Edge Detection of Petrographic Images Using Genetic Programming

Brian J. Ross
Brock University
Dept. of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
bross@cosc.brocku.ca

Frank Fueten
Brock University
Dept. of Earth Sciences
St. Catharines, Ontario
Canada L2S 3A1
ffueten@craton.geol.brocku.ca

Dmytro Y. Yashkir
Brock University
Dept. of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
dy96ab@cosc.brocku.ca

Abstract

This paper discusses work in progress that uses genetic programming to evolve edge detectors for petrographic images. Microscopic images of thin sections from mineral samples are obtained using a rotating polarizer microscope. These images are then processed using a number of filters, resulting in a set of nine filtered image parameters. In order to be useful for higher–level analysis, such as automatic mineral identification, the grain boundaries within these images must be identified. Using genetic programming, edge detecting functions are evolved for this purpose. The edge detectors may use as any of the filtered image parameters as input. Since the source images are large, a subset of the images is sampled for training, and the remainder of the image is used for testing. This training data is selected with a biased random sampling strategy. The complexity of the images dictates that a generic edge detector for all mineral specimens is infeasible. Rather, the most useful edge detectors will be those that are specialized for particular families of mineral specimens.

1 INTRODUCTION

Microscopic images of thin sections from mineral samples are useful for identifying mineral specimens. While inspecting a thin section, a geologist will visually identify the grains resident on the image. Each grain in turn will have implicit visual characteristics dependent upon the specimen of mineral comprising it. Viewing the sample under polarized light can aid in the identification of grains, since polarization enhances the visual characteristics of grains. Manually inspecting thin sections and identifying grain features is a time-consuming and error-prone process, even for an experienced geologist.

This paper discusses work in progress that is investigating the use of genetic programming (GP) to evolve edge detectors for petrographic images. Recent work has investigated the automatic analysis of petrographic thin sections using computers (Goodchild and Fueten 1998, Thompson *et al.* 1999, Ross *et al.* 1999). To satisfy this goal, grain boundaries must be automatically identified. Using genetic programming, edge detecting programs are evolved that delineate the boundaries between distinct grains on a thin section image. An edge detecting program will draw the boundaries between grains when passed over an image file. Care must be taken that the program is not too liberal (erroneously identify grain areas as edges) nor too conservative (miss identification of true grain edges). Since the source images are large, not all the image data can be used for training, and hence a biased random sampling strategy is used.

Section 2 discusses petrographic thin section images and their preparation before use in the GP system. Details about the genetic programming experiments are given in section 3. Results of the experiments are discussed in section 4. A summary discussion and comparisons to other work conclude the paper in section 5.

2 IMAGE PROCESSING OF PETROGRAPHIC THIN SECTIONS

The petrographic microscope is a basic tool employed by many geologists to identify minerals, estimate grain size or shape and to obtain modal percentages of minerals in thin sections. In plane-polarized light many minerals are colourless, which makes it impossible

Table 1: Image parameters from polarizing microscope

Cross-polarized light	Plane polarized light
p1: gradient	p6: min. position
p2: max. position	p7,p8,p9: min. intensity
p3,p4,p5: max. intensity	

to distinguish grain boundaries between two adjacent colourless grains. Similarly, in cross-polarized light the interference colour displayed depends on the mineral type, the orientation of the grain with respect to the polarizers and the thickness of the thin section. Hence, two adjacent grains may have similar interference colours at some orientations of the thin section with respect to the polarizers. This problem of lack of contrast is commonly overcome by rotating the microscope stage with respect to the fixed polarizers. The human brain and vision system have no problem keeping track of individual grains as they rotate around the field of view. Unfortunately this procedure is a major obstacle for an image processing system, as the computer has to track the behaviour of a point within a grain in colour space, as well as the motion of that point as the thin section is rotated.

(Fueten 1997) presents the design of a fully automated polarizing stage for a petrographic microscope, which allows a thin section to remain fixed while the polarizers are rotated. Hence any point within a grain is registered to the same pixel at all positions of the polarizers, greatly simplifying the computational requirements. The stage is used in conjunction with a computer with a video capture board. By selectively obtaining data from images with different polarizer positions, the stage greatly enhances the potential for petrographic image processing. Each image captured requires approximately 1 Mb of storage space, hence a complete set of images for a 180 degree rotation of the polarizers under both plane and crossed-polarized light would require approximately 400 Mb. To reduce storage requirements, a composite data set is constructed that contains selected information obtained during a 180 degree rotation of the polarizers, first under crossed-polarized, then under plane light. Combining data from different images into a composite image is possible as each pixel remains registered to the same point within a grain in the stationary thin section for all orientation of the polarizers.

