1-1-2009

Texture classification using gene expression programming

Fereshteh, Mahvarsayyad
Ryerson University

Texture Classification
Using Gene Expression Programming

By

Fereshteh Mahvarsayyad
BEng, Shiraz University, 1996

A MEng Project
presented to Ryerson University

In partial fulfillment of the
requirements for the degree of
Master of Engineering
in the Program of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2009

Declaration

I hereby declare that I am the sole author of this thesis or dissertation.

I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Abstract

In computer vision, segmentation refers to the process of subdividing a digital image into constituent regions with homogeneity in some image characteristics. Image segmentation is considered as a pre-processing step for object recognition. The problem of segmentation, being one of the most difficult tasks in image processing, gets more complicated in the presence of random textures in the image. This paper focuses on texture classification, which is defined as supervised texture segmentation with prior knowledge of textures in the image. We investigate a classification method using Gene Expression Programming (GEP). It is shown that GEP is capable of evolving accurate classifiers using simple arithmetic operations and direct pixel values without employing complicated feature extraction algorithms [5]. It is also shown that the accuracy of classification is related to the fact that GEP can detect the regularities of texture patterns. As part of this project, we implemented a Photoshop plug-in that uses the evolved classifiers to identify and select target textures in digital images.

Table of Contents

List of Tables

List of Figures

List of Appendices

1 Introduction

The goal of texture classification is to partition an unknown sample image into regions that belong to one of a set of known texture classes. Texture classification belongs to the wider problem domain of texture segmentation. In texture segmentation, the goal is to simplify an image into something that is more meaningful and easier to analyze. The image is divided into regions with homogeneity with respect to texture. Figure 1 shows some texture segmentation results. When texture segmentation is supervised and prior knowledge of textures in the image is available, the problem of texture segmentation is simplified to texture classification.

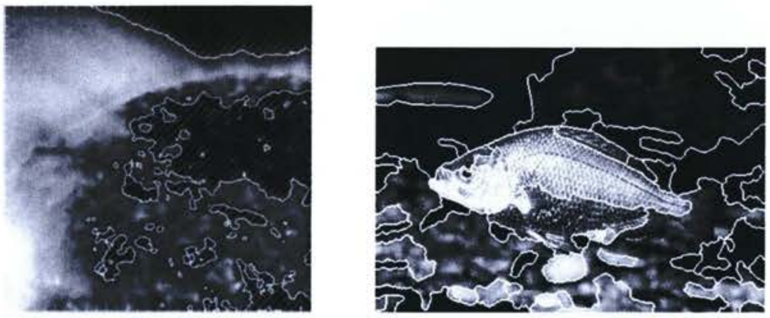


Figure 1: Texture segmentation examples[1]

[1] Image sources from left:
http://sidc.oma.be/EIT/OSTC2000/index.php
http://www.cs.washington.edu/homes/lachesis/images/classes/vision_final/report.html

Segmentation is spontaneous and instantaneous in human visual system. Humans can effortlessly partition a scene to foreground, background and objects. However, it is extremely difficult to mimic the same performance with an artificial algorithm. Some of the practical applications of image segmentation are: medical imaging, digital photography, robotic vision, object identification, scene analysis, criminology and security, geography (cartography), and multimedia access. Despite many potential applications in the industry, the problem of texture segmentation remains unsolved. This is mainly due to high computational complexity of many of the proposed texture extraction algorithms. In addition, textures in the real world are often not uniform due to changes in orientation, illumination, scale, and presence of environmental noise.

Reviewing the wide variety of human invented feature extraction algorithms is beyond the scope of this paper[2]. In this research, we explore the possibility of using GEP [4] to evolve feature extraction algorithms (hereinafter referred to as texture classifiers) out of simple arithmetic operations and direct pixel values.

First introduced by Candida Ferreira in 2001 [4], GEP is a recently developed evolutionary computation method for data analysis and knowledge discovery. Born from Genetic Algorithms (GAs) [2] and Genetic Programming (GP) [3], GEP is an adaptive search technique inspired by biological evolution. It uses procedures such as reproduction, mutation, recombination, natural selection, and survival of the fittest. GEP is both flexible at genetic operations due to its linear genotype and capable of retaining a certain extent of functional complexity due to its phenotype as expression trees. Previous research work has shown its powerful capabilities over a large range of domains. However, there has been little attention paid to using GEP for solving image processing problems. In this research, we extend the works of A. Song and V. Ciesielski [6]. We use GEP instead of GP to produce texture classifier algorithms. One of the main goals is to simplify and accelerate the evolution process.

[2] Refer to the work of Tuceryan and Jain [5] to read about feature extraction techniques.

The aim of this study is to answer the following questions:

- How to represent texture classifiers in GEP?
- Is GEP capable of evolving human competitive classifier algorithms?
- Are GEP-evolved classifiers fast and accurate enough for practical use?
- Is the developed technique suitable for real-time purposes? In other words, can GEP generate classifiers on the fly in an unsupervised texture segmentation application?
- Are regularities in the patterns detected by GEP?
- Does GEP show any advantages compared to GP?

To provide satisfactory answers to the above research questions, we developed a software application that uses GEP to generate classifiers for specific textures. The same application also provides means to test the produced classifiers for segmenting images. In order to prove the practicality of the method, we also implemented the segmentation algorithm as a Photoshop plug-in. Refer to Appendix A for more details on the developed software.

Our results show that GEP is able to generate fast and relatively accurate classifiers using main arithmetic operations as the function set and only pixel values as the terminal set. We exemplify real-time potentials of the developed technique by presenting the choices that can keep the evolution time between 5-20 seconds. We analyse a few simple classifiers that were generated by GEP for black and white textures and show how the regularities of the patterns is detected. We also highlight the advantages of using GEP as opposed to GP.

Next chapter provides details about our methodology. A summary of our experiments and the results is presented in chapter 3. Chapter 4 analyses the results and provides discussion about some of the choices we made to achieve the results. Finally, chapter 5 concludes the study and highlights some of the areas for further investigation and future work.

2 Methods and Materials

The hypothesis we put forward is that we can use GEP to detect texture patterns and generate fast and accurate classifier algorithms in real-time. We generally based our system on the technique presented in [6]; however, we replaced GP with GEP, took advantage of GEP features, and made a few simplifications to the base method to come up with a fast and nimble system. For our input data we used the textures from Brodatz album[3].

2.1 GEP System

There are two steps involved in the texture classification process: the learning step and the recognition step. In the learning phase, the goal is to build a model that describes each class of texture present in the training data. In the recognition phase, the texture content of the unknown sample is first described with the same texture description method as produced in the first step. The textural features of the sample are, then, compared to those of the training images and the sample is assigned to the category with the best match.

