic variation operator. In addition, mutation is performed according to Gaussian distribution and the mutation operator is in turn controlled by parameters that are also optimized [39]. EP is similar to a (μ + μ) ES arrangement without recombination. It must be remembered that there is no exchange of material between individuals in the population.

2.3 GENETIC PROGRAMMING

Genetic programming is a subclass of evolutionary algorithms using computer programs to represent candidate solutions. It is typically an extension of the genetic model of learning into the space of programs. The population is comprised of computer programs rather than fixed length character strings and the effectiveness of the computer program to solve the problem at hand determines its fitness. Thus, they are regularly referred to as the set of evolutionary algorithms that breed program algorithms and similar constructs.

In order to find an optimal solution for the problem at hand, GP proceeds based on the basic evolutionary cycle of initialize, evaluate, select and diversify, as shown in Figure 2. Various schemes for selection, diversification, recombination along with representation are possible for GP. They are discussed briefly as follows.

Representation: There are two main representation schemes popular for GP, tree-based and non-tree based. Traditionally, programs in GP are represented as tree structures. Every node in the tree is assigned an operator function and every terminal node has an operand. This makes mathematical expressions easy to evolve and evaluate. Non-tree based approaches used a linear format to represent programs, much like the linear string representation in GAs. The only difference here is that instead of binary or real numbered values assigned on each bit location as in GAs, each bit location represents a program.

Selection: Selection strategies for GP are similar to GAs. These are generally deterministic in nature. Selection typically occurs twice in every generation of the evolutionary cycle in the form of parent selection (creating mating pool for diversification) and survivor selection (selecting individuals to form population for next generation). There are three widely used parent selection mechanisms: Fitness proportionate, ranking and tournament selection. Similarly, there are two survivor selection mechanism widely used: generational (entire population is replaced each generation) and steady state (a few members replaced in each generation).

Diversification: Diversification in evolutionary algorithms is done using two main genetic operators: crossover and mutation. Both are used for GP.

a. **Crossover (Recombination):** Crossover occurs between two parents. It recombines the selected parents to produce one, two or more offspring. It is analogous to biological crossover (sexual reproduction). Many types of crossover exist, like one-point, two-point, n-point, uniform, cut and splice, etc.

b. **Mutation:** Mutation is one parent genetic operator. It is used to maintain genetic diversity in the population and helps the GP avoid local minima. It is also useful in fine-tuning evolved parameter values. It is analogous to biological mutation. Common forms of mutation are swap, insert, delete, alter, point, uniform, non-uniform, etc.

Elitism: Elitism ensures the best individual(s) in a population are copied to the next generation. It can help increase performance of the GP as it prevents the risk of losing the best found solution(s) in the process so far.

2.4 GENIE PRO

GENIE Pro is a general purpose, interactive and adaptive GA-based image segmentation and classification tool. It was originally developed for analyzing multispectral satellite data but has been upgraded for medical imagery also.

GENIE Pro uses a hybrid GA to assemble image-processing algorithms or *pipelines* from a collection of low-level image processing operators (example: edge detectors, textures measures, spectral orientations and morphological filters). These pipelines sequentially extract multiple features from the same image. Thus, there are multiple pipelines for each image. Each pipeline combines some spatial and spectral processing elements. The evolved programs are constructed by combining the fittest pipelines using a linear classifier (Fisher Discriminant)

Training mechanism for GENIE Pro is different than the conventional method of supplying ground truths. It employs a Java-based tool called ALADDIN which allows user to create training data by manually labeling pixels on training images as feature (True), non-feature (False) and unknown pixels. GENIE Pro thus builds its own ground truth based on this True/False labeling.

There is no in-built termination criteria i.e. the classifier inside GENIE Pro continuously tries to improve the evolved solutions. This can be a drawback as the termination point has to be decided by the user. The usual method for termination is by manually observing the displayed estimated accuracy and stopping the training phase when there is no improvement in displayed accuracy over a period of time.

In addition, the output of GENIE Pro is in the form of a segmented image. It does not provide structural details of the evolved programs. Thus, the only form of comparison possible is the respective segmentation accuracy of the evolved solution.

GENIE Pro has been used as a comparative tool for GPIS. The comparison is based on providing the same training and validation data to both the systems and observing the accuracy of the evolved solution using the same accuracy formula (accuracy formula of GPIS). Details of operating procedure for obtaining results for GENIE Pro are provided in Section 4.6.2.

CHAPTER 3: RELATED WORK

This chapter is written to offer deeper understanding of relevant and representative work accomplished in the field of evolution based image analysis. Our main focus is on GP based approaches for image segmentation. Since GP is a variant of GA, some GA based approaches are also discussed. Image segmentation has an inherent presence in mostly all image analysis applications. Thus relevant GP based feature detection/extraction/classification and object detection approaches are also discussed. This chapter is divided into four sections.

Section 3.1 re-introduces GP and image segmentation.

Section 3.2 discusses previously reported relevant work on GP based approaches for image segmentation related applications. Exception is given to Brumby *et al.* [6, 7, 8, 17, 18, 35] who have developed a hybrid evolutionary approach for feature extraction/classification. But the work is a better fit under this category.

Section 3.3 discusses relevant evolutionary based approaches for medical image segmentation.

Section 3.4 briefly discusses non-EC based approaches for cell segmentation. This is done to provide a better understanding of the problem and methods used to solve it.

3.1 INTRODUCTION

Genetic programming is a variant of genetic algorithms that evolves the entities that process the data, programs, rather than fixed-length character strings, to solve the given problem. It was first proposed by Koza [27] in 1992 and has since been successfully used in numerous applications as an automatic programming tool, a machine learning tool or an automatic problem solving engine [2, 27, 38, 47].

On the other hand, image segmentation is vital to many image processing and computer vision applications. Due to its inherent presence in image analysis related tasks, it has been of tremendous importance in fields like medical image analysis [14, 24, 40], remote sensing [6, 7, 8, 10, 11, 17, 18, 35, 46], face recognition [49], natural scene recognition [30], handwritten character recognition [29], texture classification [45], military applications [20, 21, 41, 47], target/vehicle detection [20, 21, 47], agricultural product classification [50], etc. Understandably, a fair amount of evolutionary techniques like GAs and GP have been applied to solve segmentation-related tasks.

3.2 GP Based Approaches For Image Segmentation

This section presents a representative study of relevant work done in the field of GP based image segmentation (image analysis included). It also includes relevant hybrid GP approaches. The section is concluded with a tabular representation of key features of the methods discussed.

Tackett, 1993

Tackett [47] published one of the initial works in the field of GP based image analysis. He applied GP to develop a processing tree capable of classifying features extracted from images. The work focused on using image features to locate target vehicles (tanks) in a cluttered terrain and the task of the GP was to evolve these features in order to construct a classifier. The GP was successful in creating a better strategy for target detection using primitive features directly rather than deriving statistical features from these primitive features. The evolved solutions were not only better but also faster as compared to artificial neural networks and binary classification trees, as the GP used limited number of features rather than the entire gray-scale image. This was also the first time that GP was applied to a set of non-binary images in a cluttered environment.

Johnson *et al.*, 1994

Johnson *et al.* [25] applied a variant of GP (typed GP) to a fiducial-point-localization problem (localizing a structure known to be present in the image). They described a method of automatically evolving visual routines for simple tasks using GP. The task was to create visual routines for locating the left and right hands in a silhouette images of people which were obtained from real images via segmentation. A distinct feature of their work was the way they applied crossover. Crossover was performed between two parents by exchanging sub-trees of the same root type. To avoid bloat, they simply discarded offspring which had depth greater than an admissible limit. The GP evolved programs that were correctly able to locate the left arm in 93% of the images and the right arm in 70% of the images, which was much better than the best algorithm the authors were able to write by hand.

Harris and Buxton, 1996

Harris and Buxton [16] applied GP to evolve optimal linear filters for edge detection in signals and images. This was a relatively different objective as compared to the usual image analysis applications. But the GP was able to evolve edge detectors that outperformed Canny's edge detector which is one of the most popular edge detectors in literature. Since the approach required convolving masks, the authors preferred working with 1-D signals rather than 2-D (images), even though the approach is extendable to image analysis.

Poli, 1996

Poli [37] proposed an approach to image analysis based on evolving optimal filters. He was successful in evolving effective filters for image enhancement, feature detection and image segmentation and was able to present the above as a purely filtering problem. Although GP was applied in a naïve way, he was able to outline certain criteria for terminal sets, function sets and fitness function to make the search feasible for producing efficient filters. The GP was applied to the segment Magnetic Resonance images of the brain and the results produced were far better than the ones produced using a neural network. Not only was the segmentation better but the rate of misclassifications was also found to be much lower.

Daida *et al.*, 1995-96

Daida *et al.* [10, 11] were the first to use the GP paradigm for IP applications in geosciences and remote sensing by deriving spatial classifiers for SAR imagery. Their GP was able to extract pressure-ridge (curvilinear) features in SAR images, a problem for which there has been no known satisfactory solution. Instead of specifying the performance metrics to the GP in advance, they proposed an interactive approach where the user tested the robustness and validity of the GP-derived solution on an out-of-sample subset. This approach was called *scaffolding*. Stanhope and Daida [46] furthered this approach using GP to generate rules for identification of objects in automatic target classification of SAR images.

Winkeler and Manjunath, 1997

Winkeler and Manjunath [49] used GP for face recognition purposes (typically small target classification problem). The experiments used GP as a learning strategy for detecting faces in a cluttered environment. Their approach was slightly different from the conventional approaches. They combined two programs, evolved separately in different experiments that used different features to increase the detection rate. The feature detector obtained after combining the programs produced better results when compared to a single feature detector.

Brumby *et al.*, 1999-2002

Brumby *et al.* [6, 7, 8, 17, 18, 35] applied a hybrid evolutionary approach to develop an automated feature detection/classification system called GENIE (Genetic Imagery Exploitation). GENIE was initially designed to generate image feature extraction algorithms for remote sensing applications [6, 35]. It was applied to evolve image processing pipelines (sequence of primitive image processing functions) to segment and extract features from data sets of multi-spectral aerial-photography. After [6], it was concluded that an EC approach for accelerated image analysis tool-making was not only possible but also a viable option to develop extraction algorithms for novel features and data sets. Thus, in their subsequent work [7, 8, 17, 18], GENIE was successfully applied to a variety of multispectral imagery and remote-sensing data including extracting land cover features from multiple data sources (Multispectral Thermal Imager spacecraft images) [8, 18], post-wildfire land-cover mapping (LANDSAT Enhanced Thematic Mapper images) [7, 8], automatic feature extraction for panchromatic Mars Global Survey Mars Orbiter Camera Imagery (evolving algorithms for finding mesas and craters) [35] and detecting for golf courses in remotely-sensed data [17]. In 2005, an upgraded version, Genie Pro was released and has since been used for remote sensing purposes as well as biomedical imagery applications.

Belpaeme, 1999

The approach of using GP to evolve feature detectors from primitive image processing functions (primitives) was also investigated by Belpaeme [3]. He showed how sets of visual feature detectors could be evolved starting from simple primitives. These primitives were combined using GP in a feed-forward feature-extraction hierarchy. The experimental results showed that the GP was able to successfully construct visual functionality based on the primitives under selective (selectionistic) pressure. The inputs for the feature detectors were a series of real-world images containing objects or faces.

Howard *et al.*, 1999

Howard *et al.* [20, 21, 41] presented a series of works using a GP strategy for automatic object detection purposes in real world and military image analysis problems. They proposed a staged evolutionary approach for evolution of target detectors or *discriminators.* Feature detection for the given target was broken down to a pixel-by-pixel level. Since such an operation is CPU intensive, they broke down the evolutionary cycle into stages. The first stage required the GP to discriminate every feature or *object* pixel from a random selection of unclassified non-object pixels. Upon completion of the first stage, the fittest detector was applied to the entire image which predictably resulted in a number of misclassifications (False Positives). In the second stage, a new GP was applied to classify object pixels from the previously discovered False Positives. Finally, the fittest detector from the first stage was fused with the fittest detector from the second stage. This division of the evolutionary cycle greatly reduced the CPU time as well as limited unwanted program growth. The two stage GP was later generalized into a multi-stage process and applied to a variety of real world images (two stage GP – automatic ship detectors for low-resolution SAR imagery, multi stage GP – automatic vehicle detectors for infrared linescan or IRLS imagery). In [41], they extended this approach to detect hundreds of vehicles in 25 miles of reconnaissance imagery.

Bhanu *et al.*, 2002-04

Bhanu *et al.* [4, 5] used GP to evolve composite operators for object detection. These operators were synthesized from combinations of primitive image processing operations used in object detection, thus called composite. In order to control the code-bloat problem, they proposed a size limits for the composite operators. In [5], they used a *hard* size limit for the composite operators but this provided severe restrictions on the GP search. They thus changed it to a *soft* size limit in [4]. The efficacy of the GP was tested to extract regions-of-interest from SAR images, infrared images and RGB color images and the results showed that GP provided a viable way of synthesizing composite operators for object detection problems.

Zhang *et al.*, 2006

Zhang *et al.* [51] proposed a different configuration of GP for object detection. They wanted to investigate ways of improving efficiency and effectiveness of GP techniques rather than investigating applications of GP on object detection. They proposed a two-phased approach to construct object detection programs. First, they applied GP to create trained GP classification programs on smaller sized images from the training set. These trained GP programs were then used to initialize the second phase of the GP process which used larger images from the same training set. The purpose of the second phase was to refine the trained programs evolved in the first phase for object detection. Fitness functions were different for each phase. GP in phase 1 learned classification rules while these rules were refined for object detection in phase 2. The system was tested on a dataset of coins for Heads/Tails classification. Even though the results were more effective and efficient than the basic GP approach, programs evolved still contained some redundancy.

If observed closely, the methods proposed by Tackett [47], Bhanu *et al.* [4, 5], Belpaeme [3] and Brumby *et al* [6, 7, 8, 17, 18, 35] have one thing in common, the underlying use of primitive image operators for a variety of image analysis tasks. The GP in each case uses evolution to discover these image operators that are capable of efficiently segmenting the regions/features of interest. Their terminology might differ among literature: composite operators [4, 5], primitives [3], image processing pipelines [6, 7, 8, 17, 18, 35] but each suggested the concept was extendible, provided the operators are domain-independent.

Geosciences and remote sensing is a major application domain of evolution-driven image processing and segmentation techniques [4, 5, 6, 7, 8, 10, 11, 17, 18, 20, 21 35, 41, 46, 47]. Due to the plethora of satellite imagery available, in volume, type (for example SAR, LANDSAT, AVIRIS, MTI, MODIS, SPOT, DOQ) and application, there has been substantial development of data processing models for automatic feature

TABLE 1. GP BASED APPROACHES FOR IMAGE SEGMENTATION

AUTHOR/ YEAR	IMAGE ANALYSIS TASK	REPRESENTATION SCHEME / WHAT GP EVOLVES	IMAGE TYPE / APPLICATION	SALIENT FEATURES
Tackett [47]/ 1993	Feature extraction/ classification	Tree	IR	1) Evolution of processing tree for feature classification 2) Primitive operators 2) Cluttered environment
		Feature classifiers	Military	
Johnson *et al.* [25]/ 1994	Feature extraction, Topographic correctness	Tree	Bitmap Silhouette	1) Automatic evolution of visual routines 2) High level function and terminal sets 2) Typed GP
		Visual routines	Fiducial-point-localisation/ Other simple tasks	
Harris and Buxton [16]/ 1996	Edge Detection	Tree	Not Applicable	1) Production of linear filters for edge detection 2) Outperformed Canny's edge detector
		1-D & 2-D Edge detectors	Signal/ Image Processing	
Poli [37]/ 1996	Image enhancement, Feature detection, Image Segmentation	Tree	MR Images of brain	Learning of composite filters for image analysis
		Filters	Medical	
Daida *et al.* [10,11,46] / 1995, 1996	Feature extraction/ classification	Tree	SAR	1) Generate rules for target identification & classification 2) Interactive scaffolding approach
		Spatial classifier (ROI spatial filters)	Remote Sensing, Geosciences	

(CONTINUED)

TABLE 1. GP BASED APPROACHES FOR IMAGE SEGMENTATION (CONTINUED)

AUTHOR/ YEAR	IMAGE ANALYSIS TASK	REPRESENTATION SCHEME / WHAT GP EVOLVES	IMAGE TYPE / APPLICATION	SALIENT FEATURES
Winkeler and Manju--nath [49]/ 1997	Object detection/ Small target classification	Tree	Human faces	Combining two separately evolved programs to build an improved resultant detector
		Facial features	Face recognition	
Brumby *et al.* [6,7,8,17, 18,35], 1999, 2000-02	Feature Extraction/ Classification	Linear	SAR, AVIRIS, MTI, LANDSAT	1) Hybrid EC 2) Primitive operators 3) Pixel-by-pixel classification
		Feature detection algorithms (using primitive operators)	Remote Sensing, Geosciences, Medical	
Howard *et al.* [20,21,41] / 1999	Object detection	Tree	SAR, IRLS	1) Staged evolutionary cycle 2) Drastic reduction in CPU time for the evolutionary process
		Feature detectors for: 1) Object pixels from random non object pixels 2) Object pixels from False Positives	Military/Real world	
Belpaeme [3]/ 1999	Feature Extraction (Visual feature detection)	Tree	Real world images (faces, objects)	1) Evolving feature detectors under selective pressure 2) Primitive operators 3) Feed-forward feature-extraction hierarchy
		Feature detection	Visual feature detection, Facial discrimination	

(CONTINUED)

TABLE 1. GP BASED APPROACHES FOR IMAGE SEGMENTATION (CONTINUED)

AUTHOR/ YEAR	IMAGE ANALYSIS TASK	REPRESENTATION SCHEME	IMAGE TYPE	SALIENT FEATURES
		WHAT GP EVOLVES	APPLICATION	
Bhanu *et al.* [4,5]/2002, 2004	Object detection	Tree	SAR, IR and RGB	1) Composite primitive operators 2) Soft composite size limit (bloat control)
		Composite operators	Road, lake, river, field, tank, people, car, SUV extraction	
Zhang *et al.* [51]/ 2006	Object detection	Tree	Coins	1) Two phased GP: learning (classification) and testing (detection) 2) Two fitness functions
		1) Feature classification rules 2) Refined object detection	Improving conventional GP techniques	

extraction and classification using EC techniques. Most of the initial evolution-driven image analysis approaches were hence designed for these imageries. This is important as these models can be extended to various other domains including medical imaging.