The data computed for each pixel is listed in Table 1, and will now be described. All the data take the form of filtered images, and are used as input parameters for the genetic programming system in Section 3. The entries a single parameter are monochromatic, while those with 3 parameters are RGB values.

The data collected using crossed-polarized light include the average, maximum intensity, maximum position and gradient. They are computed as follows.

Gradient (p1): This filter is of primary importance for detecting grain edges. A gradient operator, which compares the intensity of a pixel with that of its neighbours in both the positive x and y directions is passed over each incremental image. For each pixel the maximum gradient value is added to an incremental gradient array. The gradient at visible boundaries between grains is larger than gradient values in the interior of grains. A total gradient array is built up by addition of incremental gradient values while the polarizing filters are rotated through 180 degrees. Following the acquisition of a complete data set, the total gradient array is scaled to an 8 bit range, with values in the range of 0 - 255. This image is viewed as a gray-scale image with high/bright values representing grain boundaries while low/dark areas form the interior of grains.

Maximum position (p2): Position values record the rotational orientation of the polarizing filters when a pixel reaches its maximum and minimum value. For a 180 degree rotation of the polarizers, position values have a range of 1-200 as a pixel can reach its maximum or minimum value at any one of the 200 steps. Position values are represented as 8 bit grey scale images.

Maximum intensity (p3, p4, p5): The maximum intensity value corresponds to the maximum interference colour of a pixel within a grain during rotation of the polarizers through 180 degrees. The intensity of a pixel is calculated using the intensity value in Hue, Saturation, Intensity (HSI) colour space (I= (red+green+blue)/3) (Gonzalez and Woods 1992).

For observations under plane polarized light, a neutral density filter is rotated into the light path. The data collected under plane-polarized light includes and minimum position (p6) and the minimum plane intensity (p7, p8, p9). The procedures to obtain these data are similar to those described above.

Two other parameter sets, average cross polarized and maximum intensity plane polarized, were available. However, experiments showed that they contributed negligibly to the overall results, and hence were not used in subsequent runs.

Table 2: GP Parameters

Parameter	Value
Functions	float: if, avg, amin, amax, sdev, min, max, -, +, *, / integer: inc
Terminals	float: ephem, p1, ..., p9 integer: ephem
Fitness function	biased random sampling
Population size	2000
Max. generations	75
Max. runs	6
Prob. crossover	0.95
Prob. mutation	0.05
Prob. leaf mutation	0.90
Max. depth initial	6
Max. depth offspring	17
Tournament size	5

3 EVOLVING EDGE DETECTORS

3.1 System design

Table 2 lists the main parameters used. The GP system used is the typed version of the lilGP 1.1 system (Zongker and Punch 1995). Typing is useful since we use both integer and floating point data types in the programs (Montana 1995).

The program will be passed across the pixels for an image, and will indicate whether each pixel should be drawn as an edge or not. The root expression for the program trees is a floating point expression. If the expression evaluates to a value greater than zero, then that pixel of the image is considered to be an edge, and should be drawn as such. Otherwise, the pixel is not an edge, and should be skipped.

Some functions (*sdev*, *amin*, *amax*, *avg*) work on a square area of adjacent pixels surrounding the current pixel being processed. One of the integer arguments to these area functions indicates the size of grid to process. The integer value modulo 3 will indicate whether to process a 5x5, 7x7 or 9x9 grid. The other integer argument specifies which one of the 9 parameters in Table 1 to use in the computation. The function *sdev* computes the standard deviation of the grid of pixels surrounding the current pixel:

$$\sqrt{\frac{\sum (v_i - a)^2}{n}}$$

where v_i are the parameter values of the entries in the grid, a is the average of the grid area, and n is the number of entries in the grid. The *amin* and *amax* functions compute the area minimum and maximum values, while *avg* computes the area average.

The *if* function has 4 arguments – two floating point expressions to be used in a relational test, and two result expressions. If the first argument is less than the second, then the value of the third argument is returned as a result. Otherwise the fourth expression's value is returned. The remaining floating point functions are the usual arithmetic functions. The floating point terminals include the twelve parameters from the image data in Table 1, and ephemeral random constants (Koza 1992).