We developed a software application which implements both steps. Refer to Appendix A for details about the application.

The next two sections drill into the details of the abovementioned steps.

[3] Published in 1966, this collection has become a standard database in texture analysis studies: http://www.ux.uis.no/~tranden/brodatz.html

2.2 Learning Step - Generating Texture Classifiers

As the first step to the classification process, we use GEP to extract texture features and generate classifier algorithms. GEP individuals (chromosomes) are constructed from the function set and terminal set, as listed in Table 1 and Table 2 respectively.

Function	Description
+	Arithmetic addition
-	Arithmetic subtraction
*	Arithmetic multiplication
/	Arithmetic division

Table 1: Function Set

Name	Description
Pixel[x, y]	Pixel value at location (x, y)

Table 2: Terminal Set

Each individual is evaluated to a real number, which is then translated to class labels. This study focuses on binary classification, which means the output of a classifier is interpreted either as class A (target texture) or class B (non-target textures). There are several ways to divide the range of real numbers that are returned by the classifiers to represent different class labels. In this research, we examined static and dynamic range selection methods introduced by T. Loveard and V. Ciesielski [7].

In the static method, we divide the range of real numbers to negative and positive values and associate positive values with target texture (class A) and negative values with non-target textures (class B). In the dynamic method, however, we allow each chromosome to dynamically determine a separate set of ranges for class boundaries. In the latter, a subset of the training data is used to record values for each class and divide the infinite range of real numbers into regions corresponding to class boundaries [7].

We started with the full function set as proposed by Song and Ciesielski, which includes logical operations IF, <=, >=, =, and BETWEEN, as well as main arithmetic operations. We later reduced the function set to arithmetic operations only.

We used the simple definition for the fitness function as proposed in [6], which measures the success rate of each individual:

$f = \frac{\text{Number of Successes}}{\text{Total}} = \frac{\text{TP} + \text{TN}}{\text{Total}} \times 100\,\%$, where TP is the number of true positives, TN is the number of true negatives and Total is the total number of training cases.

The runs consisted of 50 generations with population size 256. We used single gene chromosomes with head length 500. The termination criteria were 100% correct classification or completion of the specified number of runs (typically 10) for 50 generations. Table 3 shows the rest of GEP parameters used in this study.

Parameter	Value
Number of generations	50
Population size	256
Gene head length	500
Number of genes	1
Linking function	N/A
Mutation rate	0.01
One-point recombination rate	0.2
Two point recombination rate	0.5
Gene recombination rate	0.1
IS transposition rate	0.1
IS elements length	1,2,3
RIS transposition rate	0.1
RIS elements length	1,2,3
Gene transposition rate	0.1
Error	0 %

Table 3: GEP Parameters

2.3 Recognition Step – Segmenting Textures

Segmentation is traditionally done using three groups of techniques: 1) edge-based; 2) pixel-based; and 3) region-based. In edge-based techniques, regions are classified by identifying edges between them. An edge can be thought of as pixel locations of abrupt changes. Pixel-based techniques focus on the greyscale or color value of individual pixels. The region-based techniques focus on the continuity of a region in the image. Most of the classic edge and pixel based algorithms are not effective in detecting segments in texture images with randomness. Since those techniques rely on pixel information, they identify local texture patterns as edge areas. We used the region-based technique that is proposed by Song and Ciesielski to implement a Photoshop plug-in as a proof-of-concept application. In this method, the unknown image is analysed in small windows, the same size as the training sub-images (16 by 16). Each window is classified and assigned with a label using the classifier produced in the training step. The window is then moved a few pixels in such a way that the windows always overlap. In fact, the more the windows overlap, the more accurate the segmentation results will be. In our experiment, we moved the window 2 pixels at a time. Once the entire image is scanned and all windows are classified, each pixel is assigned a class label based on a voting mechanism. For example, if a pixel falls more within windows that are classified as class A, the pixel is labelled as class A. Pixels that belong to equal number of windows from both classes are voted randomly.

2.4 Analysing Texture Classifiers

One of the goals of this study was to find out whether GEP is able to detect pattern regularities in the textures.

In order to conduct this part of the study, we generated simple binary (black and white) textures (Figure 2) and test data as in [8].

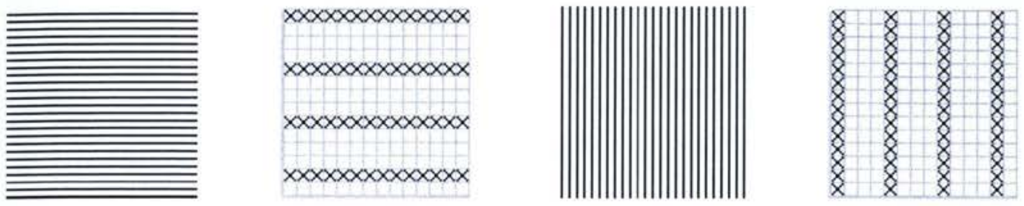


Figure 2: Black and white textures;
from left: horizontal texture, horizontal pixel pattern,
vertical texture, and vertical pixel pattern

2.5 Experimental Design

The objective of the experiments was to benchmark our GEP system against the GP-based technique developed by Song and Ciesielski. Our goal was to simplify the system as much as possible and speed up the evolution process in preparation for real-time applications. We explored four avenues for this purpose as listed in the following subsections.

- **Range Selection Method**

Two range selection methods were considered: 1) Static and 2) Dynamic [7].

- **Function Set**

We experimented with the full function set as in [6] as well as the limited function set which excluded logical operations.

- **Terminal Set**

We examined the effect of including random values as part of the terminal set as proposed in [6].

- **Size of Training Data**

In order to minimize the training time, collections of training data with different number of files were considered.

2.6 Training Data

Brodatz textures were used as input data in our project. These images are greyscale pictures of rather complex textures. We focused on D21, D24, D34 and D57 textures (Figure 3).

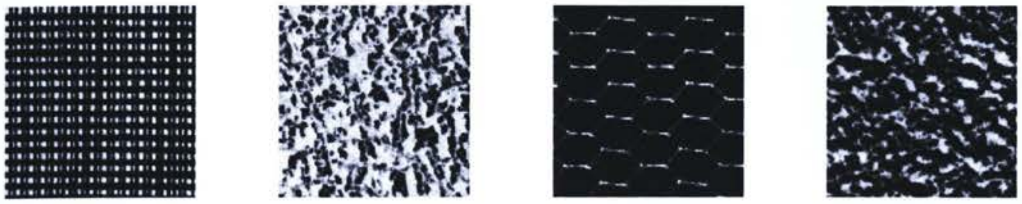


Figure 3: Input textures from Brodatz collection;
from left to right: D21, D24, D34 and D57

The training dataset consisted of 16 by 16 cut outs (sub-images) from the original texture images. In order to classify each texture against the other three, we only needed 99 sub-images from the target texture (class A) and the same number of sub-images from other textures combined (33 each). This is another contributing factor to the speed of the evolution process and an advantage over 2000 sub-images used by Song and Ciesielski.