A tabulated version of the methods discussed in Section 3.2 is presented in Table 1. It breaks down these approaches based on modality of the use of GP, chromosomal representation scheme, type of test images and applications. Knowledge based on these fields can help determine the extendibility of the approach to other problem domains. Finally, selling points of the approaches are mentioned under Salient Features.

3.3 EC Based Approaches For Medical Image Segmentation

This section reviews relevant EC based approaches for medical image segmentation. The section is concluded with a tabular representation of key features of the methods discussed.

Jiang *et al.* 2001

Jiang *et al.* [24] proposed a novel approach for cell image segmentation using kernel-based dynamic clustering and a parallel GA. Tempted by benefits of including *a priori* information, they used not only prior edge information of the cells but also shape information of cell contours. They transformed this cell segmentation problem into an optimization problem based on three steps: (i) find possible edges of the cell using Canny's edge detector, (ii) localize cells positions and find possible image points in cell boundary, and (iii) construct a parameterized cell contour model to detect cell contours. Use of *a priori* information made the approach better resistant to image noise and allowed parameterization of the problem for the application of the GA. But it also made it heavily application dependent. The parallel GA was able to produce accurate solutions and improved the algorithm's speed of convergence.

Roberts and Claridge, 2003

Roberts and Claridge [40] proposed a GP based image segmentation technique for segmenting skin lesion images. The GP was implemented in a cost based sub-tree caching fashion. The function set contained imaging functions like thresholding, quantization, region intensity functions, etc while the terminal set contained input images, and numerical and coordinate values. The fitness function used was based on sensitivity and specificity, similar to the one used by Poli [37]. They used a pool of 100 images, from which 8 were used for training and the remaining as test images and found favorable results on the database. Their GP was affected by code-bloat to some extent which increased execution time of both the GP, and the evolved solution. However, pruning of the best evolved programs helped reduce execution times.

Ghosh and Mitchell, 2006

Ghosh and Mitchell [14] proposed a GA for automating segmentation of computed tomography images of the pelvis. The approach was primarily based on active shape modeling for texture-based segmentation. In their two-staged approach, a GA was used to evolve a segmenting curve represented by a level-set function. Training images contained a hand-drawn contour around the object of interest. When a new test image was presented, the goal of the GA was to evolve a contour that segmented the desired object in the new image such that the contour obeyed shape constraints learned during training and enclosed a region whose texture was a good match for textures learned on the training images. The method combined high-level textural and shape information for segmentation and allowed use of any kind of high level textural features for doing segmentations. *A priori* information about shape and texture was able to constrain the evolution of the segmenting curve over successive generations.

TABLE 2. EC BASED APPROACHES FOR MEDICAL IMAGE SEGMENTATION

AUTHOR/ YEAR	TYPE OF EA	TASK OF EA	SEGMENTING FEATURE/ IMAGE MODALITY	SALIENT FEATURES
Jiang *et al.* [24]/ 2001	Parallel GA	Cell segmentation by building contour around cell boundary	Cell images of hypothyroid and small intestine	1) Kernel-based dynamic clustering 2) Contour modeling of cell 3) *A priori* information extensively used 4) Faster convergence due to use of parallel GA
Roberts and Claridge [40]/ 2003	GP	Evolve segmentation programs	Skin lesion images	1) Cost based sub-tree caching mechanism 2) Pruning required in evolved solutions.
Ghosh and Mitchell [14]/ 2003	GA	Evolving contours (level-set function) that enclose the object of interest	2-D Slices of Pelvis CT images	1) Fitness function based on *a priori* information (textural & shape) 2) Two-staged approach: a) Training (deriving shape/textural priors) b) Priors-based Segmentation

As seen in the methods of Jiang *et al.* [24] and Ghosh and Mitchell [14], many medical image segmentation approaches are heavily dependent on *a priori* information provided to the algorithm. It makes these novel approaches heavily application dependent which greatly reduces their scope outside their image domain without significant changes to their models.

3.4 A WORD ABOUT CELL SEGMENTATION

Little work has been done in the field of EC based approaches for cell segmentation. Other non EC based approaches specific to cell segmentation fall under two main categories: region-finding and contour detection algorithms. Region-finding approaches are basically thresholding based while contour-detection algorithms rely on discontinuity in texture or intensity at the object boundary. While region-finding approaches are computationally expensive, contour-detection algorithms are prone to noisy images. Another popular technique for cell segmentation is use mathematical morphological operations [12, 42]. Many approaches combine use of morphological operations in conjunction with traditional segmentation techniques [12, 28]. Some of the other popular cell segmentation approaches include watershed transform [23], shape analysis [31], scale-space filtering and HSV histogram clustering [23]. Understanding of these non-EC based approaches acts as a useful pointer while designing a primitive operator based segmentation approach for cell images.

CHAPTER 4: THE PROPOSED ALGORITHM
GENETIC PROGRAMMING BASED IMAGE SEGMENTATION (GPIS)

GPIS is a genetic programming based image segmentation algorithm. It employs a classic evolutionary paradigm: a population of candidate solutions known as *chromosomes* is maintained, each composed of interchangeable parts called *genes*, and each chromosome is evaluated and assigned a scalable fitness value based on how well it performs the required task. Once fitness is assigned, evolutionary operators for selection, recombination (crossover) and mutation are applied to the entire population. The population is evaluated again and followed by selection, crossover and mutation. The cycle is iterated till the time some termination condition is satisfied.

This chapter details every element of the evolutionary cycle (Part A) as well as the experimental settings (Part B) for GPIS.

PART A: THE APPROACH

4.1 OVERVIEW OF THE APPROACH

An overview of the approach is provided in Figure 3. As seen in the figure, the GP follows a classical training-validation paradigm of EC. The GP is given access to an image library which provides the necessary images for training and validation along with their corresponding ground truths (GT). A ground truth is a perfect representation of the ideal solution that the GP strives to attain. We use manually hand segmented ground truths for both databases in the Image Library.

In the beginning, a population of individuals is randomly initialized and is trained on a set of training data, typically ground truths of the corresponding set of training images. Individuals are nothing but image segmentation programs composed of a set of primitive image analysis operators chosen from the given operator pool. The effectiveness of these programs is checked using a fitness function. Thus, every program in the population is assigned a *fitness* measure. Fitness determines the optimality of a program and greatly affects its chances for selection to produce an offspring or survive a generation. Every generation, a set of programs are *selected* to undergo crossover and mutation operations, thus producing new *offspring* programs. Selection of these programs is fitness-based. As in natural evolution, chances here of fitter parent programs producing fitter offspring programs are high. The cycle is repeated iteratively till the time an optimal program is evolved. Once an optimal program is discovered, training ends and validation begins.

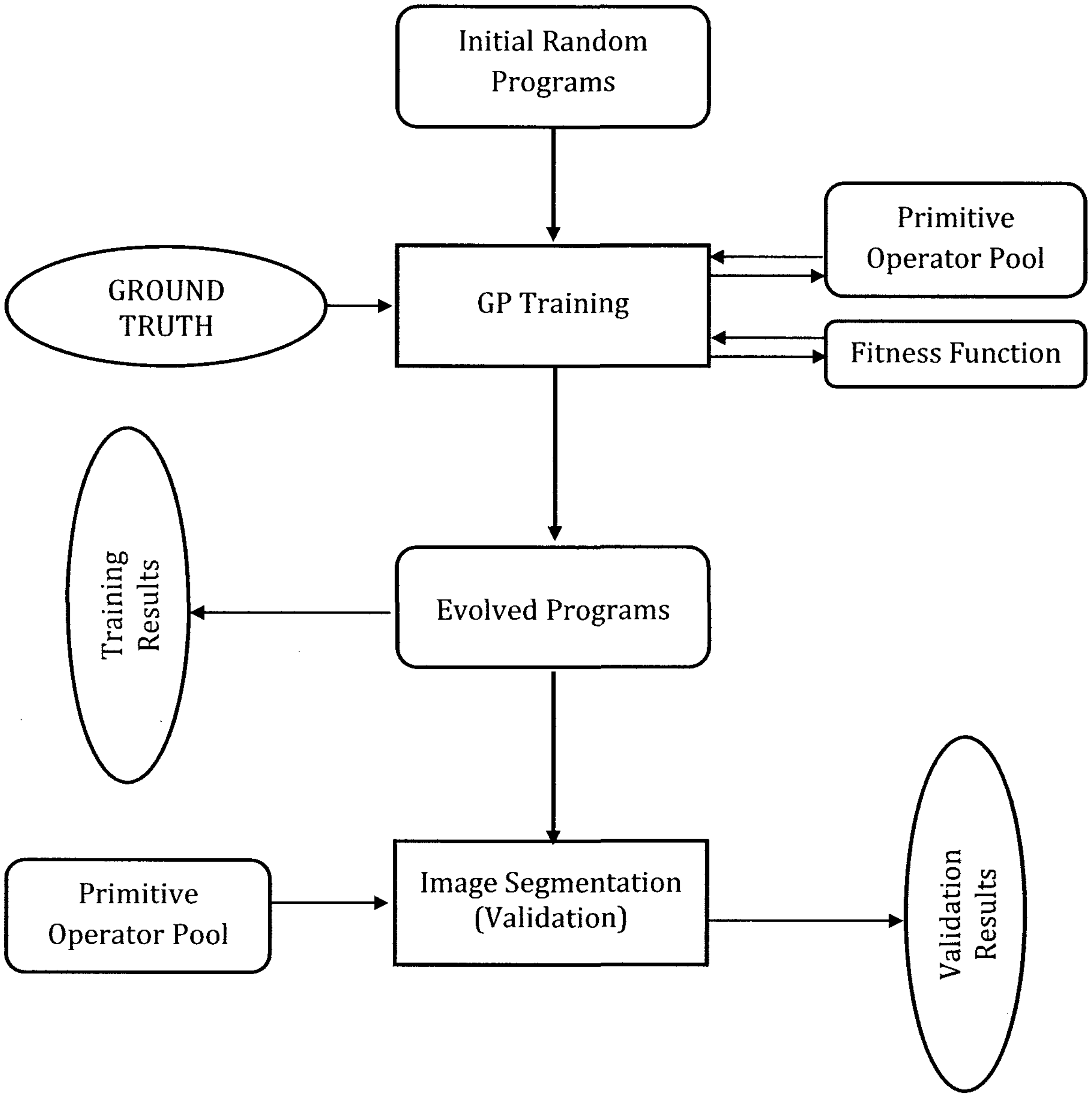


FIGURE 3. GPIS: AN OVERVIEW OF THE GP APPROACH

This is essential in order to appraise the quality of the evolved program. During validation, this evolved program is tested on a set of validation images, separate from the set of training images and are generally much higher in number than the training images. If training is accurate, the evolved program should produce segmentations similar to those on the training set. In order to achieve this, it is essential that firstly, diversity of the population is maintained so that the search does not get trapped in local optima. Secondly, the pool of primitive image operators should be sufficient, both in validity and consistency.

4.2 SOFTWARE ARCHITECTURAL OVERVIEW

An alternative view of the approach is provided in Figure 4. Here the approach is described from the *Software Architecture* point of view. As mentioned earlier, individuals in a population are in effect image segmentation programs (*chromosome*). Each image segmentation program is composed of a number of image analysis primitives (*genes*). Each primitive works on the input image provided to it and produces an output image. In reality, this is determined by its relative position in the sequence of primitives comprising the segmentation program. There is a possibility that an intermediate output produced in the sequence might greatly improve the fitness when combined with another such output or the final output. Hence, it is wasteful if these intermediary outputs are discarded. Thus, every intermediate output is stored on a separate plane known as *InterOut Plane*. These planes are of the same dimensions as the output image they store. In addition to this, the original input image from the training set is stored on

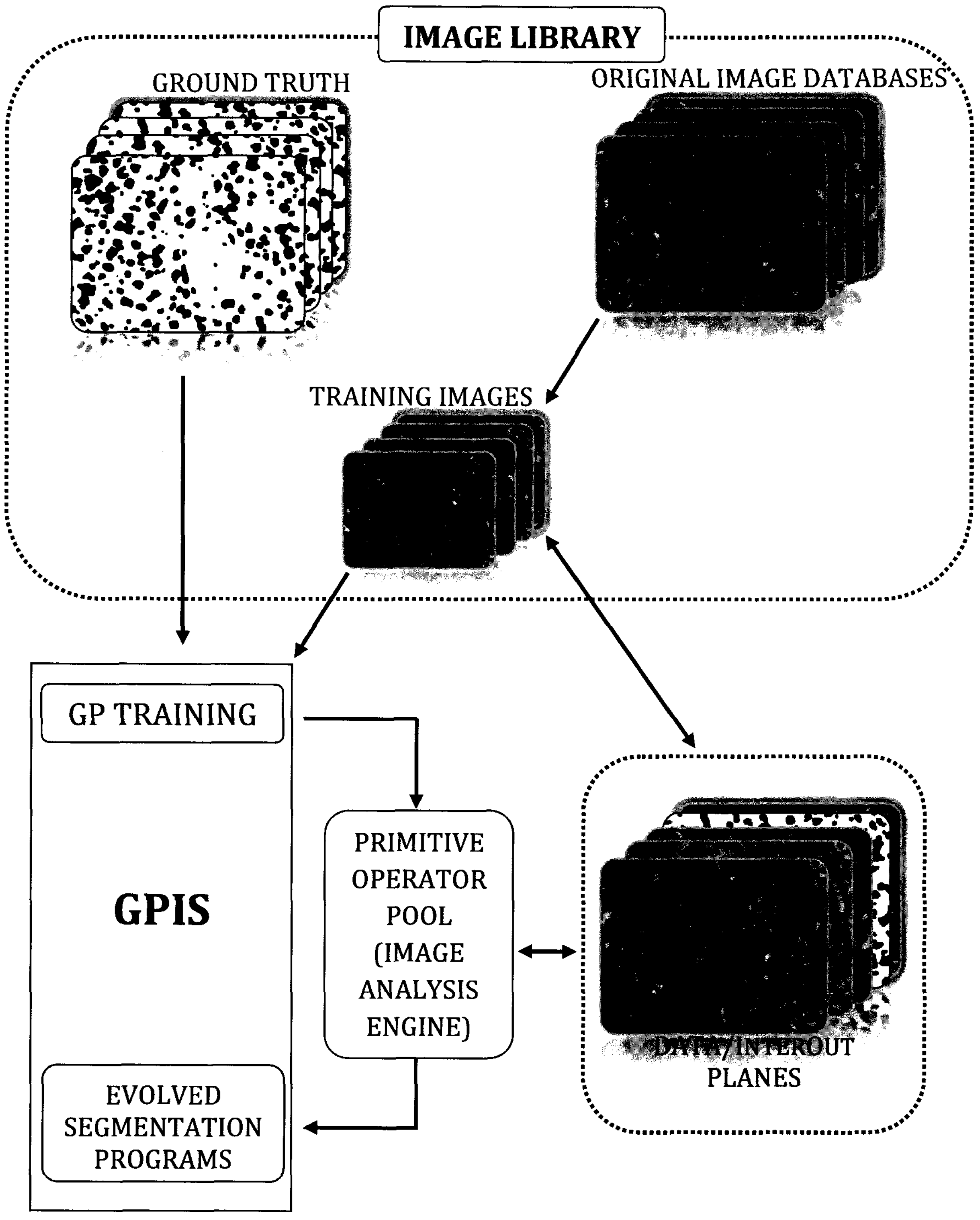


FIGURE 4. SOFTWARE ARCHITECTURE OF GPIS

what are known as *Data Planes*. Data planes are also of the exact same dimensions as the input image it stores. Storing the images in this format greatly helps in maneuverability of images during the operation of the GP.

4.2.1 TRAINING WINDOW OPERATION

GPIS uses a smart training mechanism to increase speed of the evolutionary cycle and assist in refinement of primitive operator features. Before a training image is copied onto the data planes, the GP gets an option to choose the size of the training image in question which is known as the training window of the image. Thus, instead of using the entire image for training, a portion of it can be selected and copied onto the data plane. Seeding as well as size of the training window is determined randomly. We found this method especially helpful as it increased the possible modalities of training images manifold. In addition, a successful implementation of a similar approach was reported in [51] as part of their two-phased GP. We kept the minimum size of the training window at $80 \times 80\, pixels^2$. The maximum size is the entire image. Correspondingly, the ground truth of the training image is also chosen of the same size and spatiality as the training image. The training window operation is shown in Figure 5.