Integer expressions consist of either ephemeral random constants or the function *inc*, which increments an integer expression. The *inc* function permits variable integer values without needlessly complicating the integer expression syntax. Any function using an integer expression argument will convert it to the required integer value modulo K – either for grid sizes (K=3), or parameter reference (K=9).

Contrary to other's experiences (Poli 1996), we did not find it desirable to preprocess the image data to compute the filtered results as denoted by functions such as *sdev* and *avg* in Table 2. It is too expensive in time and memory resources to pre-calculate these function values for all the possible combinations of input images and parameters. Given that the training uses a sampled subset of the image data, the overhead in computing these functions during evolution is not prohibitive.

3.2 Fitness strategy

Table 3: Biased Sampling Parameters

Parameter	Value
Within edge:	700
Border edge:	700
Adjacent to edge:	700
Not edge:	800
Total pixels per sample:	2900
Resampling rate:	5 generations

A biased random sampling strategy is used for selecting image data for training. A classification scheme is used that permits sampling from specific categories of pixels from the training data. Each set of training data has supplied with it a solution image, which indicates all the true edges in the data. Pixels on this solution image are either edges or non-edges. Based on this information, each pixel in the training data is classified into one of the following four categories:

1. Within edge: This is a pixel within the boundaries of an edge. All the pixels surrounding it are classified as edges in the solution image.
2. Border edge: This is a pixel on an edge, but some of the surrounding pixels are not classified as edges in the solution image.
3. Adjacent to edge: The pixel is not on an edge, but it is adjacent to one.
4. Not edge: The pixel is not on an edge, nor is it adjacent to one.

The rationale behind this classification is that the critical training cases are those in or near edges, and these categories are used to identify such occurrences. In addition, because the majority of the pixels on an image comprise grain areas, this classification will permit sampling to be biased against over-selection of grain area data.

Table 3 shows the parameters used for directed sampling. Resampling of the training data is undertaken every 5 generations. A large enough set of representative data is sampled in each training set to prevent problematic hill climbing (Ross 2000).

The fitness value for a program is computed as:

$$Fitness = 1 - \left(\frac{ce}{te} * \frac{cn}{tn}\right)$$

where ce is the number of correctly identified edges, te is the total number of edges, cn is the number of correctly identified non-edges, and tn is the total number of non-edges.

4 RESULTS

Two 640 by 480 pixel thin section images of granitic gneisses were used, along with their filtered parameters of Table 1. Figures 1 and 2 shows their grey-scale maximum intensity cross polarized images (a), the intended target edge solution (b), and edge detection results (c, d, e). The specimen in Figure 1 is considered to be the more difficult of the two sets. A number of different experiments were undertaken on each set of image data. A "best" GP edge detector from 6 GP runs was determined for the mineral specimens in (c). The edge detector trained on the other image data was tested on the given mineral specimen in (d). For comparison, the results from an artificial neural network (ANN) edge detector is included in (e), and is discussed below.

Although all the edge detector results have visible noise, it is felt that it can be removed with either conventional filtering or a second pass noise removal program evolved for this purpose (more discussion of this is in Section 5). Also note that the solution results in image (c) in Figures 1 and 2 are a combination of training and testing, since only 2900 pixels in the images are used for training per sample, which is less than 1% of the entire image. Given that 25 re-samples are used during the course of a 75 generation run, at most 23.6% of the image is used for training. Hence over 75% of these images can be considered "testing data". The image in (d) is entirely testing data with respect to the solution edge detector evolved from the other data set.

Table 4: Results summary

Experiment	Fitness	Correct edges	Correct grains	Correct overall
GP 1 image 1	0.590	0.798	0.836	0.829
GP 2 image 1	-	0.761	0.887	0.863
ANN image 1	-	0.588	0.737	0.708
GP 2 image 2	0.633	0.835	0.909	0.900
GP 1 image 2	-	0.750	0.890	0.874
ANN image 2	-	0.618	0.885	0.854

A numerical summary of the results is given in Table 4. Note that the values in this table are somewhat deceptive. For example, if an edge detector classifies all pixels as edges, then the value for correct edges will be 100%. This similar holds for correct grains. Given that the majority of pixels in an image are grains, the grain value biases the correct overall value. From this perspective, the qualitative results shown in Figures 1 and 2 are perhaps a better measure of success.