3 Results

3.1 GEP Detecting Texture Patterns

Using only addition and subtraction operations as the function set, it takes literally no time (less than 10 milliseconds) for GEP to produce simple and short classifiers with perfect accuracy (100% fitness) for black and white patterns (Figure 2). The following classifiers are examples of GEP results:

```
((Pixel[3,1]+(Pixel[1,1]-Pixel[2,1]))-Pixel[0,1])
((Pixel[0,2]-(Pixel[0,0]-Pixel[2,1]))-Pixel[2,3])
```

As shown in the pattern sketches in Figure 2, the horizontal texture has one row of black pixels (pixel = 0) followed by three rows of white pixels (pixel = 255) and the vertical texture has one column of black pixels (pixel = 0) followed by three columns of white pixels (pixel = 255).

It can be seen that each classifier correctly involves at least 4 pixels and work based on the pixels in four consecutive rows or columns. This confirms the fact that GEP is capable of recognizing texture patterns.

Figure 4 illustrates the results of classifying horizontal texture. Both regular and irregular boundaries have been perfectly handled by the classifier. The pictures on the left show the original images and, on the right, are segmentation results. In the output images, class B pixels were eliminated by setting their value to black.

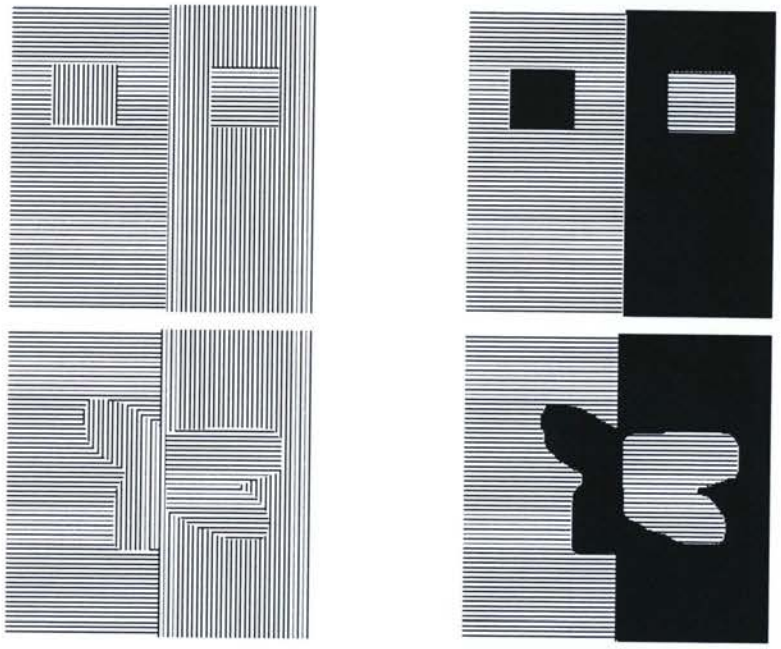


Figure 4: Black and white classification results;
the target is horizontal texture

3.2 Segmenting Greyscale Textures

In this section some of the texture segmentation results from our study are presented. All experiments were conducted using a PC with Pentium M processor 1.50 GHz, 2 GB RAM and Windows XP SP3.

The test database consisted of 250 by 250 greyscale images that were created using D21, D24, D34 and D57 textures from Brodatz album. The images include a variety of cases such as two textures, multiple textures and regular and irregular boundaries. Although the results are only captured and presented for D21 classification, similar results were observed using the classifier for other textures.

The results presented in the following sections were captured in the conditions as listed in Table 4.

Setting	Value
GEP parameters	As listed in Table 3
Function set	Main arithmetic operations only
Terminal set	Pixel values only
Range selection method	Static
Class A	D21 texture
Class B	D24, D34 and D57 textures
Training dataset	99 class A+ 99 class B
Sub-image size	16 by 16
Sliding step size	2 pixels
Classifier fitness	91.5%

Table 4: Test Conditions

The pictures on the left show the original images and, on the right, are segmentation results. In the output images, class B pixels were eliminated by setting their RGB value to black.