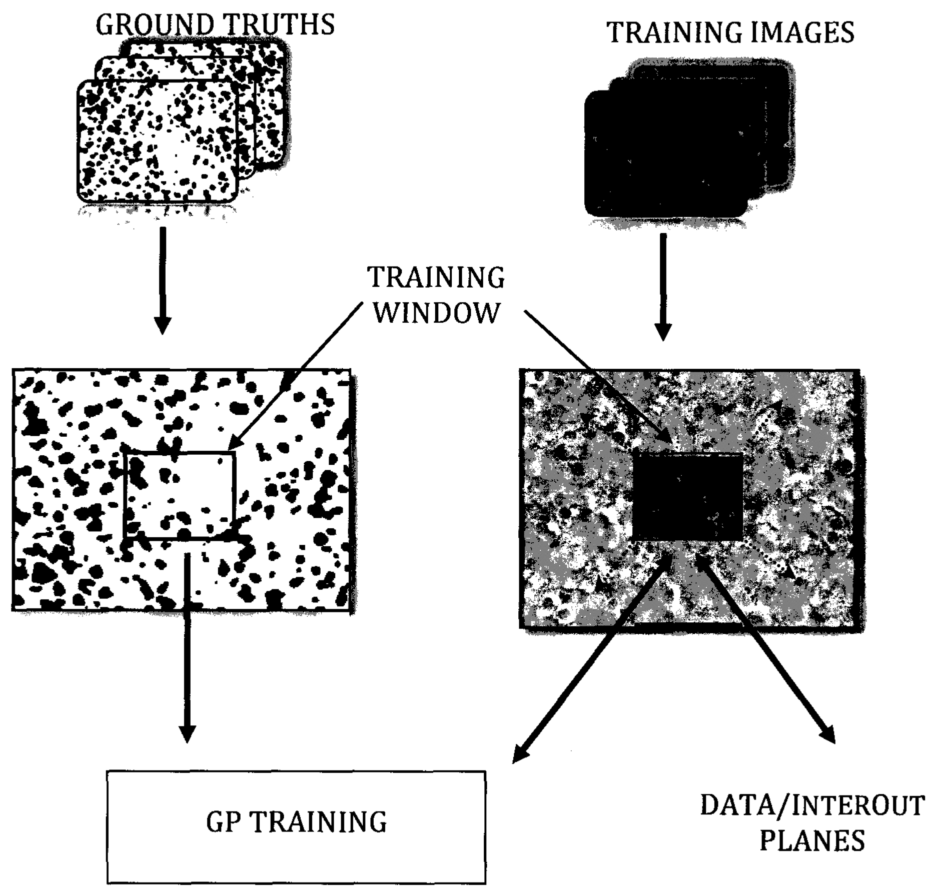


FIGURE 5. TRAINING WINDOW OPERATION

4.3 DESIGN OVERVIEW

GPIS uses an evolutionary based approach to discover image segmentation algorithms from a given pool of primitive image analysis operators. A population of such algorithms is created and evaluated every generation by measuring their fitness to the given environment. Natural selection takes place and survivors are allowed to diversify giving rise to newer algorithms. This process continues till the time the GP evolves an optimal algorithm. This section provides details the building blocks of the evolutionary process of GPIS.

The flowchart for the GP is presented in figure 6. Individual blocks of the flowchart are explained later in the chapter.

4.3.1 FLOWCHART – OPERATIONAL WORKING OF GPIS

The GP begins with a randomly generated population of programs having a maximum length of 15 primitive operators (*Initialization*). This population is evaluated for fitness and is ready to undergo diversification (*Evolutionary Operators*). But in order to do so, a selection mechanism is applied to create a pool of parent programs (*Parent Selection*). This is done using tournament selection with the size of the tournament window being 10% of the size of the population. 50% of the population is selected to be parents. Evolutionary operators in the form of *Crossover* and *Mutation* are applied probabilistically to create new programs (*offspring*). In number, this new population that consists of offspring programs is 49% of the total population size. This is because the top 1% of the original population is copied directly to the next generation (*Elitism*) in order to avoid them from getting destroyed or not selected for the next generation. However, these *elite* programs are not removed from the original population; therefore, they are still available to undergo crossover and mutation. The selected parents that undergo diversification *survive* the generation and are sent to the next generation. The remaining population of the subsequent generation is completed by the *offspring* programs created in the present generation. In addition, to maintain diversity of the population, we use an *Injection* mechanism to inject a new randomly initialized population of programs (20%) every 5 generations. This avoids creation of *too many similar individuals* which might cause the search to get trapped in local optima. Whenever injection takes place, a survivor selection mechanism is implemented. From the 99% new population created from the present generation, the top 79% is selected based on fitness and added to the injected population. The *elite* programs are now added to this population and concluding the formation of the new population for the next generation. Once the new population is created, it undergoes fitness evaluation and the cycle repeats till the time the stipulated termination condition is satisfied.

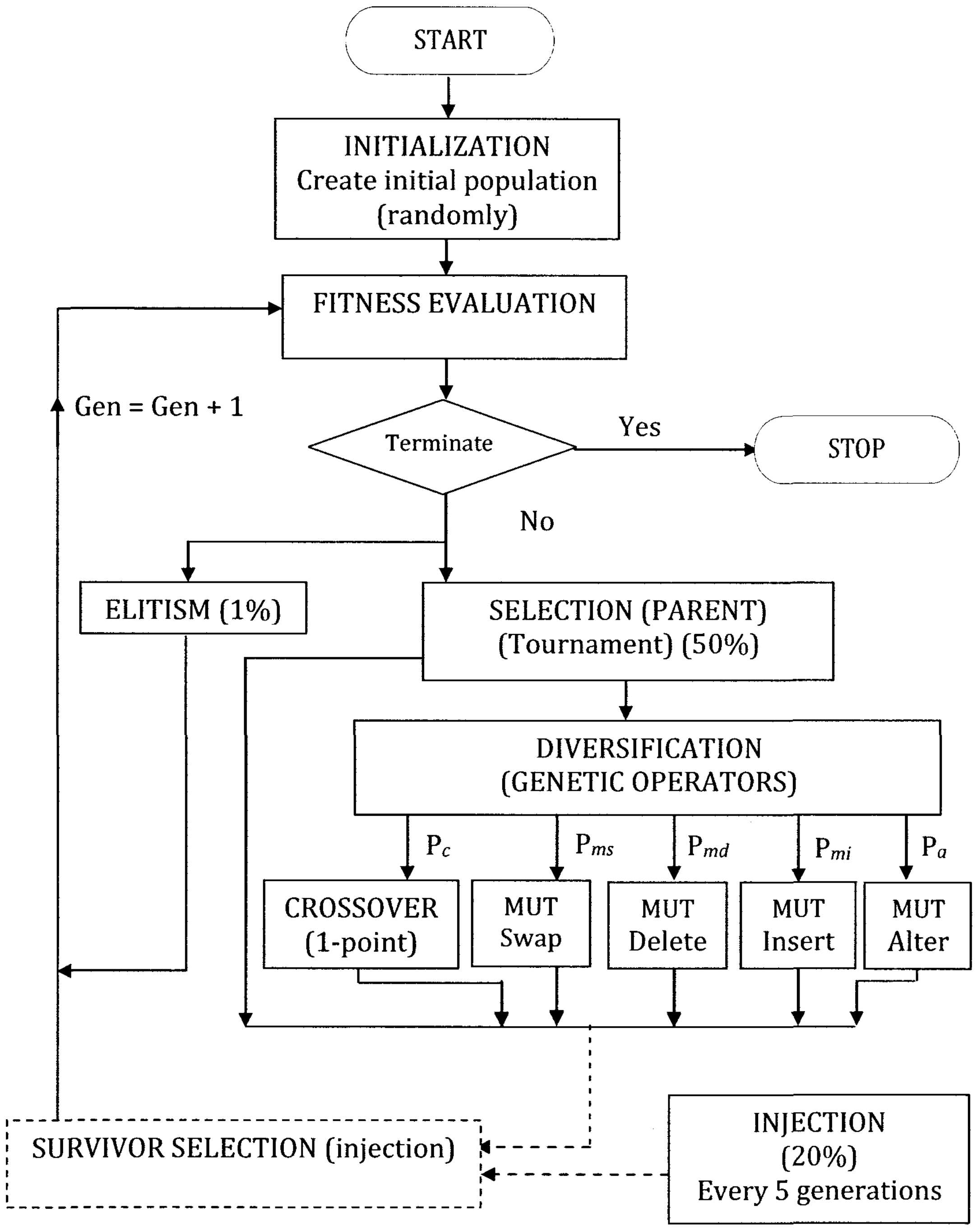


FIGURE 6. FLOWCHART OF THE GP

4.3.2 REPRESENTATION

Representation in essence is a way to define what a possible solution to the problem must look like. In actuality, it defines the space of candidate solutions the GP can find to the given problem [2]. The representative scheme for GPIS is as follows:

4.3.2.1 GENES: PRIMITIVE IMAGE ANALYSIS OPERATORS

Genes are the building blocks of a chromosome. In GPIS, these genes are basic low-level image analysis operators, also known as *image primitives*. We use a simple notation to encode genes. The general layout of a gene can be seen as below:

[OPERATOR, INPUT 1, INPUT 2, WEIGHT, STRUCTURING ELEMENT]

The first field represents operator name, typically an image analysis function. The second and third fields represent the input planes to the operator. A gene can operate on one or two inputs, depending on the type of the operator. The fourth field indicates a weight or parameter value (if needed) for the operator. Finally, the last field refers to the type of morphological structuring element used by the operator (if used). Morphological operations are an essential component of the imaging process and typically work by using structuring elements. The first two bits always have a value (operator name, input plane 1). The rest may or may not have a value at all times. This is dependent on the nature of the operator. If these fields don't have a value, they will be typically represented as 0 (zero). Therefore, a possible gene might look like: [HIST, d1, 0, 0, 0], [OPEN, io1, io4, 0, 4] or [ADDP, io1, io2, .2, 0].

4.3.2.2 CHROMOSOMES (ENCODING)

We use a linear representation scheme for chromosomes. Chromosomes in effect are text strings of genes. Therefore, a possible chromosome of length 4 might look like:

[HIST, d1, 0, 0, 0] [SUBP, io1, d1, .2, 0] [DIL, io2, 0, 0, 4] [LAPL, io3, 0, -4, 0]

where *d1* denotes *Data Plane 1* and *io* denote the corresponding *InterOut Planes.* HIST, SUBP, DIL and LAPL are the primitive operators.

There have been two checks included in the algorithm for avoiding run-time errors and limiting unnecessary computation to reduce redundancy of operators in a chromosome. These are namely, an image compatibility check and a parsimony operator. The image compatibility check ensures that the format of the image (RGB, grayscale, binary) is compatible with the operator in use. This is essential as some operators typically function on only a particular image format. If such a check is not included, it can lead to run-time errors. A *parsimony operator* is implemented to remove portions of the chromosome which do not contribute to the final outcome the segmentation produced by the chromosome. This happens primarily due to the crossover operation. These unwanted portions typically make the resultant program bulky and increase execution time of the program.

4.3.2.3 OPERATOR POOL

We use 20 primitive operators in all. Table 3 provides the complete list of these. In addition, table 4 provides a list of the structuring elements used. A total of 8 structuring elements have been used.

The generality of each operator has been maintained. This allows for the algorithm's availability for future use on databases other than the ones it was originally built on. Another important factor is to maintain the sufficiency of the operator pool without over-flooding it with operators. In order to achieve this, the pool was built using a progressive process of elimination. Initially, maximum numbers of operators were included in the pool. Based on the performance and utility of each operator, its significance was observed and only those operators that were necessary for the pool were finally included.

4.3.3 INITIALIZATION

The initial population to the GP is randomly generated i.e. programs (chromosomes) are formed by a random sequence of operators. The parameter values of operators are also assigned randomly. It is a blind random parallel search of the search space that is made up of the primitive image analysis operators. For practical reasons, the size of each program is limited to a *maximum depth*. In our case, we define the maximum depth of 15. The fitness of this population is expected to be low.

In addition, respective values of crossover rates and mutation rates are also set at initialization. These values are provided by the user.

TABLE 3. LIST OF PRIMITIVE IMAGE ANALYSIS OPERATORS

No.	OPERATOR	DESCRIPTION	INPUTS	TYPE	STRUCTURING ELEMENT
1.	ADDP	Add Planes	2	Arithmetic	No
2.	SUBP	Subtract Planes	2	Arithmetic	No
3.	MULTP	Multiply Planes	2	Arithmetic	No
4.	DIFF	Absolute Difference	2	Arithmetic	No
5.	AVER	Averaging Filter	1	Filter	No
6.	DISK	Disk Filter	1	Filter	No
7.	GAUSSIAN	Gaussian Filter	1	Filter	No
8.	LAPL	Laplacian Filter	1	Filter	No
9.	UNSHARP	Unsharp Filter	1	Filter	No
10.	LP	Lowpass Filter	1	Filter	No
11.	HP	Highpass Filter	1	Filter	No
12.	DIL	Image Dilate	1	Morphological	Yes
13.	ERODE	Image Erode	1	Morphological	Yes
14.	OPEN	Image Open	1	Morphological	Yes
15.	CLOSE	Image Close	1	Morphological	Yes
16.	OPCL	Image Open-Close	1	Morphological	Yes
17.	CLOP	Image Close-Open	1	Morphological	Yes
18.	HISTEQ	Histogram Equalization	1	Enhancement	No
19.	ADJUST	Image Adjust	1	Enhancement	No
20.	THRESH	Thresholding	1	Post-processing	No

TABLE 4. LIST OF MORPHOLOGICAL STRUCTURING ELEMENTS

No	Name	SHAPE
1.	SE1	Disk
2.	SE2	Octagon
3.	SE3	Diamond
4.	SE4	Square
5.	SE5	Rectangle
6.	SE6, SE7	Pair
7.	SE8, SE9	Line
8.	SE10	Periodicline

4.3.4 FITNESS EVALUATION

Fitness is the measure of the optimality of a program present in the population. It reflects the accuracy of the segmentation algorithm. The sum of the absolute errors made by a program for all the pixels of all the images of a training set can be transformed into a fitness function using some scaling techniques [37].

In our case, a segmented image consists of positive (object) and negative (non-object) pixels. Ideally the segmentation of an image would result in an output image where positive pixels cover object pixels perfectly and nothing else while negative pixels cover non-object pixels perfectly and nothing else. Based on this ideal, we can view segmentation as a pixel-classification problem. Thus, the task of the segmentation program becomes assignment of the right class to every pixel in the image. As such, we can apply measure of classification accuracy to the problem of image segmentation.

Every segmentation program can be expected to identify not only pixels belonging to the objects of interest (True Positives, TPs), but also some non-object pixels identified as objects (False Negatives, FNs). Further, in addition to identifying non-object pixels (True Negatives, TNs), some pixels belonging to non-objects can be identified as object pixels (False Positives, FPs). Therefore for an ideal segmentation, the number of FPs and FNs should be zero while the number of TPs and TNs should be exactly equal to number of object and non-object pixels. If we normalize the value of TPs and TNs by the total number of object and non-object pixels respectively, their individual values in the best case scenario would be 1 and

0 in the worst case scenario. However, for the segmentation problem, achieving this is a challenging task, thus we define two more measures based on TPs, TNs, FPs and FNs called the False Positive Rate (FPR) and False Negative rate (FNR). FPR is the proportion of negative instances that were erroneously reported as being positive while FNR is the proportion of positive instances that were erroneously reported as negative.

$$False\ Positive\ Rate\ (FPR) = \frac{Number\ of\ False\ Positives\ (FP)}{Total\ number\ of\ Negative\ instances\ (FP + TN)}$$

$$False\ Negative\ Rate\ (FNR) = \frac{Number\ of\ False\ Negatives\ (FN)}{Total\ number\ of\ Positive\ instances\ (FN + TP)}$$

Therefore, for an ideal segmentation, the values of FPR and FNR should be zero. For finding accuracy of a segmentation program, we use a pixel-based accuracy formula based on FPR and FNR. This formula reflects the training and validation accuracy for GPIS. It is as follows:

$$\boxed{Accuracy = k * (1 - FPR) * (1 - FNR)}$$

where *FPR* - False Positive Rate,

FNR - False Negative Rate, and

k is a constant.

The value of *k* is calculated as follows:

$$k = 4 * Wp * (1 - Wn)$$

where W_p – Weight for False Positives, $W_p \in [0, 0.5]$

W_n – Weight for False Negatives, $W_n = 1 - W_p$

The weights W_p and W_n are used as a measure to balance the FPR/FNR trade-off.

The above formula for accuracy extends image segmentation problem to a pixel-classification problem. Therefore, ideally value of accuracy should be 1 (or 100%) for a perfectly segmented image. We also see that the formula is mono-modal i.e. if *image A* is better segmented than *image B* $\Rightarrow$ *Accuracy (A)* > *Accuracy (B)*

However, we further extend this formula by introducing a term that penalizes longer programs. Therefore, the fitness function for GPIS is as follows:

$$\boxed{Fitness = Accuracy * \left(\frac{1}{e^{\beta * len}}\right)}$$

where:

len = Length of the program

β – Scaling factor for the length of a program, $\beta \in [0.004, 0.008]$

As seen above, the length of the evolved program is fused into the fitness function and is represented by the last term in the equation. It acts as a means to keep a check on the length of the programs. We have observed that by doing so, natural evolution promotes shorter and fitter programs. β is a scaling factor for the length and its value lies between 0.004 and 0.008. Optimally, a value of 0.005 is sufficient.

The above approach of representing the image segmentation problem as a pixel-classification problem can be summarized as shown below in Figure 7.

		ACTUAL CONDITION	
		OBJECT PIXELS	**NON-OBJECT PIXELS**
EVOLVED SEGMENTATION RESULTS	**POSITIVE (OBJECT PIXELS)**	True Positive	**False Positive** (Non-object pixels segmented as cell pixels) **TYPE I ERROR** ↓ **Specificity = 1-FPR**
	NEGATIVE (NON-OBJECT PIXELS)	**False Negative** (Cell pixels segmented as non-cell pixels) **TYPE II ERROR** ↓ **Sensitivity = 1-FNR**	True Negative

FIGURE 7. SEGMENTATION AS A PIXEL CLASSIFICATION PROBLEM

4.3.5 SELECTION AND ELITISM

The selection operation involves selecting chromosomes from the population to undergo some form of diversification or surviving a generation. In either case, there are more chances of a fitter chromosome getting selected. In addition to the selection methods, elitism is a measure to retain a certain percentage of the best available programs in the population pool.