1 Two textures, regular boundaries

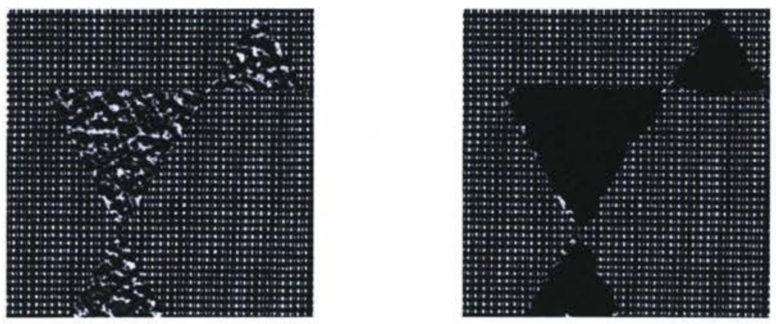


Figure 5: D21 segmentation results;
two textures, regular boundaries

Image Size	Segmentation Time
250 by 250	2.184 sec

2 Multiple textures, regular boundaries

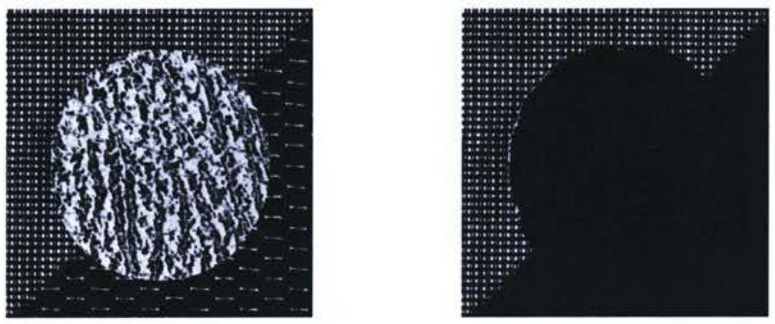


Figure 6: D21 segmentation results;
multiple textures, regular boundaries

Image Size	Segmentation Time
250 by 250	2.266 sec

3 Two textures, irregular boundaries

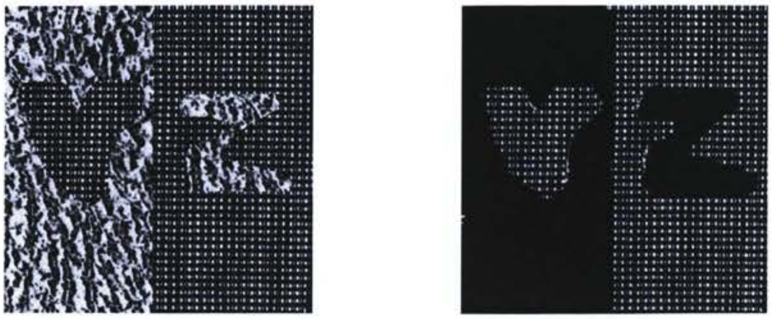


Figure 7: D21 segmentation results;
two textures, irregular boundaries

Image Size	Segmentation Time
250 by 250	2.546 sec

4 Multiple textures, irregular boundaries

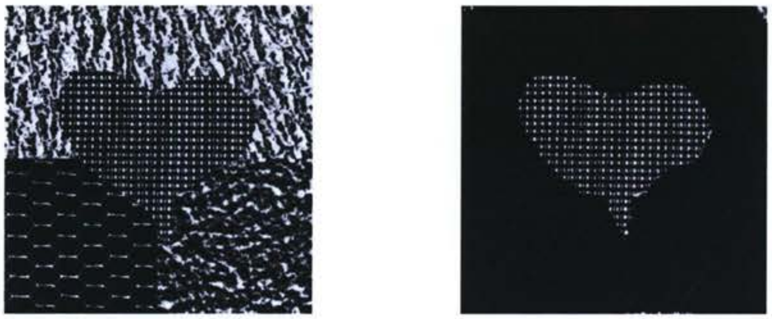


Figure 8: D21 segmentation results;
multiple textures, irregular boundaries

Image Size	Segmentation Time
250 by 250	2.732 sec

Table 5 compares our GEP-based method with the GP-based technique proposed by Song and Ciesielski.

	GEP	GP
# of training data files	198	2000
Range selection method	Static	Dynamic
Function set	Arithmetic operations only	Arithmetic and logical operations
Terminal set	Pixel values only	Pixel values and Random (-1,1)
Maximum # of functions	100	700
Maximum evolution time	20 sec	Unknown
Segmentation time	< 3 sec	< 3 sec

Table 5: GEP vs. GP

4 Discussion

Our experiments proved that dynamic range selection method, being considerably more demanding on computational resources, provides very little benefit over simple static range selection method. As shown in Figure 9, compared to static method, it takes almost twice the time to generate a comparable classifier with the dynamic range selection algorithm.

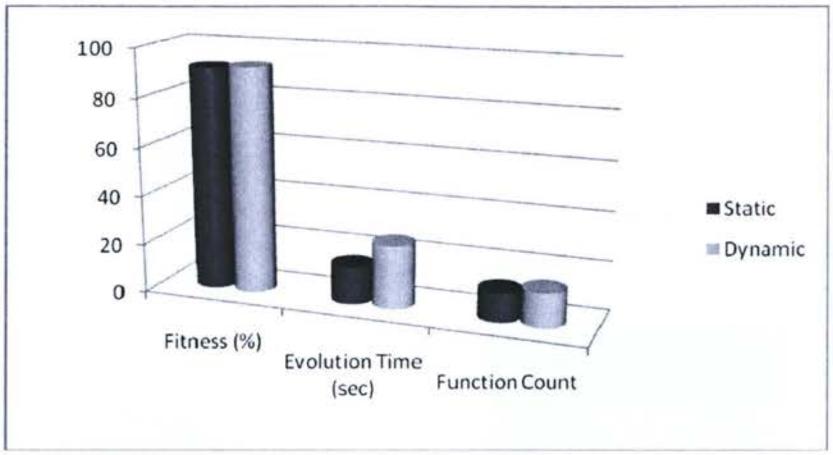


Figure 9: Static Range Selection Method vs. Dynamic

Another simplification we made was reducing the function set to main arithmetic operations. The decision was made based on the fact that logical operators provided little advantage in the system. In fact, they increased the size and complexity of the individuals which led to unnecessary computational complexity, greater resource consumption and slower evolution process. Figure 10 illustrates the abovementioned fact. Limiting the function set to main arithmetic operations (Table 1), we managed to produce simple and short individuals with satisfactory classification results (above 90% fitness). Maximum number of functions in the fittest individual did not exceed 100, which is quiet an advantage over 700 functions in GP-based classifiers [6].

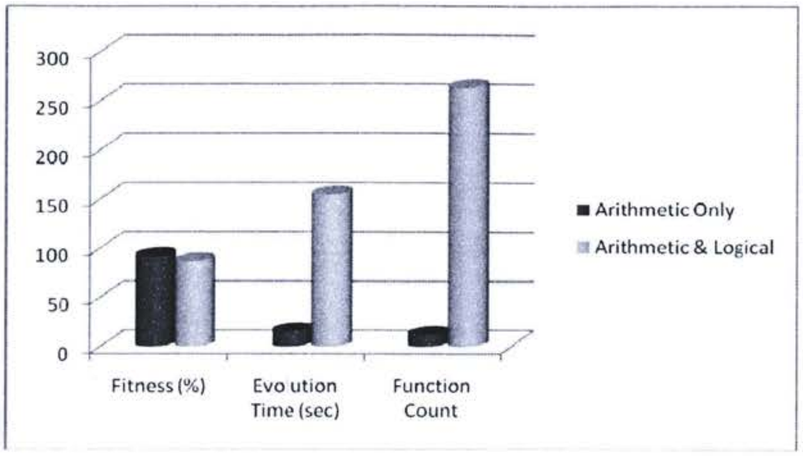


Figure 10: Arithmetic Operations vs. Full Function Set

Figure 11 illustrates the fact that the GEP-based technique is capable of generating satisfactory results with limited training data. Increasing the number of training cases neither improved the accuracy of the classifier (fitness value) nor simplified the best evolved algorithm (function count), while it did increase the evolution time.

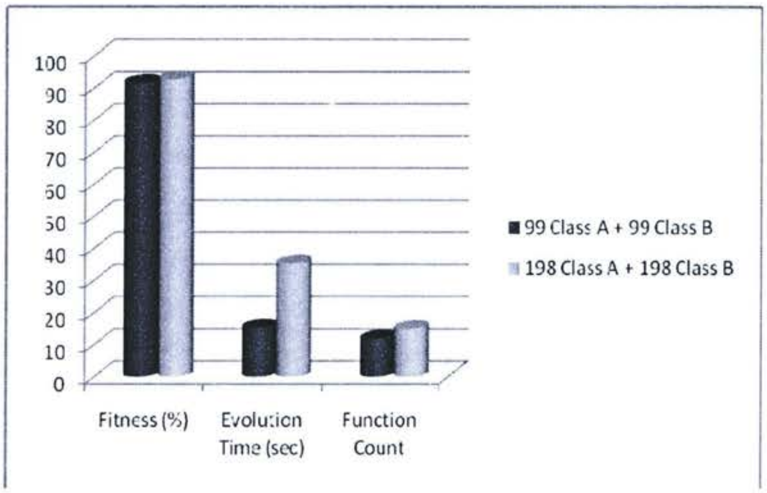


Figure 11: Increasing training dataset

Figure 12 shows the progression of average fitness of the population and the best individual in a typical run. The average fitness curve confirms the presence of genetic material in the population.