In our case, we use *elitism* as a means of saving the top 1% chromosomes of a population. Copies of the best 1% of the chromosomes in the population are copied without change to the next generation. Since there is always a possibility that the best chromosomes available might be lost or get destroyed by crossover or mutation, elitism ensures that the best 1% of chromosomes always survive a generation.

In order to select chromosome for diversification (*parent* chromosomes), we use a *tournament selection* scheme. It is chosen instead of rank selection as it is computationally more efficient. The size of the tournament window λ is kept as 10% of the size of the population. The quantity of parents selected is 50% of the size of the population. The 50% that undergo some form of diversification (crossover or mutation) survive the generation and are sent to the population of the next generation. The remaining 49% come in the form of *offspring* chromosomes produced by diversification.

In case, an injection is used at the end of the particular generation, a *survivor selection* mechanism is applied to build the new population. Injection inserts 20% new randomly generated chromosomes to the population. In order to balance this, the top 79% of the parent and offspring chromosomes are selected based on fitness and added to the injected pool.

In either case, 99% of the population is produced via selection and diversification and the remaining 1% comes from the elite set.

4.3.6 DIVERSIFICATION: GENETIC OPERATORS

Genetic operators define what type of jumps a system can make through a search space. We employ one crossover and four mutation operators. These are selected probabilistically based on their respective rate of crossover and mutation.

When a parent chromosome is selected for diversification, it can undergo either crossover or mutation. If crossover is chosen, it waits till the time another chromosome is chosen for crossover. If mutation is chosen, it undergoes one of the four mutations.

4.3.6.1 CROSSOVER

Crossover is typically a two parent genetic operator. It works by exchanging the "genetic material" between two parent chromosomes.

We have used a 1-point crossover for our GP. Two parents are chosen randomly from the parent pool. A random location is chosen in each of the parent chromosomes. The subsequences before and after this location in the parents are exchanged creating two offspring chromosomes, as shown in Figure 6.

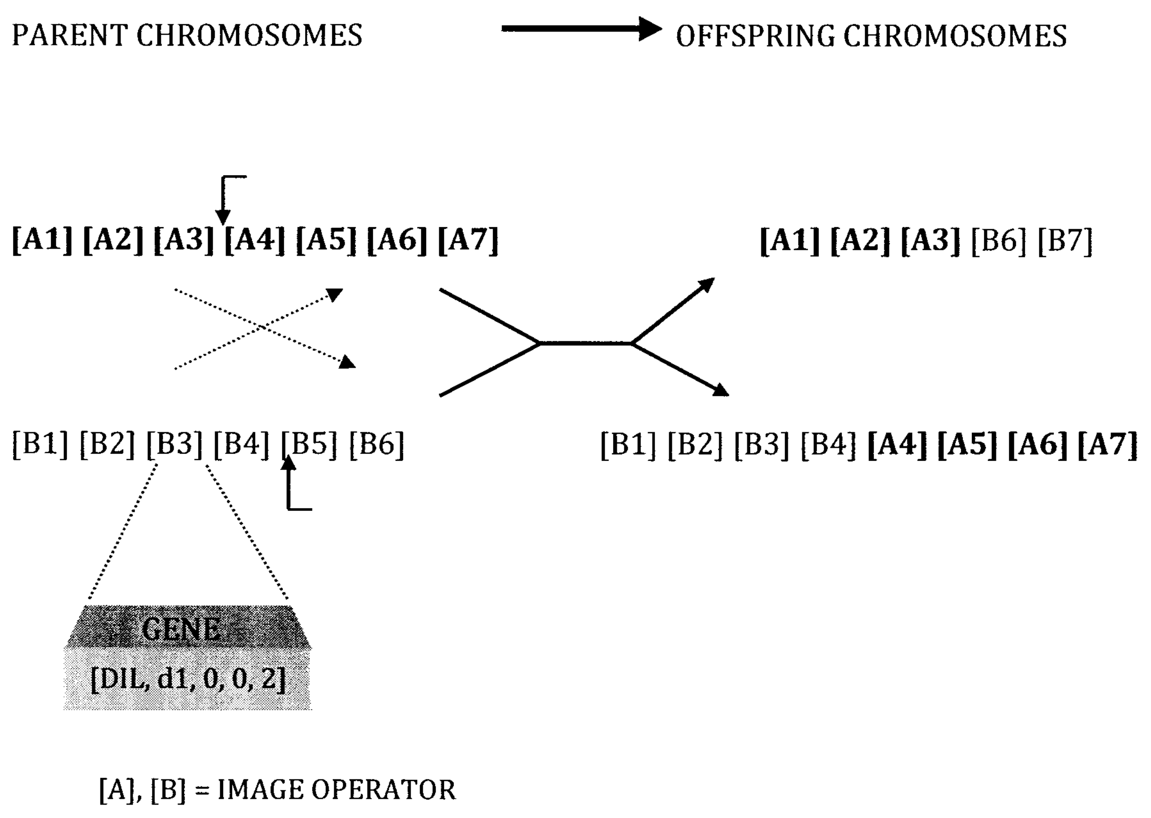

PARENT CHROMOSOMES
OFFSPRING CHROMOSOMES
[A1] [A2] [A3] [A4] [A5] [A6] [A7]
[A1] [A2] [A3] [B6] [B7]
[B1] [B2] [B3] [B4] [B5] [B6]
[B1] [B2] [B3] [B4] [A4] [A5] [A6] [A7]
GENE
[DIL, d1, 0, 0, 2]
[A], [B] = IMAGE OPERATOR

FIGURE 8. CROSSOVER SCHEME: ONE-POINT CROSSOVER

4.3.6.2 MUTATION

Mutation is a one parent genetic operator. It is applied to a single chromosome at a time and makes (small) changes in the genetic code of an individual.

We have used four mutations operators. They can be divided into two categories, Type A (inter-genomic - *swap, insert* and *delete*) and Type B (intra-genomic - *alter*).

TYPE A MUTATION:

Type A mutation is typically inter-genomic. There are three inter-genomic mutations used - *swap, insert* and *delete*. The three inter-genomic mutations are as follows:

a) *Swap*: Two random locations inside a chromosome are chosen and the respective genes are swapped.
b) *Insert*: A new gene is inserted in a randomly chosen position inside a parent chromosome.
c) *Delete*: A gene at a randomly chosen location gets removed from the chromosome. The remaining chromosome joins back together at the point of the deletion.

TYPE B MUTATION:

Alter: Type B mutation is intra-genomic. *Alter* is a fitness based mutation. It is only performed if the fitness of the parent chromosome is above a *minimum threshold* value. If so, one of the genes inside the chromosome is chosen randomly and it undergoes this mutation. Typically, the *weight* bit of the gene is altered based on the type of image operator. If however, the chromosome's fitness is below the *minimum threshold* fitness, the parent is discarded back to the parent pool and no mutation takes place. This operation essentially performs parameter tuning for the primitive image operators. The threshold was set at 70% accuracy, based on results of trial runs.

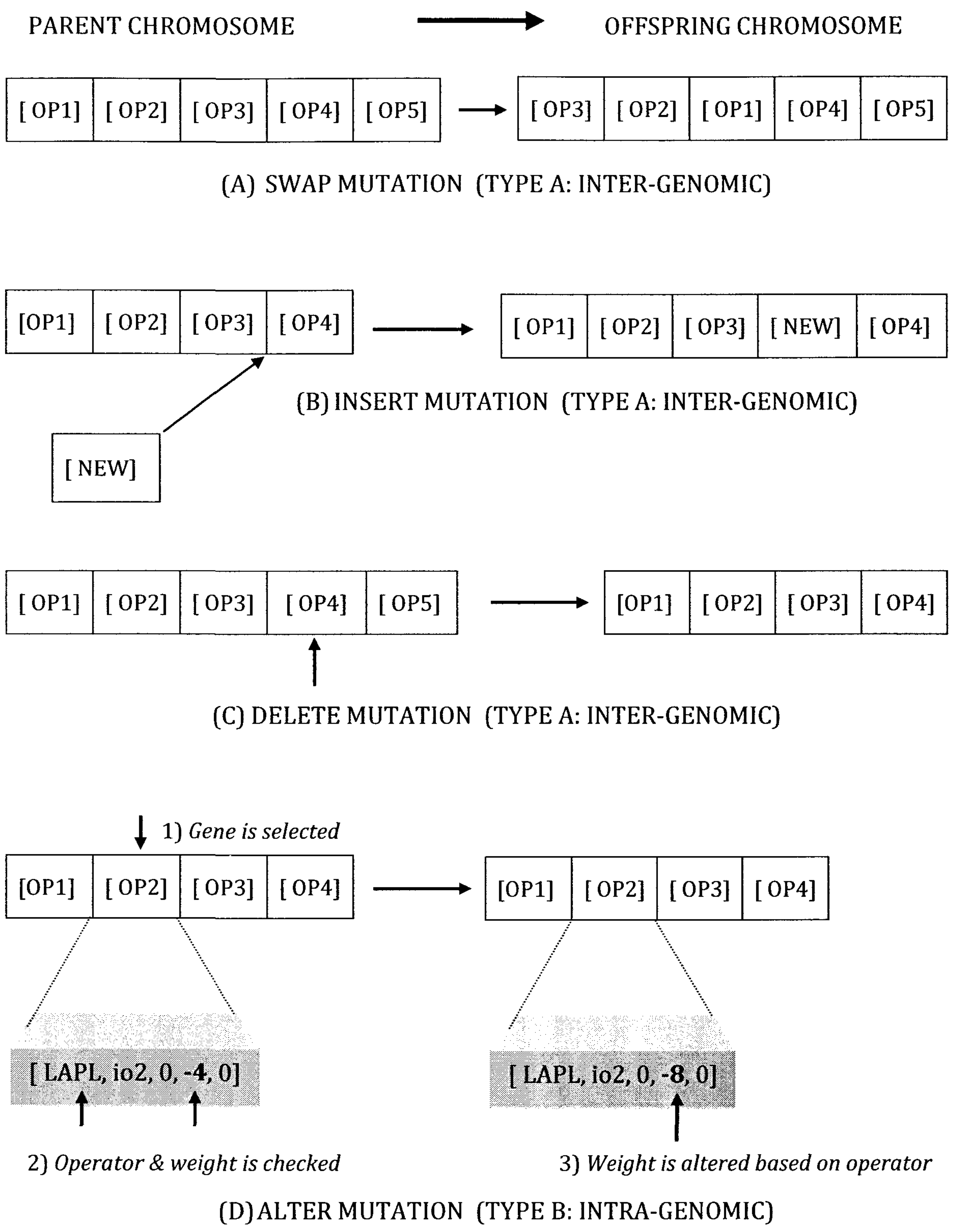

PARENT CHROMOSOME
OFFSPRING CHROMOSOME
[OP1] [OP2] [OP3] [OP4] [OP5]
[OP3] [OP2] [OP1] [OP4] [OP5]
(A) SWAP MUTATION (TYPE A: INTER-GENOMIC)
[OP1] [OP2] [OP3] [OP4]
[NEW]
[OP1] [OP2] [OP3] [NEW] [OP4]
(B) INSERT MUTATION (TYPE A: INTER-GENOMIC)
[OP1] [OP2] [OP3] [OP4] [OP5]
[OP1] [OP2] [OP3] [OP4]
(C) DELETE MUTATION (TYPE A: INTER-GENOMIC)
1) Gene is selected
[OP1] [OP2] [OP3] [OP4]
[OP1] [OP2] [OP3] [OP4]
[LAPL, io2, 0, -4, 0]
[LAPL, io2, 0, -8, 0]
2) Operator & weight is checked
3) Weight is altered based on operator
(D) ALTER MUTATION (TYPE B: INTRA-GENOMIC)

FIGURE 9. MUTATION SCHEMES

4.3.7 INJECTION

Traditional EAs often suffer from loss of diversity through premature convergence of its population. This causes the search to get trapped in local optima and creation of *too many similar individuals* in a population. Thus, the maintenance of diversity in the population is one of the most fundamental issues for any EA. We use an injection mechanism to overcome this problem and provide an option of introducing a fixed percentage of new randomly initialized programs to the population after every *n* generation. In the current configuration, we inject 20% new programs after every 5 generations. However, GPIS provides the user with an option to vary values of either.

4.3.8 TERMINATION

Termination of the GP is purely fitness based and the evolutionary cycle continues till the time there is no major change in fitness over a couple of generations. In order to do this, first we calculated an acceptable fitness value based on our trial runs. This value was found to be 95% for database 1 (HeLa Cells) and 90% for database 2 (Liver Cells). Till the time, these values of fitness were not achieved, the GP kept running. Once, these values were reached, a method of calculating cumulative means of the fitness of successive generations was used. If the absolute difference between the means of 10 successive generations was less than 5% of the highest fitness achieved, the GP stops. If however, the GP is used on any other database, a default value of 90% is set.

4.4 SUMMARY

We propose an evolutionary algorithm based approach to image segmentation. We have developed an algorithm; *Genetic Programming based Image Segmentation* (*GPIS*) in this regard. This chapter details the approach, the software architecture, explanation of the inherent elements and working of the proposed algorithm.

GPIS is an image segmentation tool. It uses a GP based framework for its working. It follows the typical *training-validation* model of EAs. *Training* includes evolution of a population of image segmentation programs. These programs are represented as a *chromosome* with a linear structure. Each chromosome contains a sequence of *genes* which are typically image analysis operators or *primitive image operators*. Using principles of natural selection and reproduction (diversification), *fitter* and more accurate programs are evolved. Selection mechanism consists of parent selection by means of *Tournament Selection* as well as a combination of *fitness based* (injection) and *steady state* survivor selection (no injection - all parents and offspring survive). In order to perform diversification, *crossover* (one-point) and *mutation* (swap, insert, delete and alter) operations are done. Fitness of a program is based on its accuracy to segment a set of training images. It is typically based on the *False Positive Rate* and the *False Negative Rate* of the segmentation produced. Once optimum programs have been discovered, they are applied on new images known as *validation* images. If the segmentation accuracy is similar to the training accuracy, the program is saved as a *solution* segmentation program. It order to achieve a collection of such programs, the above model is repeated multiple times to verify the authenticity of the approach. The following table summarizes the elements of the approach.

TABLE 5. SUMMARY OF GPIS APPROACH

Genetic Programming based Image Segmentation (GPIS)	
Problem/Goal	Evolution of accurate Image Segmentation algorithms
Type of EA used	GP
Representation	Linear
Terminal Set	Pool of primitive image analysis operators (20 in number)
Function Set	Data/InterOut Planes
Parent Selection	Tournament Selection (λ = 10% of population size)
Survivor Selection	Steady-state (No injection) Fitness-based (Injection)
Elitism	1%
Injection	20% every 5 generations
Genetic Operators	Crossover and Mutation
Crossover	One-point
Mutation	Swap, Insert, Delete (Inter-genomic) Alter (Intra-genomic)
Fitness	FPR, FNR based
Termination	Fitness based

This section is followed by Part B of the chapter which provides the experimental set up for the algorithm.

PART B: EXPERIMENTAL SETUP

This section provides details of the practical setup for GPIS.

4.5 IMAGE DATABASES

Various experiments (runs) were performed to test the efficacy of our GP in segmenting regions of interest from the given cell databases. We tested our GP on two medical databases. These were typically cell databases but significantly different in nature as seen below in figure 8. Corresponding ground truth has been manually hand-segmented.

The *first database* consisted of 1026 images of HeLa cell images (in culture). They were available in three magnifications, 10×, 20× and 40×, of dimensions $512 \times 384\,pixels^2$. The task of the GP for this database was to find effective cell segmentation algorithms to extract the cells present in the image.

The *second database* consisted of images from liver tissue specimen of mouse. There were 120 images in total of dimensions $340 \times 780\,pixels^2$. The task of the GP for this database was to segment the nuclei seen in the images.

(a) DATABASE 1 - HeLa CELLS

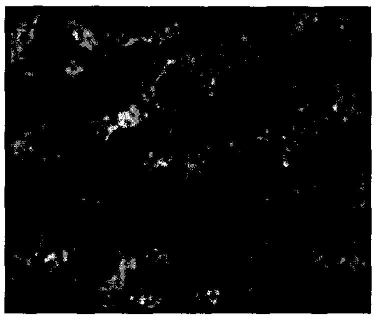

(b) DATABASE 2 - LIVER CELLS

FIGURE 10. SAMPLES FROM IMAGE DATABASES

4.6 PROCEDURE FOR TRAINING AND VALIDATION

The first task in order to plan a GP run is to decide on the training and validation set. In order to do so, we define the following terms:

G – Global total number of images in a database

T – Training set (number of images used for training)

V – Validation set (number of images used for validation)

R – Number of times optimal individuals are evolved for the same database.

The values for G, T, V and R for both databases are shown in Table 6.

TABLE 6. FINAL VALUES OF G, T, V, R FOR BOTH DATABASES

PARAMETER	DATABASE 1 (HeLa CELLS)	DATABASE 2 (LIVER CELLS)
G	1026	120
T	30	25
V	100	75
R	28	26

4.6.1 PROCEDURE FOR OBTAINING RESULTS FOR GPIS

1. Randomly select T images and other V images from the G images in the database.
2. Train on T images by providing corresponding ground truth.
3. Validate on V images to produce one optimal individual.

4. Repeat Steps 1 to 3, R times producing a set of optimal individuals (result set).
5. Calculate values of average training and validation accuracy of the result set.
6. Conduct the required statistical analysis (central tendency, divergence and upper-lower bounds).

4.6.2 PROCEDURE FOR OBTAINING RESULTS FOR GENIE PRO

1. Select the same T and V images from the G images in the database, used for corresponding GPIS run.
2. Load each of the T images as a *base image* and create a *training overlay* for each image by marking *Foreground* (object) and *Background* (non-object) pixels manually.
3. Train on these manually marked training overlays using the in-built Ifrit Pixel Classifier till there is no change in displayed estimated accuracy.
4. Apply learned solution on V images to produce corresponding segmented images.
5. Calculate validation accuracy for these V images using GPIS accuracy formula (Section 4.3.4)
6. Repeat Steps 1 to 5, R times like GPIS.