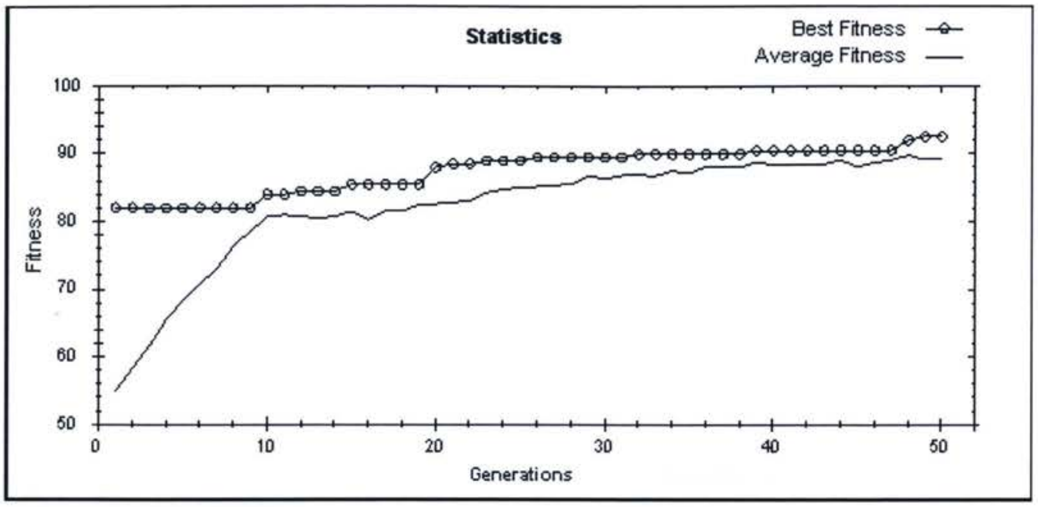


Figure 12: Best Fitness and Average Population Fitness

5 Conclusion and Future Work

In this study we used GEP to extract texture features and produce classifier algorithms out of simple arithmetic operations and direct pixel values.

We managed to evolve classifiers for Brodatz textures with acceptable accuracy and speed. In fact, we used the resultant classifiers in a Photoshop plug-in to demonstrate its commercial applicability.

GEP showed superior performance compared to the GP-based method presented by Song and Ciesielski:

- The algorithm used:
  1. less training data (198 images compared to 2000);
  2. simpler range selection method (static vs. dynamic);
  3. limited function set (arithmetic operations vs. arithmetic and logical operations);
  4. and, only direct pixel values as terminal set as opposed to pixel values and random (-1, 1).
- The classifiers produced by GEP were also smaller compared to the GP-based classifiers (100 vs. 700 functions).
- GEP-based classifiers for Brodatz textures were evolved in as little time as 5-20 seconds. However, since the evolution time in the GP-based technique is not mentioned by Song and Ciesielski, the results cannot be compared.

We also confirmed the fact that GEP is capable of recognizing pattern regularities in textures.

The quick evolution process (5-20 seconds) suggests the practicality of the developed technique for real-time applications such as unsupervised texture segmentation.

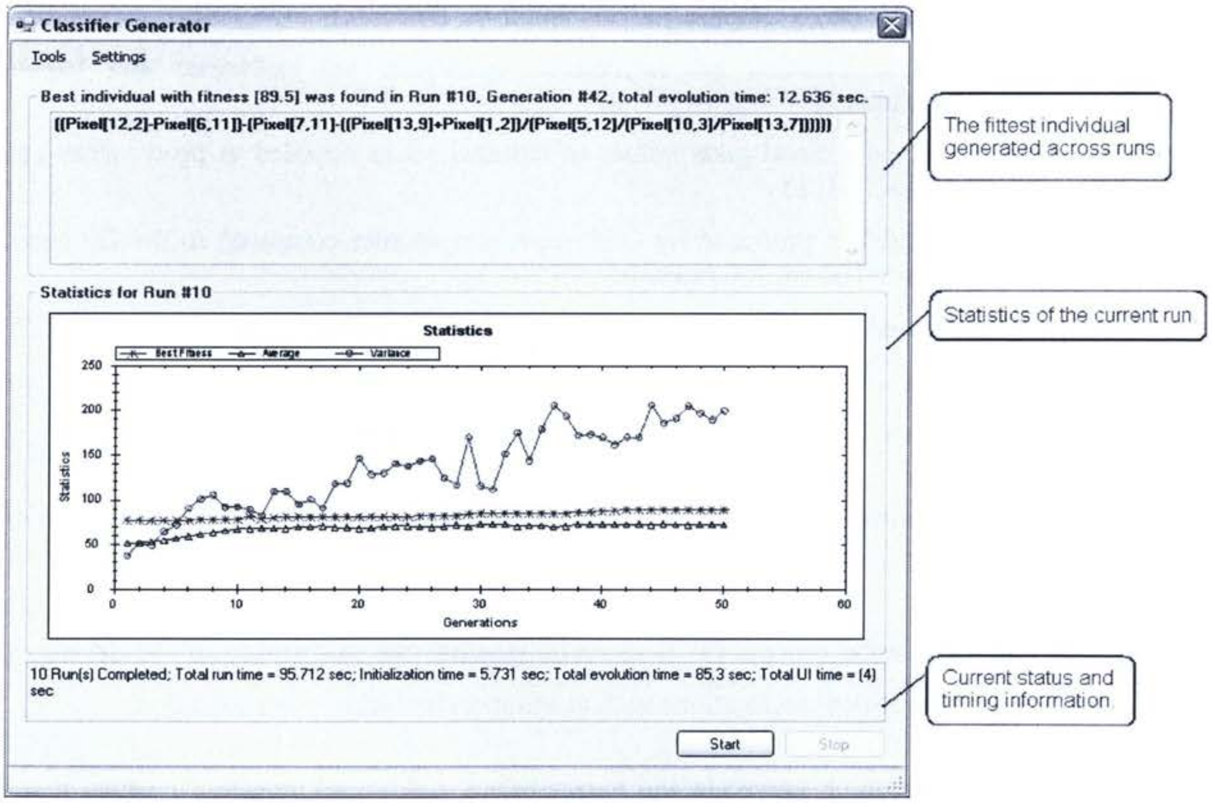
This study focuses on greyscale and binary (black and white) images. However, it can easily be extended to colour images. Moreover, the technique that is developed is a supervised segmentation method, which in future works could be transformed into a more generic segmentation mechanism.