4.7 PARAMETER SETTINGS FOR GPIS

It is hard to quantify an optimal set of parameter values for any EA. We found the best way to find optimal parameter setting is to conduct trial runs focusing on one parameter at a time. Primary parameter values were determined using this strategy. Once, primary values were determined for all parameters, some fine tuning was performed and final parameter values were set. The parameter values that consistently gave the best results were used for the final runs. The parameter values used for both databases are shown in Table 7 and 8 respectively. In addition, an optimum set of parameter values is shown in Table 9. These values can be used as a starting point while using GPIS for other databases.

TABLE 7. SYSTEM PARAMETERS SETTINGS FOR DATABASE 1

Total number of images in database: G	1026
Training set: T	30
Validation set: V	100
Number of runs: R	28
Population size: μ	200
Crossover Rate: P_c	0.45
Swap Mutation Rate:P_{ms}	0.25
Insert Mutation Rate: P_{mi}	0.25
Delete Mutation Rate: P_{md}	0.2
Alter Mutation Rate: P_{ma}	0.7
Weight constant: k	0.9216
Scalability factor for length: β (Fitness Function)	0.005

TABLE 8. SYSTEM PARAMETERS SETTINGS FOR DATABASE 2

Total number of images in database: G	120
Training set: T	25
Validation set: V	75
Number of runs: R	26
Population size: μ	200
Crossover Rate: P_c	0.5
Swap Mutation Rate:P_{ms}	0.3
Insert Mutation Rate: P_{mi}	0.3
Delete Mutation Rate: P_{md}	0.25
Alter Mutation Rate: P_{ma}	0.7
Weight constant: k (Fitness Function)	0.9216
Scalability factor for length: β (Fitness Function)	0.005

TABLE 9. RECOMMENDED PARAMETER VALUES FOR GPIS

Population size: μ	200
Crossover Rate: P_c	0.45
Swap Mutation Rate:P_{ms}	0.25
Insert Mutation Rate: P_{mi}	0.25
Delete Mutation Rate: P_{md}	0.2
Alter Mutation Rate: P_{ma}	0.7
Weight Constant: k (Fitness Function)	0.9216
Scalability factor for length: β (Fitness Function)	0.005

CHAPTER 5: RESULTS AND ANALYSIS

We have proposed a GP based algorithm for Image Segmentation, GPIS. This chapter provides experimental results for GPIS as well as a comparison with GENIE Pro. We have tested our algorithm on 2 image databases, HeLa Cells and Liver Tissue Cells (C57BL/6 mouse). The task of the GP for Database 1 (HeLa Cells) was to extract complete cell structures present in the images while for Database 2 (Liver Tissue), it was required to extract the nuclei present in the images.

We have based our results on two criteria, *effectiveness* of the algorithm to correctly and accurately segment the given images, and *efficiency* of the algorithm in doing so. Effectiveness is based on two measures, pixel accuracy of the evolved solution and the percentage of structures (cell count) correctly identified. In order to calculate the cell density measure, we have categorized cells into two types: Type1 and 2. Type 1 cells are those which can be identified by eye with relative ease. Type 2 cells are those which are relatively difficult to be identified by eye. Their number can vary based on the database in question. Efficiency is also based on two measures, number of fitness evaluations (generations) taken to converge to an acceptable solution, and the relationship between length of an evolved program and its execution time. Efficiency based on number of fitness evaluations is provided for each database in Section 5.1.2 and 5.2.2 while efficiency measured by length of program is presented in Section 5.3 as it is not dependent on nature of the database.

For comparison of performance, we have used a well accepted GA based automatic image segmentation/classification tool GENIE Pro (details of GENIE Pro are provided in Chapter 2, section 2.4). For comparing accuracy of the evolved solutions in either algorithm, we provide fitness values recorded over the respective runs. We also compare the output image from GENIE Pro based on our fitness function. This provides a common ground for comparing pixel-level accuracy. We use a stricter fitness function as compared to GENIE Pro.

The result tables reflect the substantive performance of our algorithm and are supported by evolved solutions and image results obtained.

The chapter is concluded with a reflective analysis of the algorithm's performance on both databases.

5.1 RESULTS FOR DATABASE 1 –HELA CELLS

The task of the algorithm for Database 1 was to extract cell structures from the given cell images.

5.1.1 EFFECTIVENESS

A. <u>PIXEL-BASED PERFORMANCE:</u>

Table 10 provides a comparison of the training and validation accuracies achieved by GPIS and Genie Pro for Database 1. These reflect the capability of an evolved program to correctly segment each pixel in an image as cell and non-cell. Thus, they are the pixel-based accuracy values of the segmentations produced by the fittest evolved programs for each algorithm, over 28 runs of each (average of segmentation accuracy of the fittest program of every run). It can be seen here that the segmentation accuracy achieved by GPIS was higher than GENIE Pro.

TABLE 10. SEGMENTATION ACCURACY: GPIS Vs GENIE PRO (DATABASE 1)

ALGORITHM	TRAINING DATA	VALIDATION DATA
GPIS	98.76%	97.01%
GENIE PRO	94.12%	93.12%

Table 11 provides the statistical results for GPIS based on validation *accuracy* of segmentations produced by the evolved solutions. These are averaged values based on results from 28 runs.

TABLE 11. STATISTICAL RESULTS FOR GPIS: SEGMENTATION ACCURACY (DATABASE 1)

STATISTICAL MEASURE		ACCURACY
CENTRAL TENDENCY	MEAN	97.01%
	MEDIAN	97.02%
DISPERSION/ BOUNDS	STANDARD DEVIATION	1.84
	UPPER BOUND	99.31%
	LOWER BOUND	93.69%

B. <u>CELL COUNT BASED PERFORMANCE:</u>

Table 12 reflects each algorithm's ability to detect the cell structures. It was seen that GPIS had a higher rate of detection as compared to GENIE Pro. In order to calculate the above, cells were counted by hand in the evolved solutions.

TABLE 12. CELL COUNT: GPIS Vs GENIE PRO (DATABASE 1)

CELL COUNT MEASURE	GPIS		GENIE PRO	
	TRAINING DATA	VALIDATION DATA	TRAINING DATA	VALIDATION DATA
Detected Cells	98.24%	97.98%	97.02%	96.56%
(i) Type 1 Cells	100%	100%	100%	100%
(ii) Type 2 Cells	98.78%	98.22%	97.49%	96.89%
Undetected Cells	1.32%	1.55%	2.12%	2.25%

5.1.2 EFFICIENCY (NUMBER OF FITNESS EVALUATIONS)

This section provides details of the efficacy of GPIS for Database 1 over 28 runs conducted. Table 13 provides details of the number of generations and fitness evaluations needed for an optimal solution to be produced. Table 14 provides statistical results for GPIS based on efficiency (number of generations/fitness evaluations).

TABLE 13. NUMBER OF GENERATIONS AND FITNESS EVALUATIONS FOR GPIS (DATABASE 1)

	NUMBER OF GENERATIONS	NO OF FITNESS EVALUATIONS
BEST (HIGHEST FITNESS)	114	10,532
AVERAGE	122.07	11,257.67

TABLE 14. STATISTICAL RESULTS FOR GPIS: NUMBER OF GENERATIONS AND FITNESS EVALUATIONS (DATABASE 1)

STATISTICAL MEASURE		**NUMBER OF GENERATIONS**	**NUMBER OF FITNESS EVALUATIONS**
CENTRAL TENDENCY	**MEAN**	122.07	11,257.67
	MEDIAN	122	11,107
DISPERSION/ BOUNDS	**STANDARD DEVIATION**	6.85	639.88
	UPPER BOUND	138	12,658
	LOWER BOUND	112	10,510

5.1.3 INTUITIVE REFLECTION: EVOLVED PROGRAMS

This section details two evolved programs for Database 1. The chromosomal and genealogical structure is provided below

EVOLVED PROGRAM 1:

GAUSSIAN FILTER	AVERAGING FILTER	IMAGE ERODE	AVERAGING FILTER	IMAGE CL-OP	THRESHOLDING

[GAUSS, d1, 0, 6, 0.8435] [AVER, io1, 0, 4, 0] [EROD, io2, 0, 0, 1]

[AVER, io3, 0, 6, 0] [CLOP, io4, 0, 0, 1] [THRESH, io5, 0, 0.09022, 0]

FIGURE 11. CHROMOSOMAL AND GENE STRUCTURE OF PROGRAM 1

Number of operators used = 6

Fitness on validation set = 99.04%

Execution time for program = 1.252 seconds

The program evolved above is a combination of filters and morphological operators. The first gene is a 6 × 6 *Gaussian* low pass filter with a sigma value of 0.8435 followed by a 4 ×4 *Averaging* filter. The output plane from gene 2 is *eroded* with a flat, *disk*-shaped structuring element of radius 2 (note: 1 denotes type of structuring element and not the radius here, refer Table 4 for details). A 6 × 6 *averaging* filter is again applied to the output plane of the eroded image. Its output plane undergoes a composite morphological operation of *closing* and *opening* with the same structuring element as above. Finally this plane is thresholded at 0.09022 to provide the final output image. As seen here, the GP was able to evolve a valid program and also optimize crucial parameter values. A validation implementation of the program is shown in Figure 13. The superimposed input-evolved image is shown below (Figure 12).

FIGURE 12. SUPERIMPOSED INPUT-EVOLVED IMAGE (EVOLVED PROGRAM 1)

ACCURACY = 99.02%

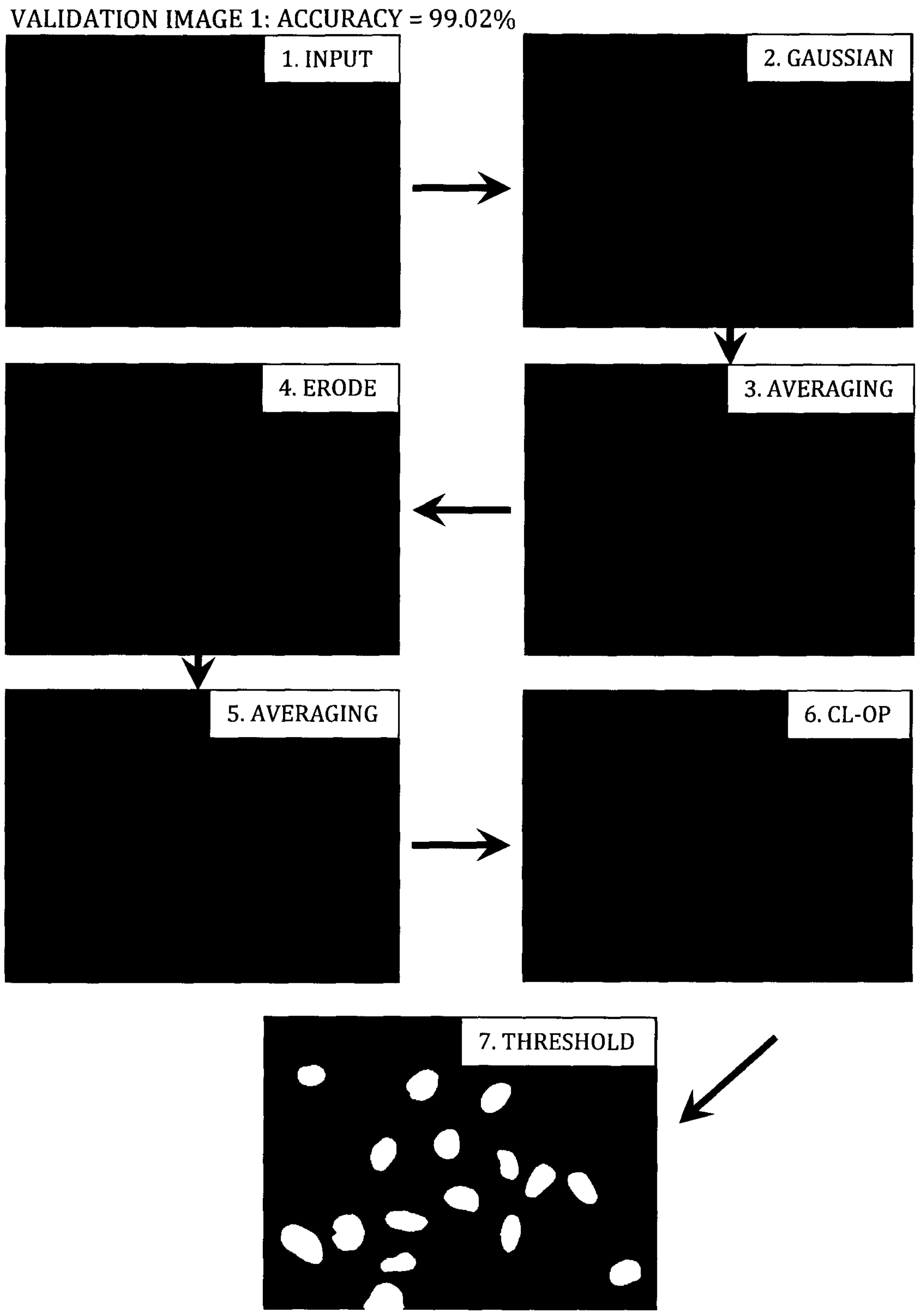


FIGURE 13. STEP-BY-STEP IMAGE EVOLUTION OF EVOLVED PROGRAM 1

EVOLVED PROGRAM 2:

DISK FILTER	AVERAGING FILTER	IMAGE CLOSE	ADD PLANES	IMAGE ERODE	IMAGE ERODE	THRESHOLDING

[DISK, d1, 0, 3, 0] [AVER, io1, 0, 6, 0] [CLOSE, io2, 0, 0, 2] [ADDP, io1, io2, 0, 0]

[EROD, io3, 0, 1] [EROD, io4, 0, 1] [THRESH, io5, 0, 0.1264, 0]

FIGURE 14. CHROMOSOMAL AND GENE STRUCTURE OF PROGRAM 2

Number of operators used= 7

Accuracy on validation set = 99.31%

Execution time of program = 1.382 seconds

This program initializes itself with a 3 × 3 circular *Averaging filter* (pillbox - disk) followed by another 6 × 6 *Averaging filter.* The output plane undergoes image *Closing* with a flat, *octagonal* structuring element of radius 3. Then output planes from gene 1 and 2 are *added* and *eroded* using a flat, *disk* shaped structuring element of radius 2. Finally this output plane is *thresholded* using a threshold of 0.1264. The validation implementation of the program is shown below in Figure 16. The superimposed input-evolved image is shown below (Figure 15).

FIGURE 15. SUPERIMPOSED INPUT-EVOLVED IMAGE (EVOLVED PROGRAM 2)

ACCURACY = 99.31%

VALIDATION IMAGE 2: ACCURACY = 99.31%

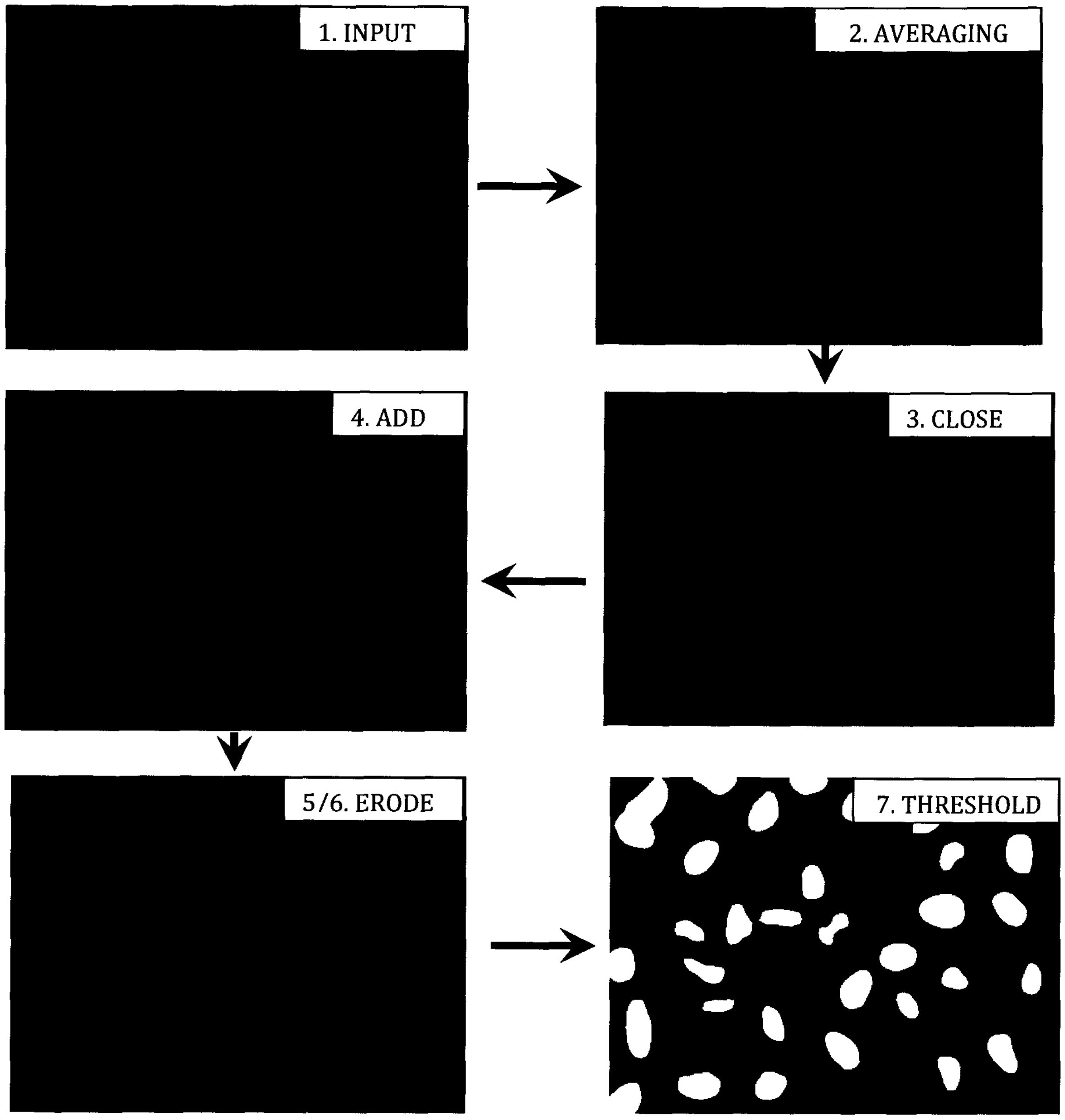


FIGURE 16. STEP-BY-STEP IMAGE EVOLUTION OF EVOLVED PROGRAM 2

5.2 RESULTS FOR DATABASE 2 – LIVER TISSUE SPECIMEN

The task of the algorithm for Database 2 was to extract nuclei structures from the given images. This database is relatively much tougher and complex as compared to Database 1. In some areas of the image, the inter- and outer-object regions don't have much in difference. These results are compiled based on 26 runs of the GP on Database 2.

5.2.1 EFFECTIVENESS

A. PIXEL-BASED PERFORMANCE:

Table 15 provides a comparison of the training and validation accuracies achieved by GPIS and Genie Pro for Database 2. These reflect the capability of an evolved program to correctly segment each pixel in an image as nuclei and non-nuclei. Thus, they are the pixel-based accuracy values of the segmentations produced by the fittest evolved programs for each algorithm, over 26 runs of each (average of segmentation accuracy of the fittest program of every run). It can be seen here that the segmentation accuracy achieved by GPIS was higher than GENIE Pro.

TABLE 15. SEGMENTATION ACCURACY: GPIS Vs GENIE PRO (DATABASE 2)

ALGORITHM	TRAINING DATA	VALIDATION DATA
GPIS	94.38%	91.12%
GENIE PRO	86.22%	84.04%

Table 16 provides the statistical results for GPIS based on validation *accuracy* of segmentations produced by the evolved solutions. These are averaged values based on results from 26 runs.

TABLE 16. STATISTICAL RESULTS FOR GPIS: SEGMENTATION ACCURACY (DATABASE 2)

STATISTICAL MEASURE		**FITNESS**
CENTRAL TENDENCY	**MEAN**	91.12%
	MEDIAN	91.42%
DISPERSION/ BOUNDS	**STANDARD DEVIATION**	2.08
	UPPER BOUND	94.82%
	LOWER BOUND	88.25%

B. CELL COUNT BASED PERFORMANCE:

Table 17 reflects each algorithm's ability to detect the nuclei structures. It was seen that GPIS had a higher rate of detection as compared to GENIE Pro. In order to calculate the above, segmented nuclei were counted by hand in the evolved solutions.

TABLE 17. CELL (NUCLEI) COUNT: GPIS Vs GENIE PRO (DATABASE 2)

CELL COUNT MEASURE (NUCLEI)	GPIS		GENIE PRO	
	TRAINING DATA	VALIDATION DATA	TRAINING DATA	VALIDATION DATA
Detected Cells	92.18%	89.98%	89.36%	87.42%
(i) Type 1 Cells	98.89%	97.12%	96.23%	95.01%
(ii) Type 2 Cells	91.82%	90.23%	90.02%	86.78%
Undetected Cells	3.87%	4.65%	4.78%	5.39%

5.2.2 EFFICIENCY (NUMBER OF FITNESS EVALUATIONS)

This section provides details of the efficiency of GPIS for Database 2 over 26 runs conducted. Table 18 provides details of the number of generations and fitness evaluations needed for an optimal solution to be produced. Table 19 provides statistical results for GPIS based on efficiency (number of generations/fitness evaluations).

TABLE 18. NUMBER OF GENERATIONS AND FITNESS EVALUATIONS FOR GPIS (DATABASE 2)

	NUMBER OF GENERATIONS	NO OF FITNESS EVALUATIONS
BEST (HIGHEST FITNESS)	206	18,732
AVERAGE	214.15	19,563.87

TABLE 19. STATISTICAL RESULTS FOR GPIS: NUMBER OF GENERATIONS AND FITNESS EVALUATIONS (DATABASE 2)

STATISTICAL MEASURE		NUMBER OF GENERATIONS	NUMBER OF FITNESS EVALUATIONS
CENTRAL TENDENCY	MEAN	214.15	19,397.31
	MEDIAN	216	19,636
DISPERSION/ BOUNDS	STANDARD DEVIATION	7.19	674.95
	UPPER BOUND	224	20,452
	LOWER BOUND	202	18,280

5.2.3 INTUITIVE REFLECTION: EVOLVED PROGRAMS

EVOLVED PROGRAM 1:

LOWPASS FILTER	**AVERAGING FILTER**	**AVERAGING FILTER**	**IMAGE ADJUST**	**IMAGE CLOSE**	**THRESHOLD**

[LOWPASS, d1, 0, 32, 0.793] [AVER, io1, 0, 4, 0] [AVER, io2, 0, 3, 0]

[ADJUST, io3, 0, .205, 0.517] [CLOSE, io4, 0, 0, 1] [THRESH, io5, 0, 0.9852, 0]

FIGURE 17. CHROMOSOMAL AND GENE STRUCTURE OF PROGRAM 1

Number of operators used = 6

Accuracy on validation set = 94.68%

Execution time of evolved program = 0.8410 seconds

The program evolved above displays the capacity of the GP to optimize parameter values once a stable structure has been evolved. The input image undergoes a *low-pass* filtering action with a filter of the order N = 32 and a cut-off frequency of 0.793. The output plane undergoes two successive averaging operations by a *4 × 4* and *3 × 3 Averaging filter*. The resultant output plane undergoes an image enhancement operation where the intensity of the pixel planes is *mapped* to new values such that the intensity values between 0 and 1 are now mapped to 0.205 and 0.517 (intensity transformation of grayscale). From an image processing point of view, this operation increases the contrast of the image. This mapped plane now undergoes an *image closing* action with a flat, *disk* shaped structuring element of radius 2. Finally the resulting image undergoes *thresholding* (0.9852). The most impressive part of the GP here is maintaining a short program size with losing accuracy. A step-by-step processing of a validation image is shown below in Figure 18 as well as the superimposed input-evolved result (Figure 19).

VALIDATION IMAGE 1: ACCURACY = 94.68%

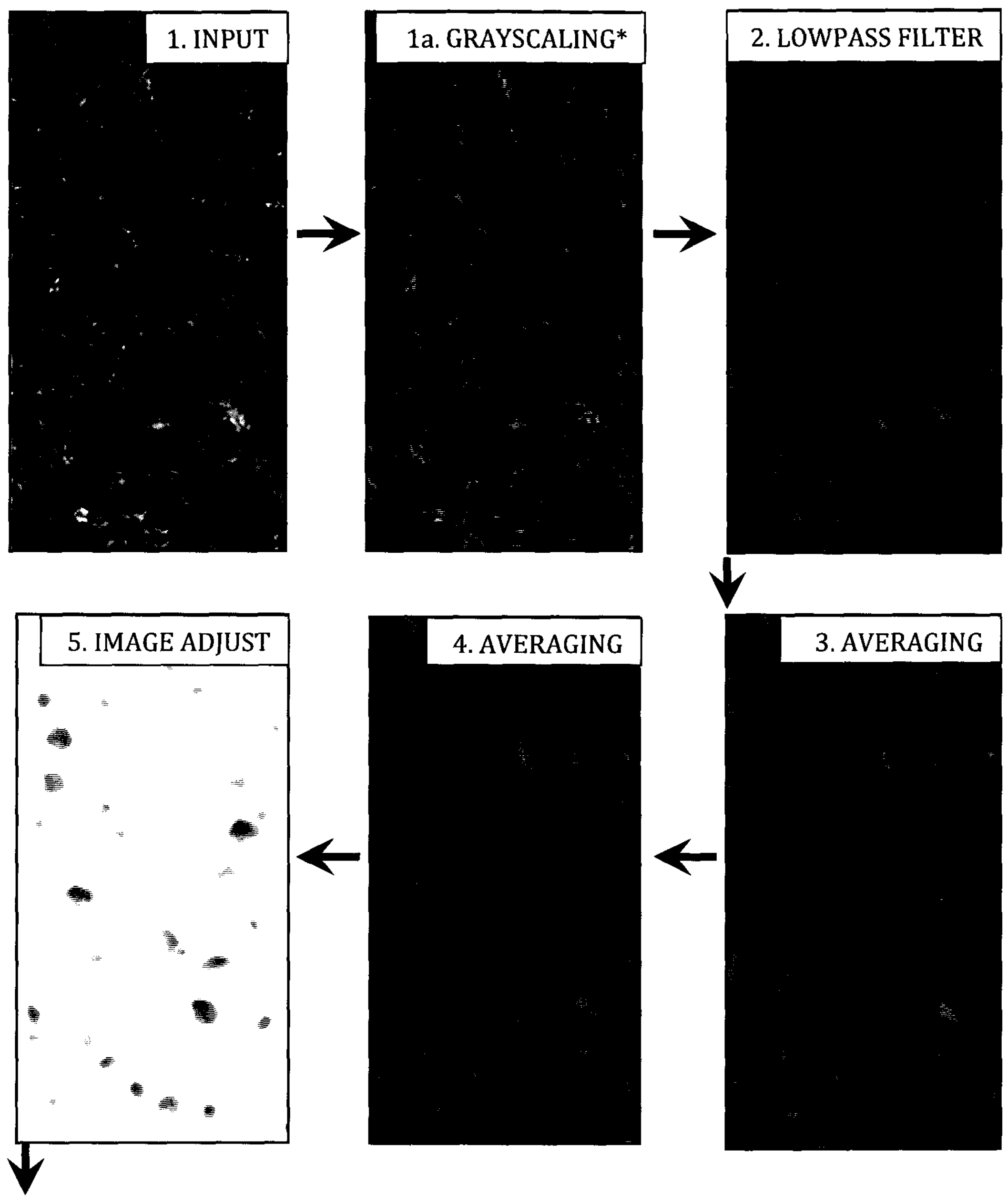


FIGURE 18a. STEP-BY-STEP IMAGE EVOLUTION – VALIDATION IMAGE 1

(EVOLVED PROGRAM 1) * denotes operation done

CONT: NEXT PAGE

CONT:

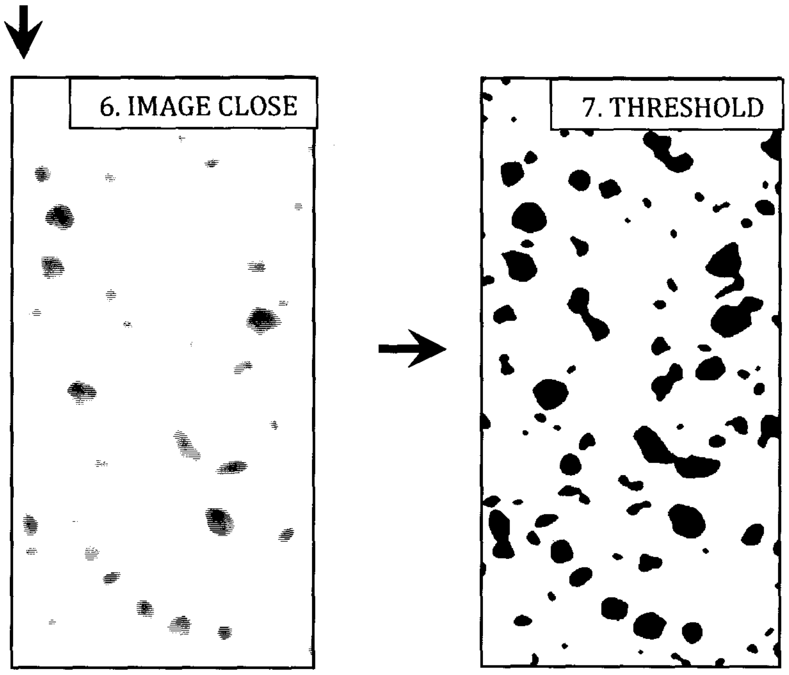


FIGURE 18b. STEP-BY-STEP IMAGE EVOLUTION – VALIDATION IMAGE 1 (EVOLVED PROGRAM 1)

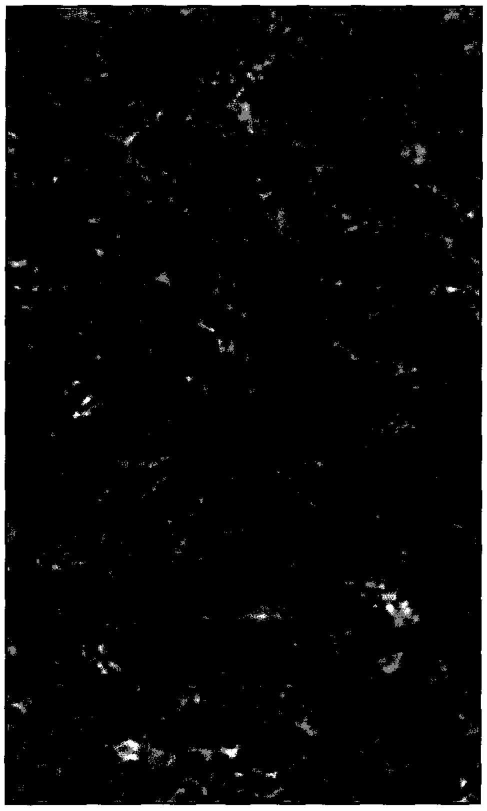


FIGURE 19. SUPERIMPOSED INPUT-EVOLVED IMAGE (EVOLVED PROGRAM 1)

ACCURACY = 94.68%

EVOLVED PROGRAM 2:

UNSHARP FILTER	HIST. EQ.	LAP. FILTER	DISK FILTER	AVERAG. FILTER	HIST. EQ.	ADJUST	OPEN	ERODE	THRESH

[UNSHARP, d1, 0, 0.82, 0] [HISTEQ, io4, 0, 0, 0] [LAPL, io1, 0, -8, 0]

[DISK, io2, 0, 6, 0] [AVER, io3, 0, 6, 0] [HISTEQ, io4, 0, 0, 0]

[ADJUST, io5, 0, 0, 0.202] [OPEN, io6, 0, 0, 1] [ERODE, io7, 0, 0, 1]

[THRESH, io8, 0, 0.752, 0]

FIGURE 20. CHROMOSOMAL AND GENE STRUCTURE OF PROGRAM 2

Number of operators used = 10

Accuracy on validation set = 94.98%

Execution Time of evolved program = 1.2153 seconds

The evolved program shown above has 10 operators. The first operator is a 3 × 3 *unsharp* contrast enhancement filter with an alpha value of 0.82. Its output plane undergoes histogram equalization. It is followed by a 3 × 3 *Laplacian* filter (Laplacian of a Gaussian filter – [1,1,1; 1,-8,1; 1,1,1]). This is followed by a 6 × 6 circular *disk* and *averaging* filtering operation. Their output plane undergoes histogram equalization followed by a grayscale *intensity transformation* between 0 and 0.202. The resulting plane undergoes image *opening* and *erosion* operation with a flat, *disk* shape structuring element of radius 2. Finally the plane is thresholded at a threshold of 0.752 to produce the final image.