6 Appendix A - Software

Two pieces of software were developed for this research: 1) A C# Windows application to generate texture classifiers and 2) A Photoshop plug-in written in C++ to use the generated classifiers and segment unknown images in the Photoshop environment.

1 Classifier Generator

The application was developed in Microsoft Visual Studio 2005 and using C# language. It compiles to a single executable file (.exe) and requires no installation. Once the .exe is executed, the main form is displayed (Figure 13). Use menu "Settings > Training Data..." to load the training data (Figure 14), which are collections of sub-images cut out of the original texture image.



```
Best individual with fitness [89.5] was found in Run #10, Generation #42, total evolution time: 12.636 sec.
((Pixel[12,2]-Pixel[6,11])-(Pixel[7,11]-((Pixel[13,9]+Pixel[1,2])/(Pixel[5,12]/(Pixel[10,3]/Pixel[13,7])))))

10 Run(s) Completed; Total run time = 95.712 sec; Initialization time = 5.731 sec; Total evolution time = 85.3 sec; Total UI time = (4) sec
```

Figure 13: Classifier Generator – Main Form

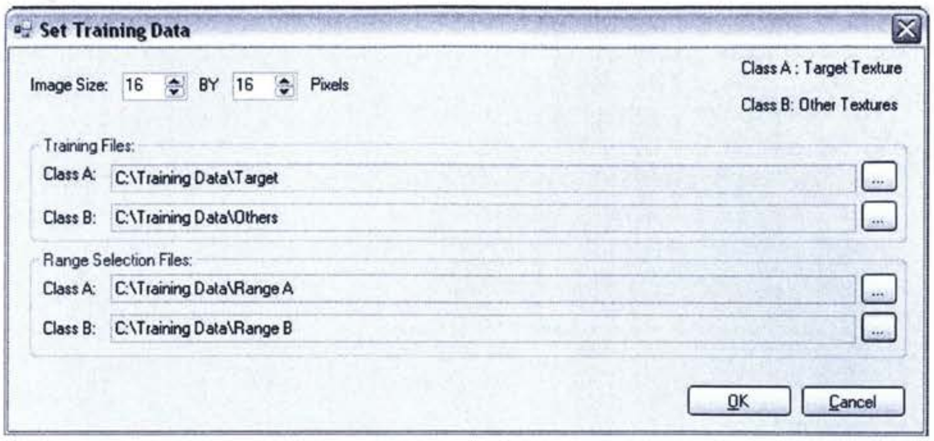

Set Training Data

Image Size: 16 BY 16 Pixels

Class A : Target Texture

Class B: Other Textures

Training Files:

Class A: C:\Training Data\Target

Class B: C:\Training Data\Others

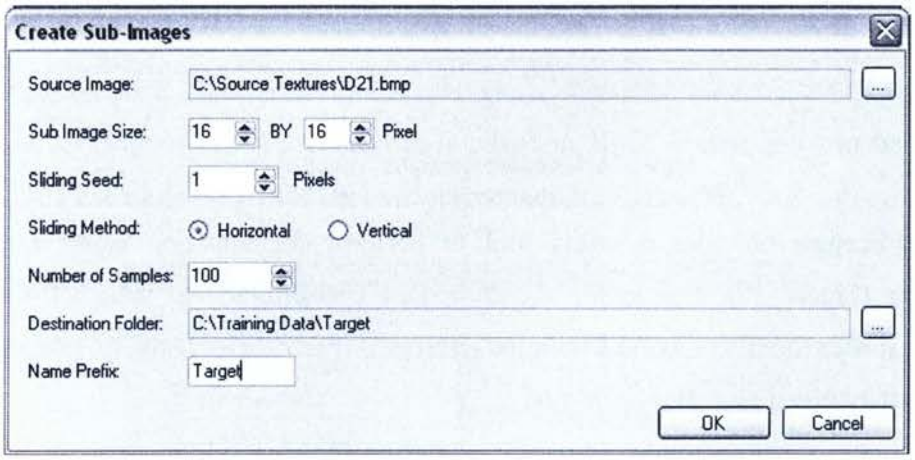
Range Selection Files:

Class A: C:\Training Data\Range A

Class B: C:\Training Data\Range B

Figure 14: Classifier Generator – Set Training Data

The application provides a tool to generate sub-images for training the system. Use menu "Tools > Create Sub-Images…" for this purpose (Figure 15).


Create Sub-Images

Source Image: C:\Source Textures\D21.bmp

Sub Image Size: 16 BY 16 Pixel

Sliding Seed: 1 Pixels

Sliding Method: [✓ Horizontal] Vertical

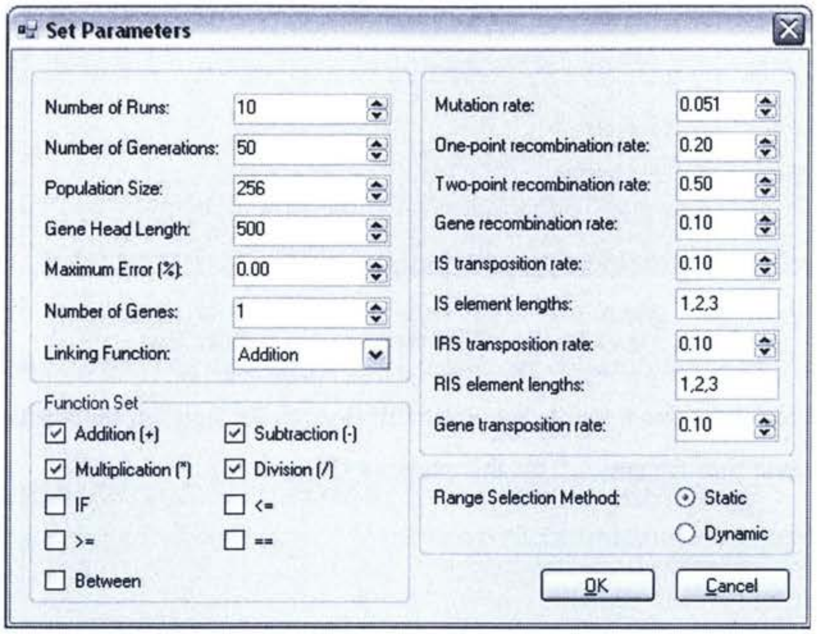
Number of Samples: 100

Destination Folder: C:\Training Data\Target

Name Prefix: Target

Figure 15: Classifier Generator – Create Sub-Images

The rest of the parameters are set to default values, so that as soon as the training data is loaded, you can start the evolution process. In order to modify default parameters, use menu "Settings > Parameters..." (Figure 16).