VALIDATION IMAGE 2: ACCURACY = 94.98%

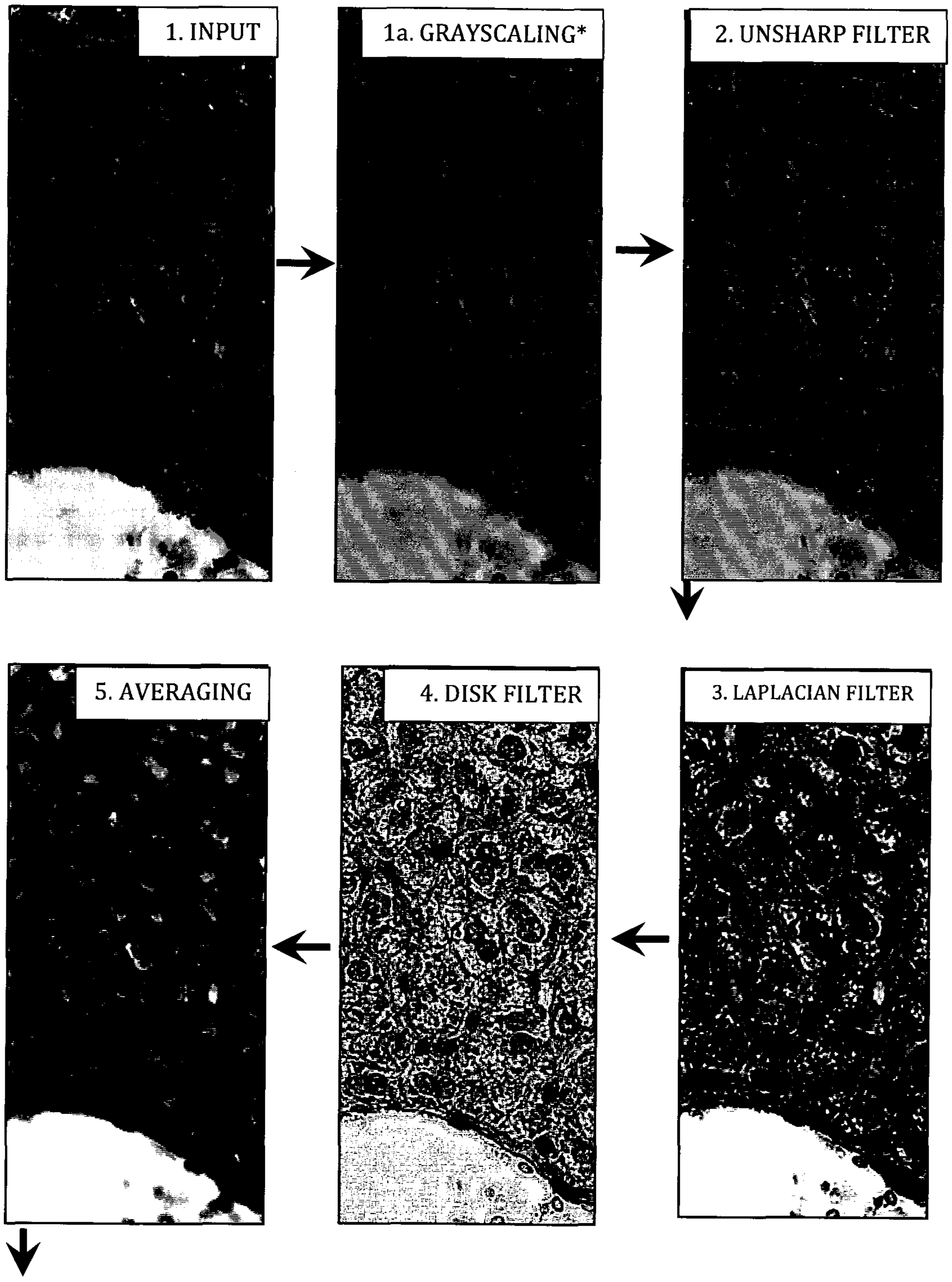


FIGURE 21a. STEP-BY-STEP IMAGE EVOLUTION – VALIDATION IMAGE 2

CONT: NEXT (EVOLVED PROGRAM 2) * denotes operation done

CONT:

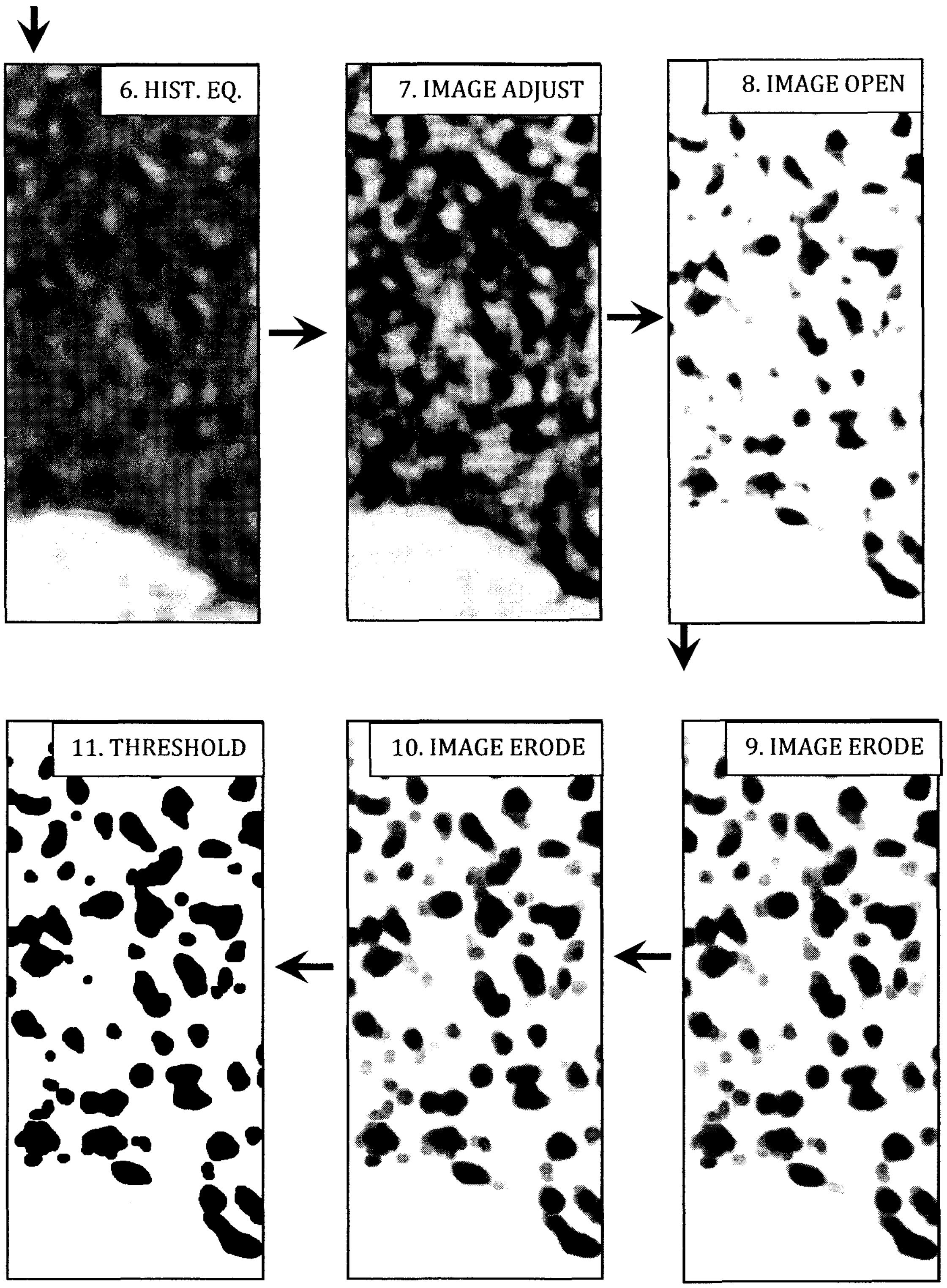


FIGURE 21b. STEP-BY-STEP IMAGE EVOLUTION – VALIDATION IMAGE 2

(EVOLVED PROGRAM 2)

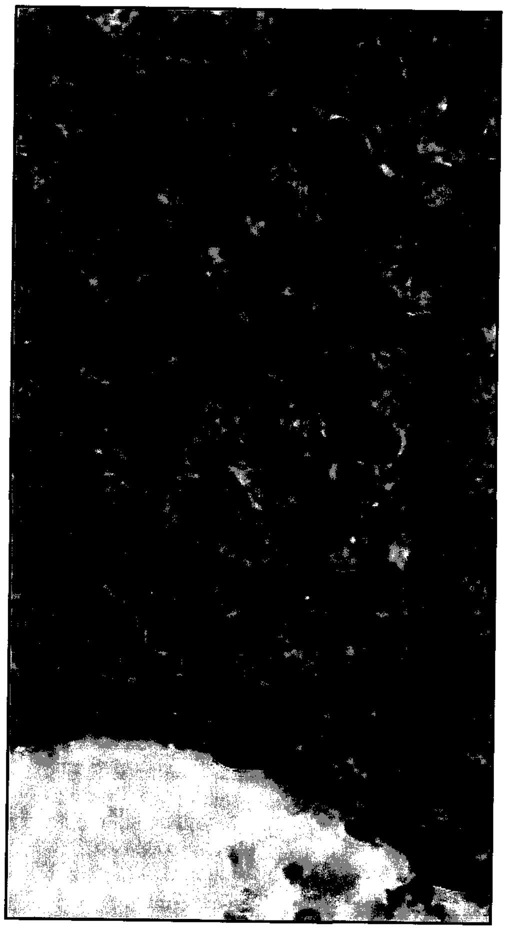


FIGURE 22. SUPERIMPOSED INPUT-EVOLVED IMAGE (EVOLVED PROGRAM 2)

ACCURACY = 94.98%

5.3 EFFICIENCY: EXECUTION TIME Vs LENGTH OF PROGRAM

The efficacy of GPIS is also measured by comparing length of an evolved program with the time it takes to execute the program. Figure 23 and 24 provide graphs for this comparison. These graphs are built using the evolved programs from combined runs of GPIS on both the databases i.e. 54 runs.

The graph in Figure 23 is a plot between execution time and length of the evolved program for the *fittest programs* produced in the runs. There were in total 54 runs of GPIS on both the databases, therefore, this graph compares execution speed of the 54 fittest programs evolved. The size of these programs was between 6 to 12 operators and their execution time was between 0.8312 seconds to 2.6104 seconds.

The graph in Figure 24 is a plot between execution time and length of evolved program other than the fittest programs produced during the runs. 4 programs with average fitness (above 70%) were chosen from the final population of each run; therefore, this graph compares execution speed of the 216 evolved programs. The size of these programs was between 5 to 20 operators and their execution time was between 0.5322 seconds to 4.2352 seconds.

It can be seen from both the graphs that the execution speed of the evolved programs is linearly correlated with their lengths, therefore, shorter the program, faster the execution.

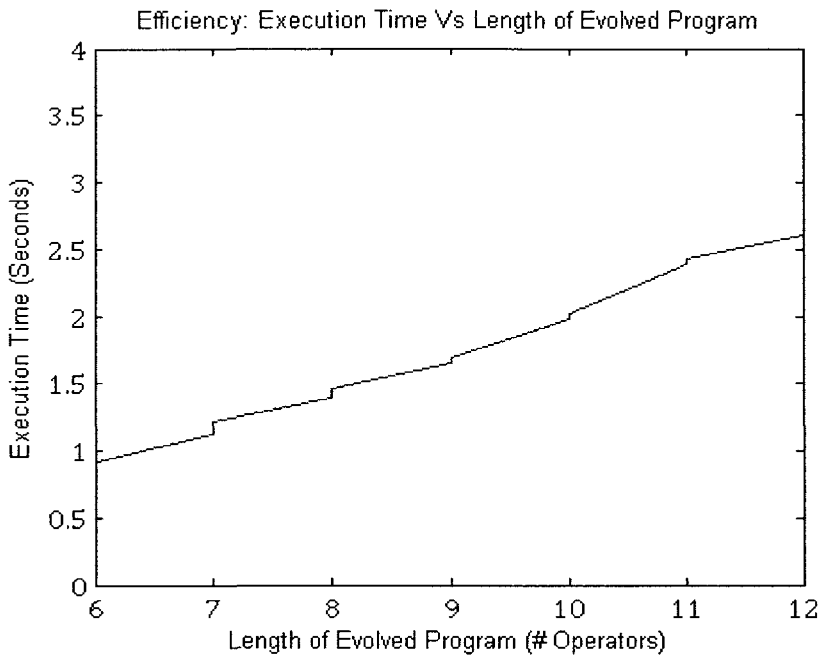


FIGURE 23. GRAPH 1 - EXECUTION TIME Vs LENGTH OF PROGRAM (FITTEST PROGRAMS)

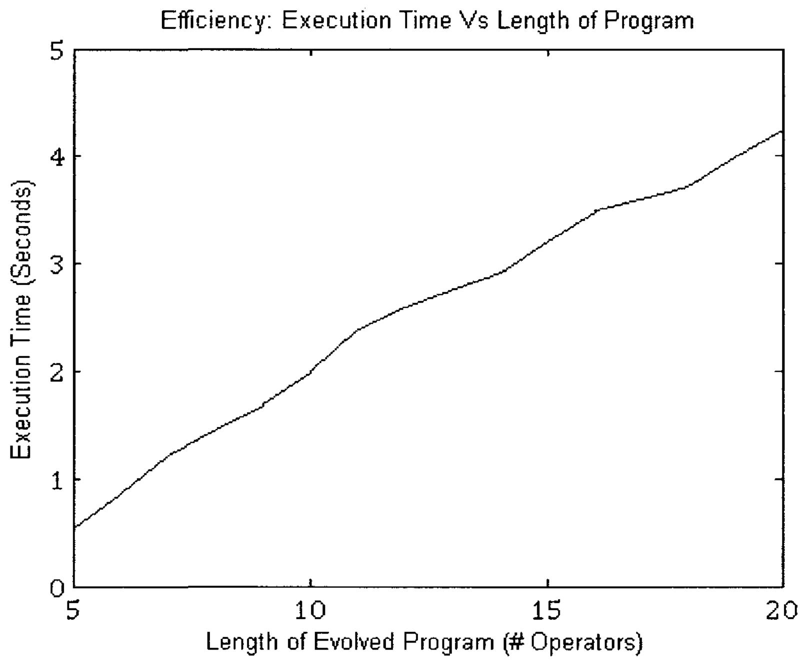


FIGURE 24. GRAPH 2 - EXECUTION TIME Vs LENGTH OF PROGRAM (PROGRAMS WITH FITNESS > 70%)

5.4 REFLECTIVE ANALYSIS OF RESULTS

We have proposed an algorithm for image segmentation based on genetic programming, *GPIS* and presented the experimental results obtained in this chapter. We also provided a comparison with a well accepted GA-based image segmentation/classification tool, GENIE Pro and observed that GPIS performed much better, both in comparison with pixel-accuracy as well as cell detection rate. We present a reflective analysis of the results to conclude this chapter.

GPIS works on a training-validation model; therefore, it requires a set of ground truths while training. Similar to [4] and [51], we implemented a training window which allowed the GP to choose the relative size of the training image. Reflected by the distinct parameter optimization seen in the values of operator thresholds and weights, we believe a training window approach allowed refinement of operators and that lead to further sharpening of parameter values. In addition, the training window approach is less computationally intensive and aids to increase the speed of the evolutionary cycle.

Secondly, it is evident that including a factor for length of the evolved program in the fitness function encourages the GP to produce shorter and more optimal programs. As seen from the evolved programs presented earlier in the chapter, the average length of an optimal solution was between 6 to 11 operators. Although it's hard to compare algorithms based on operator length, thus far this has been the lowest operator-length evolved for an efficient image segmentation algorithm.

A significantly important observation can be made based on the structure of the evolved programs seen for the two databases. While mostly all evolved solutions for Database 1 worked on color images, it was seen in the evolved programs for Database 2 that the GP was able to adapt based on the images at hand and most of the processing took place in a grayscaled mode. This shows the GP's capacity to adapt to two different modes of image processing.

5.4.1 ANALYSIS OF RESULTS FOR DATABASE 1 (HELA CELLS)

The task required from the GP for Database 1 was to extract cell structures from the given images. This could be considered as a small object extraction problem. The nature of the problem in this database was relatively simpler as compared to Database 2 and it was evident in results also. The GP had an overall segmentation value of 97.01% over 28 runs and the best evolved individual produced a fitness of 99.31% over a validation set of 100 images. It took the GP 114 generations and 10,532 fitness evaluations to produce the best individual. At an average, the GP took around 122 generations and approximately 11,257 fitness evaluations to produce optimal individuals. In comparison to GENIE Pro, GPIS performed better, both in regards to fitness achieved as well as cell detection rate. The corresponding fitness achieved by GENIE Pro was 95.5%. As far as cell detection rate was concerned, GPIS was able to detect 98.98% of the cell structures present in the image as compared to

96.56% detected by GENIE Pro. It was observed that the evolved programs for this database used 6-9 operators in order to perform the segmentation. Most of the processing of images through the evolved programs took place in color mode. These results are based on 28 runs of the GP on Database 1.

5.4.2 ANALYSIS OF RESULTS FOR DATABASE 2 (LIVER CELL SPECIMEN)

The task of the GP for Database 2 was to extract the nuclei seen in the images. This was a high complexity task based on the nature of the images in question. Understandably, the GP took longer and more fitness evaluations to evolve optimal individuals. The overall segmentation accuracy achieved by the GP for this database was 91.12% and the best individual produced a fitness of 94.82% over 75 validation images. It took the GP 206 generations and 18,732 fitness evaluations to produce the best individual. At an average, it took the GP around 214 generations and approximately 19,562 fitness evaluations to produce optimal results. As compared to GENIE Pro, GPIS performed better on this database also. GENIE Pro produced an overall fitness of 85.80%. As far as the cell detection rate is concerned, due to the nature of images, the detection rate was slightly lower as compared to Database 1. The overall cell detection rate for GPIS was 89.98% as compared to 87.42% for GENIE Pro. The general length of the evolved programs for Database 2 was between 6 and 11. Most of the processing of images through the evolved programs took place in grayscaled format. These results are based on 26 runs of the GP on Database 2.

CONCLUSION AND FUTURE WORK

An automated image segmentation system, GPIS, has been described. In the experiments conducted, results were compared to a well accepted GA based image segmentation/classification tool, GENIE Pro and GPIS consistently delivered better results.

At the outset, we set out to investigate the role of genetic programming as a viable automatic programming tool for image segmentation. Image segmentation is a deceptively difficult problem but of great practical importance. Thus, a possible automatic segmentation tool is always desirable. In the process, we used a primitive-operator based approach to image segmentation as every segmentation algorithm could be broken down into a series of basic image analysis functions or *primitive-operators*. We use GP to synthesize these primitive operators to segment potential objects in images.

Our experimental results indicate the effectiveness of our approach as well as the sufficiency of our primitive operator pool. Further conclusions made are as follows:

6.1 CONCLUSIONS

1. The GP was able to distinguish the complexity of the images provided to it. It evolved shorter programs for easier images as compared to difficult images.
2. Extendibility of a primitive operator based approach is dependent on the validity and sufficiency of the operators. If it is desired to build a general purpose image segmentation tool using primitive operators, it is best to use basic, low-level image analysis operators. However, if such an approach needs to be implemented for a particular type of images, certain specialized operators can be added to the pool of primitives. Based on the variety of images we tested our GP on, we found our operator pool was sufficient and effective.
3. We encouraged evolution of shorter and fitter programs. Based on a scalability factor included in our fitness function, we found the GP was able to recognize the reward based on length and effectiveness and evolved shorter, accurate programs.
4. Although it is hard to compare effectiveness based on length of program with respect to past works [37, 47], we found the relative length of our programs to be the shortest amongst similar works done in the past. Since we implemented the algorithm on MATLAB, it is easy for an interested observer to base a comparison on length with our work. The evolved solutions are in a *what you see is what you get* format, thus, readily implementable.

5. We also found injection as a reliable means of maintaining population diversity. We felt adding a 20% new randomly initialized population every 5 generations helped the search from slipping into a local optima. None of the runs revealed the search getting trapped into local optima.
6. A *training window approach* during training phase proved to be very effective for operator refinement. Since training in this mode allows for using sections of an image instead of the entire image, sharp parameter tuning is possible. This can be observed especially in thresholding operations. In addition, this approach greatly improves computational time required for training.
7. We also found that if a small but accurate set of ground truths is provided, the GP can produce accurate segmentation algorithms without inclusion of any a priori spatial or textural information of the images.

6.2 FUTURE WORK

Many observations were made during the course of this thesis. In our future work, we hope to investigate performance of the GPIS algorithm on other image databases, medical and non-medical. Most of the segmentation techniques available are application dependent. We have tried to make GPIS as application independent as possible. The nature of images in the two databases used was considerably different. Based on the performance of GPIS on these databases, we feel that it should be able to perform well on other databases also. As a prelude to this observation, we conducted preliminary testing on some such images. The results were promising and indicate further generalization of GPIS. Some of the results are enclosed at the end of this chapter.

Secondly, we hope to include *automatically defined functions* into the GP architecture. If portions of the chromosome for fitter individuals contain certain repeated operators, those portions can be extracted and can qualify as ADFs. This would lead to an accumulation of ADFs during the progression of the evolutionary cycle. These ADFs can be then added to fit and relatively fit individuals in the population as a means of improving their fitness. Based on their acceptance, ADFs could then be ranked and can later be used as operator sub-routines.

Thirdly, use of competitive co-evolution can offer a different perspective to assigning fitness to individuals. In this case, individual fitness of programs would be evaluated through competition with other programs in the population, rather than through an absolute fitness measure. Fitness in this scenario would signify the relative strengths of programs; an increased fitness for one program would lead to a decreased fitness for another.

Finally, we hope to add conditional jumps (IF, THEN, ELSE, CASE, SWITCH statements) into the function set. This would further improve its sufficiency.

We hope that the above would allow further generalization of GPIS.

6.3 PRELIMINARY TESTING OF GPIS ON OTHER IMAGES

We tested performance of GPIS on other images apart from the two databases used. The following are some preliminary results. The first two images are from the Vision and Autonomous Systems Center's Image Database at Carnegie Mellon University and the other two are from the internet. In the first set of images, we performed two different tasks on the same image. The first task was lane detection and the second task was tree detection. This can be seen in Figure 24 (a) and (b). The second task was to extract intra-cellular content from Wright Stained White Blood Cells. These preliminary results can be seen in Figure 25, (a) and (b). Ground truth for all four images was prepared by hand.

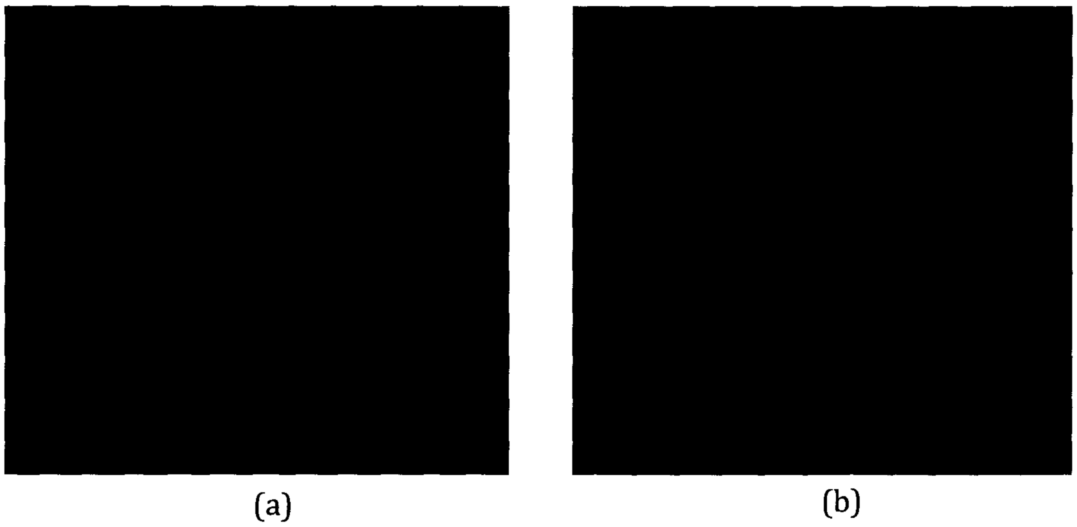


FIGURE 24. PRELIMINARY RESULTS FOR: (a) LANE DETECTION (b) TREE DETECTION

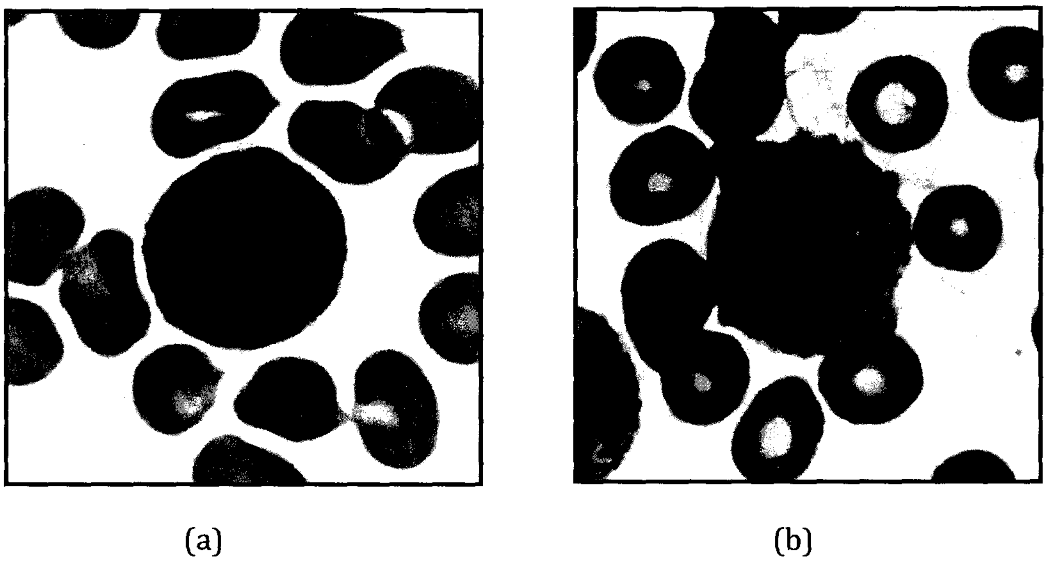


FIGURE 25. PRELIMINARY RESULTS FOR EXTRACTING INTRA-CELLULAR CONTENT OF WRIGHT STAINED WHITE BLOOD CELL IMAGES

LIST OF REFERENCES

[1] D. Agnelli, A. Bollini, and L., Lombardi, "*Image Classification: An Evolutionary Approach*", Pattern Recognition Letters, 23 (1-3), 2002, pp. 303-309.

[2] 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.* San Francisco, CA. Morgan Kaufmann Publishers; Heidelburg: Dpunkt-verlag, 1998

[3] T. Belpaeme, "*Evolution of Visual Feature Detectors*", Proceedings of 1st Conference on Evolutionary Computation in Image Analysis and Signal Processing, 1999, pp. 1–10.

[4] B. Bhanu and Y. Lin, "*Object Detection in Multi-modal Images using Genetic Programming*", Applied Soft Computing, 4 (2), 2004, pp. 175-201.

[5] B. Bhanu, Y. Lin, "*Learning Composite Operators for Object Detection*", Proceedings of the Conference on Genetic and Evolutionary Computation, July 2002, pp. 1003–1010.

[6] S. P. Brumby, J. P. Theiler, S. J. Perkins, N. R. Harvey, J. J. Szymanski, and J. J. Bloch, "*Investigation of Image Feature Extraction by a Genetic Algorithm*", Proceedings of SPIE, Vol. 3812, 1999, pp. 24-31.

[7] S. P. Brumby, S. Koch, and L. A. Hansen, "*Evolutionary Computation and Post-wildfire Land-cover Mapping with Multispectral Imagery*", Proceedings of SPIE, Vol. 4545, 2002, pp. 174-183.

[8] S. P. Brumby, N. R. Harvey, J. J. Bloch, J. P. Theiler, S. J. Perkins, A. C. Young, and J. J. Szymanski, "*Evolving Forest Fire Burn Severity Classification Algorithms for Multispectral Imagery*", Proceedings of SPIE, Vol. 4381, 2001, pp. 236-245.

[9] S. Cagnoni, and R. Poli, Editorial, Special issue of the EURASIP Journal of Applied Signal Processing on Genetic and Evolutionary Computation for Signal Processing and Image Analysis, April 2003.

[10] J. M. Daida, J. D. Hommes, T. F. Bersano-Begey,S. J. Ross, and J. F. Vesecky, *"Algorithm Discovery using the Genetic Programming Paradigm: Extracting Low-contrast Curvilinear Features from SAR Images of Arctic Ice"*, Advances in Genetic Programming II, P. J. Angeline, K. E. Kinnear, (Eds.), Chapter 21, The MIT Press, 1996, pp. 417-442.

[11] J. M. Daida, J. D. Hommes, S. J. Ross, A. D. Marshall, and J. F. Vesecky, *"Extracting Curvilinear Features from SAR Images of Arctic Ice: Algorithm Discovery Using the Genetic Programming Paradigm,"* Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, Italy, IEEE Press, 1995, pp. 673–75.

[12] L. B. Dorini , R. Minetto, and N. J. Leite, *"White Blood Cell Segmentation using Morphological Operators and Scale-space Analysis"*, Proceedings of the Twentieth Brazilian Symposium on Computer Graphics and Image Processing, 2007, pp. 294-304.

[13] K. S. Fu, and J. K. Mui, *"A Survey on Image Segmentation"*, Pattern Recognition, 13, 1981, pp. 3-16.

[14] P. Ghosh and M. Mitchell, *"Segmentation of Medical Images using a Genetic Algorithm"*, Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 1171—1178.

[15] R. C. Gonzalez, and R. E. Woods. *Digital Image Processing*, NJ: Prentice Hall, 2002.

[16] C. Harris and B. Buxton, *"Evolving Edge Detectors using Genetic Programming"*, Proceedings of the First Annual Conference on Genetic Programming, MIT Press, 1996, pp. 309-314.

[17] N. R. Harvey, S. Perkins, S. P. Brumby, J. Theiler, R. B. Porter, A. C. Young, A. K. Varghese, J. J. Szymanski, J. J. Bloch, *"Finding Golf Courses: The Ultra High Tech Approach"*, EvoWorkshops, 2000, pp. 54-64.

[18] N. R. Harvey, S. P. Brumby, S. Perkins, J. Theiler, J. J. Szymanski, J. Bloch, R. B. Porter, M. Galassi, A. C. Young *"Image Feature Extraction: GENIE Vs Conventional Supervised Classification Techniques"*, IEEE Transactions on Geoscience and Remote Sensing, Vol. 40, 2001, pp. 393-404.

[19] F. Herrera, M. Lozano, and A. M. Sanchez, "*A Taxonomy for Crossover Operator for Real-Coded Genetic Algorithms: An Experimental Study*", International Journal of Intelligent Systems, Vol. 18, 2003, pp. 309-338.

[20] D. Howard and S. C. Roberts, "*A Staged Genetic Programming Strategy for Image Analysis*", Proceedings of the Genetic and Evolutionary Computation Conference, 1999, pp. 1047—1052.

[21] D. Howard, S. C. Roberts, and R. Brankin, "*Evolution of Ship Detectors for Satellite SAR Imagery*", Proceedings of EuroGP'99, Vol. 1598, 1999, pp. 135-148.

[22] *Image Processing Toolbox User's Guide*, Version 5, The MathWorks, Inc, 2006

[23] K. Jiang, Q. Liao, and S. Dai, "*A Novel White Blood Cell Segmentation Scheme using Scale-space Filtering and Watershed Clustering*", Second International Conference on Machine Learning and Cybernetics, 2003, pp. 2820-2825.

[24] T. Jiang, F. Yang, Y. Fan, and D. J. Evans, "*A Parallel Genetic Algorithm for Cell Image Segmentation*", Electronic Notes in Theoretical Computer Science, Vol. 46, 2001.

[25] M. Johnson, P. Maes, and T. Darrell, "*Evolving Visual Routines*", Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV), 1994, pp. 198—209.

[26] K. Krawiec, D. Howard, and M. Zhang , "*Overview of Object Detection and Image Analysis by Means of Genetic Programming Techniques*", Frontiers in the Convergence of Bioscience and Information Technologies, 2007, pp. 779 - 784.

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

[28] B. Kumar and T. Sreenivas, "*Teager Energy based Blood Cell Segmentation*", International Conference on Digital Signal Processing, Vol. 2, 2002, pp. 619-622.

[29] S. Lazebnik, C. Schmid, and J. Ponce, "*Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories*", Proceedings of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, 2006, pp. 2169- 2178.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "*Gradient-based Learning applied to Document Recognition*", Intelligent Signal Processing, 2001, pp.

306–351.

[31] Q. Liao and D. Yingyang, *"An Accurate Segmentation Method for White Blood Cell Images"*, IEEE Symposium on Biomedical Imaging, 2002, pp. 245-248.

[32] N. R. Pal, and S. K. Pal, *"A Review on Image Segmentation Techniques"*, Pattern Recognition, 26, 1993, pp. 1277-1294.

[33] C. Pantofaru and M. Hebert, *"A Comparison of Image Segmentation Algorithms"*, Technical report CMU-RI-TR-05-40, Carnegie Mellon University, September, 2005.

[34] D. L. Pham, C. Xu, J. L. Prince, *"Survey of Current Methods in Medical Image Segmentation"*, Annual Review of Biomedical Eng, 2, 2000, pp. 315—337.

[35] S. J. Perkins, J. Theiler, S. P. Brumby, N. R. Harvey, R. B. Porter, J. J. Szymanski, J. J. Bloch, *"GENIE: A Hybrid Genetic Algorithm for Feature Classification in Multispectral Images"*, Proceedings of SPIE 4120, 2000, pp. 52-62.

[36] C. S. Plesko, S. P. Brumby, and C. B. Leovy, *"Automatic Feature Extraction for Panchromatic Mars Global Surveyor Mars Orbiter Camera Imagery"*, Proceedings of SPIE, Vol. 4480, 2002, pp. 139-146.

[37] R. Poli, *"Genetic Programming for Feature Detection and Image Segmentation"*, T.C. Forgarty (Ed.), Evolutionary Computation, Springer-Verlag, Berlin, Germany, 1996, pp. 110–125.

[38] R. Poli, W. B. Langdon, and N. F. McPhee (with contributions by J. R. Koza). *A Field Guide to Genetic Programming.* Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008, pp. 111.

[39] R. Poli, *"The Evolutionary Computation Cookbook: Recipes for Designing New Algorithms"*, Proceedings of the Second Online Workshop on Evolutionary Computation, 1996.

[40] M. E. Roberts and E. Claridge, *"An Artificially Evolved Vision System for Segmenting Skin Lesion Images"*, Proceedings of the 6th International Conference on Medical Image Computing and Computer-Assisted Intervention, Vol. 2878, 2003, pp. 655-
662.

[41] S. C. Roberts and D. Howard, "Evolution of Vehicle Detectors for Infrared Linescan Imagery", EvoIASP'99 and EuroEcTel'99, Vol. 1596, 1999, pp. 110-125.

[42] C Ruberto, A. Dempster, S. Khan, and B. Jarra, "*Segmentation of Blood Images using Morphological Operators*", International Conference on Pattern Recognition, 2000, pp. 3401-3405.

[43] J. C. Russ. *The Image Processing Handbook*, Third Edition, Chapter 6, CRC Press LLC, 1998.

[44] J. L. Semmlow, *Biosignal and Biomedical Image Processing, MATLAB-Based Applications*, Marcel Dekker, Inc, Chapter 12, 2004, pp. 343-374.

[45] A. Song, T. Loveard, and V. Ciesielski, "*Towards Genetic Programming for Texture Classification*", Proceedings of the 14th Australian Joint Conference on Artificial Intelligence, 2001, pp. 461–472.

[46] S.A. Stanhope and J.M. Daida, "*Genetic Programming for Automatic Target Classification and Recognition in Synthetic Aperture Radar Imagery*", Proceeding of the Seventh Conference on Evolutionary Programming, Springer-Verlag, Berlin, Germany, 1998, pp. 735–744.

[47] W. Tackett, "*Genetic Programming for Feature Discovery and Image Discrimination*", In S. Forrest, editor, Proceedings of 5th International Conference on Genetic Algorithm, 1993, pp. 303–311.

[48] T. Weise, *Global Optimization Algorithms – Theory and Application*. Published via http://www.it-weise.de, 2008, pp. 75.

[49] J. F. Winkeler and B. S. Manjunath, "*Genetic Programming for Object Detection*", Genetic Programming 1997: Proceedings of the Second Annual Conference, 1997, pp. 330-335.

[50] P. Winter, W. Yang, S. Sokhansanj, and H.Wood, "*Discrimination of Hard-to-pop Popcorn Kernels by Machine Vision and Neural Network*", ASAE/CSAE Meeting, Saskatoon, Canada, Sept. 1996, Paper No. MANSASK 96-107.

[51] M. Zhang, U. Bhowan, and B. Ny, "*Genetic Programming for Object Detection: A Two-Phase Approach with an Improved Fitness Function*", Electronic Letters on Computer Vision and Image Analysis 6(1), 2007, pp. 27-43.

[52] Y. J. Zhang, "*Influence of Segmentation over Feature Measurement*", Pattern Recognition Letters, 16(2), 1992, 201-206.