Set Parameters

Number of Runs: 10
Number of Generations: 50
Population Size: 256
Gene Head Length: 500
Maximum Error (%): 0.00
Number of Genes: 1
Linking Function: Addition

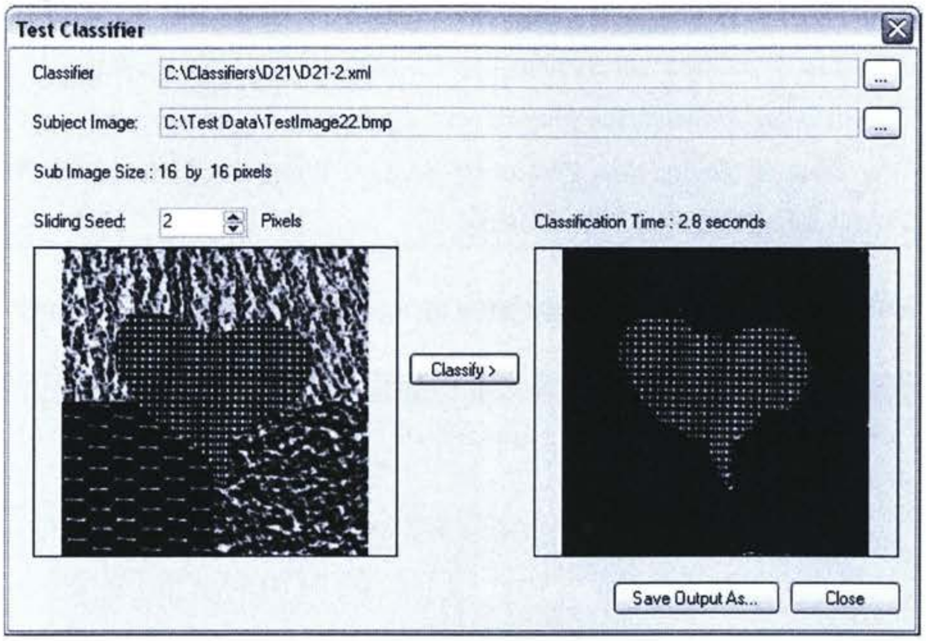
Function Set
[✓] Addition (+)
[✓] Subtraction (-)
[✓] Multiplication (*)
[✓] Division (/)
[ ] IF
[ ] <=
[ ] >=
[ ] ==
[ ] Between

Mutation rate: 0.051
One-point recombination rate: 0.20
Two-point recombination rate: 0.50
Gene recombination rate: 0.10
IS transposition rate: 0.10
IS element lengths: 1,2,3
IRS transposition rate: 0.10
RIS element lengths: 1,2,3
Gene transposition rate: 0.10

Range Selection Method: [✓ Static] Dynamic

Figure 16: Classifier Generator – Set Parameters

The application provides a utility tool to perform segmentation using an evolved classifier (Figure 17). Use menu "Tools > Test Classifier..." to launch the tool. By default, it uses the fittest individual in the latest run, if any. Alternatively, you could load a previously saved classifier (.xml file).



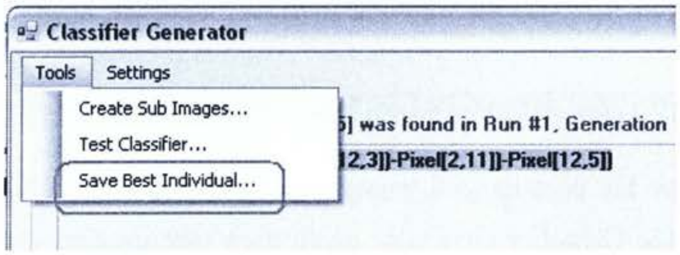
Classifier: C:\Classifiers\D21\D21-2.xml

Subject Image: C:\Test Data\TestImage22.bmp

Sliding Seed: 2 Pixels

Figure 17: Classifier Generator – Test Classifier

In order to save the fittest individual in the latest run, use menu "Tools > Save Best Individual..." (Figure 18). The classifier is saved in XML format and can be used as input to the Photoshop plug-in to perform segmentation (Figure 20).



]] was found in Run #1, Generation

]2,3])-Pixel[2,11])-Pixel[12,5])

Figure 18: Classifier Generator – Save Best Individual

2 Segmentation Plug-in

The plug-in was developed using Photoshop CS SDK and in C++ language. It compiles to .8bs file. In order to install the plug-in, exit Photoshop, if running, and copy the .8bs file to the Plug-in folder for Photoshop (e.g. C:\Program Files\Adobe\Photoshop CS\Plug-Ins\Adobe Photoshop Only\Filters).

Once Photoshop is started, the plug-in shows up under the "Select" menu (Figure 19).

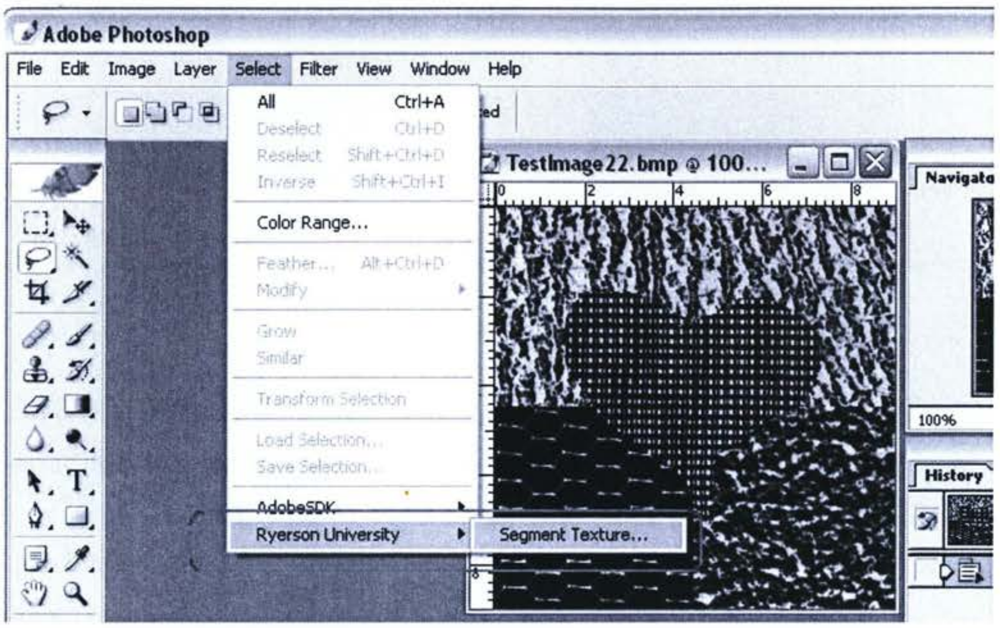
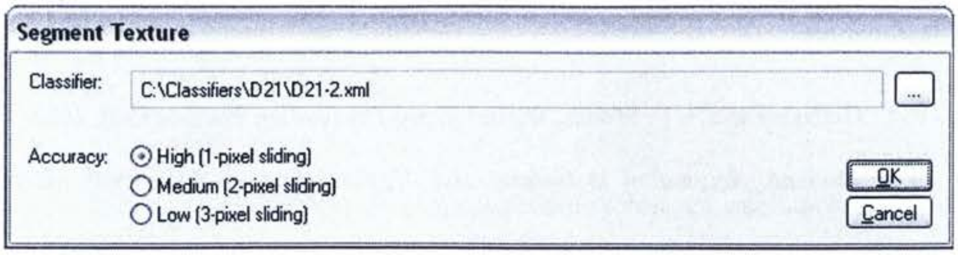


Figure 19: Photoshop - Texture Segmentation Plug-in

To be able to use the plug-in, you would need a classifier file (.xml), which can be produced using the Classifier Generator application (see previous section). Use menu "Select > Ryerson University > Segment Texture..." to specify the classifier and segmentation accuracy (Figure 20) and launch segmentation. Once the dialog is dismissed, the segmentation algorithm (refer to section 2.3; page 7) is performed and the pixels which belong to the target texture are selected as a region (Figure 21). The resultant selection can be used in the Photoshop environment just like a native selection region to perform tasks such as move, cut, copy, paste, form layers, etc.



Segment Texture

Classifier: C:\Classifiers\D21\D21-2.xml

Accuracy: [✓ High (1-pixel sliding)] Medium (2-pixel sliding) Low (3-pixel sliding)

Figure 20: Photoshop – Launching Texture Segmentation

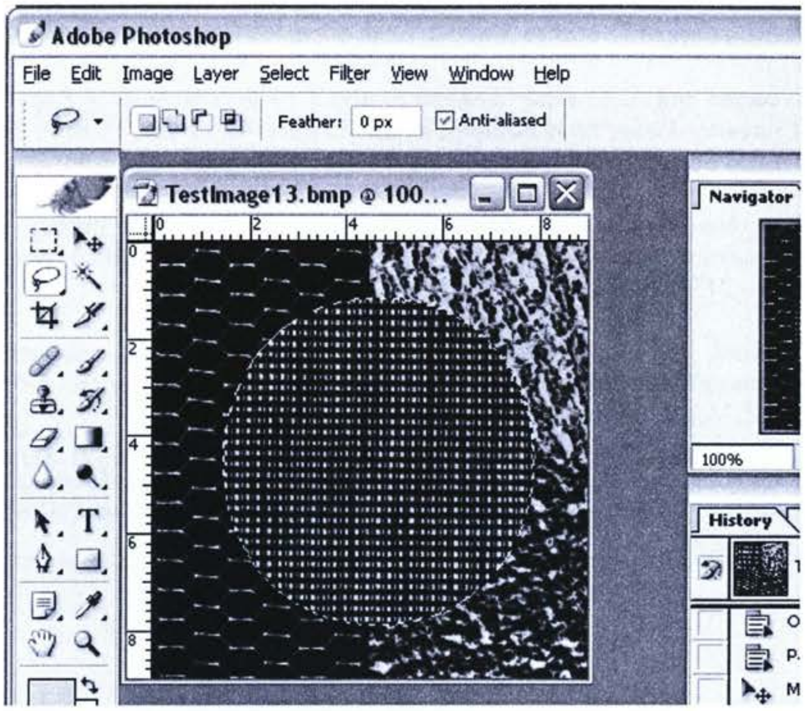


Figure 21: Photoshop – Segmenting Target Texture

7 References

1. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, 2002.

2. J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

3. J. R. Koza, et al., *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann, 1st edition, 1999.

4. C. Ferreira, "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems", *Complex Systems*, Vol. 13, issue 2, pp. 87-129, 2001.

5. M. Tuceryan and A. K. Jain, "Texture Analysis", *The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*, by C. H. Chen, L. F. Pau, P. S. P. Wang, pp. 207-248, World Scientific Publishing Co., 1998. (Book Chapter)

6. A. Song and V. Ciesielski, "Fast texture segmentation using genetic programming" *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on volume 3*, Vol. 3, pp.2126 – 2133, 2003.

7. T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming", *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on Volume 2*, Vol. 2, pp. 1070 - 1077, 2001.

8. A. Song, A., et al., "Texture classifiers generated by genetic programming", *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on Volume 1*, Vol. 1, pp. 243 – 248, 2002.

8 Glossary

B

Brodatz textures A collection of greyscale texture images published in 1966. This collection has become a standard database in texture analysis studies.

C

Chromosome A GEP individual which is a candidate solution to the problem at hand.

Class A The label associated with the target texture.

Class B The label associated with non-target textures.

Classifier An algorithm evolved by GEP, which is able to classify certain texture, a GEP chromosome.

D

Dynamic range selection In this method each chromosome is allowed to dynamically determine a separate set of ranges for class boundaries as opposed to the fixed ranges that are chosen in a static range selection method that all GEP individuals must adhere to.

E

Edge Pixel locations of abrupt changes in the image.

Edge-based segmentation One of the three groups of segmentation techniques. In this method segments are classified by identifying the edges between them. The other two categories are pixel-based and region-based methods.

F

Feature Extraction The process of defining a mathematical model to represent a random texture.

G

Gene Expression Programming A new evolutionary algorithm that evolves computer programs to solve variety of problems.

GEP See Gene Expression Programming.

P

Pixel-based segmentation One of the three groups of segmentation techniques. In this method segments are identified based on the greyscale or color value of individual pixels. The other two categories are edge-based and region-based methods.

R

Region-based segmentation One of the three groups of segmentation techniques. In this method segments are identified based on the continuity of the regions in the image. The other two categories are pixel-based and edge-based methods.

S

Static range selection In this method, the range of real values that the GEP individuals are evaluated to is divided into negative and positive numbers. Positive numbers are associated with class A texture while negative numbers are associated with class B texture(s).

Sub-image A small cut out from the texture image that is used as training data.

T

Texture segmentation In texture segmentation, the image is divided into regions with homogeneity with respect to texture.

Texture classification In texture classification, an unknown sample image is partitioned into regions that belong to one of a set of known texture classes.

9 Index