The best edge detector for set 1 is:

```
(* (avg 91 64)
   (- (avg (Inc (Inc (Inc (Inc 23))))
           (Inc (Inc (Inc 36))))
      (amax (Inc (Inc (Inc (Inc 36))))
            (Inc 84))))
```

A translation into more meaningful parameter and grid notation is:

```
(* (avg p2 "7x7")
   (- (avg p1 "5x5")
      (amax p5 "7x7")))
```

This is a surprisingly simple program.

Set 2's best edge detector (translated) is:

```
(+ (min (+ (- (/ (if (/ p7 p4)
```

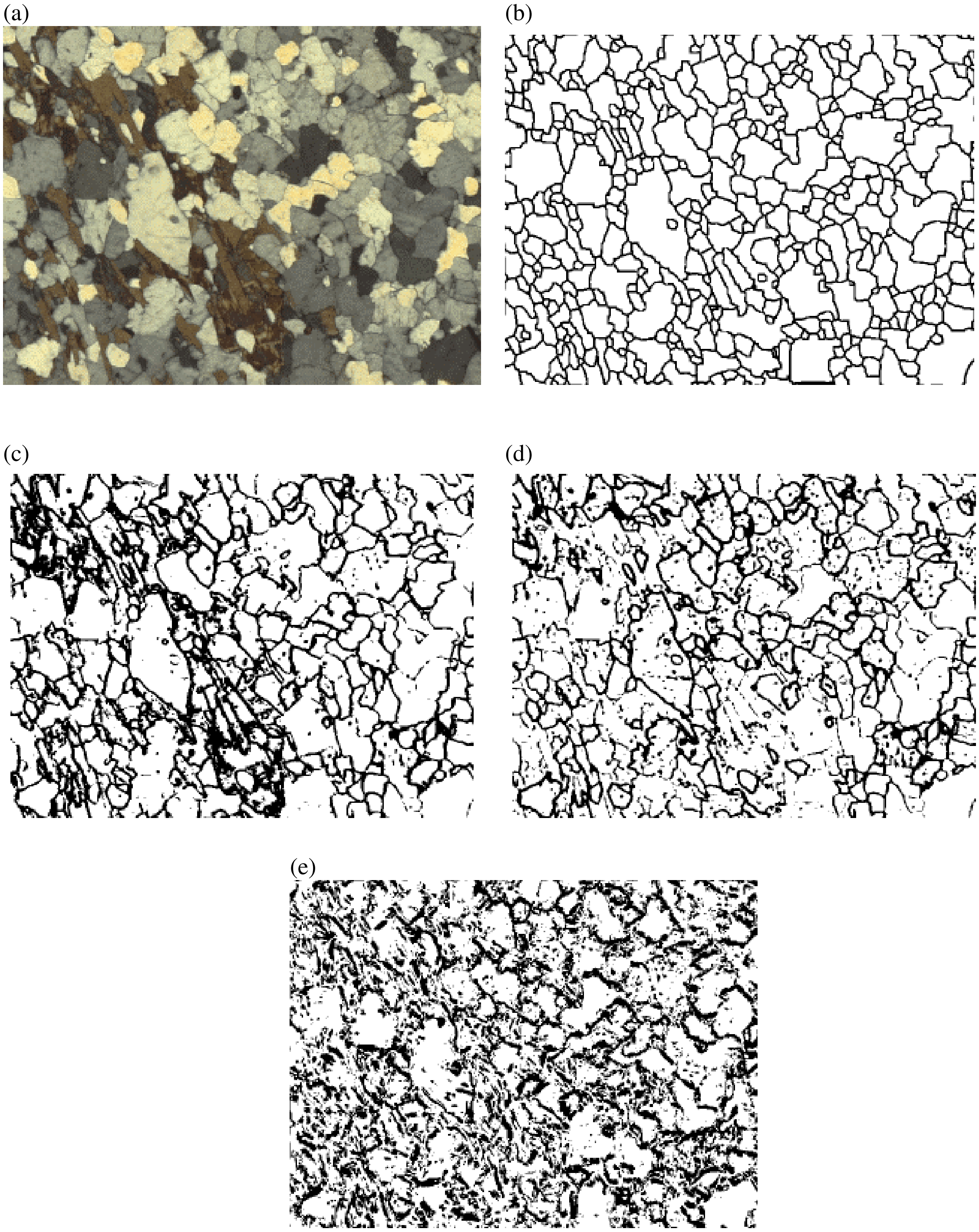

Figure 1: Results for experiment 1: (a) maximum intensity crossed; (b) solution edges; (c) GP edge detector; (d) GP edge detector from experiment 2; (e) neural network.

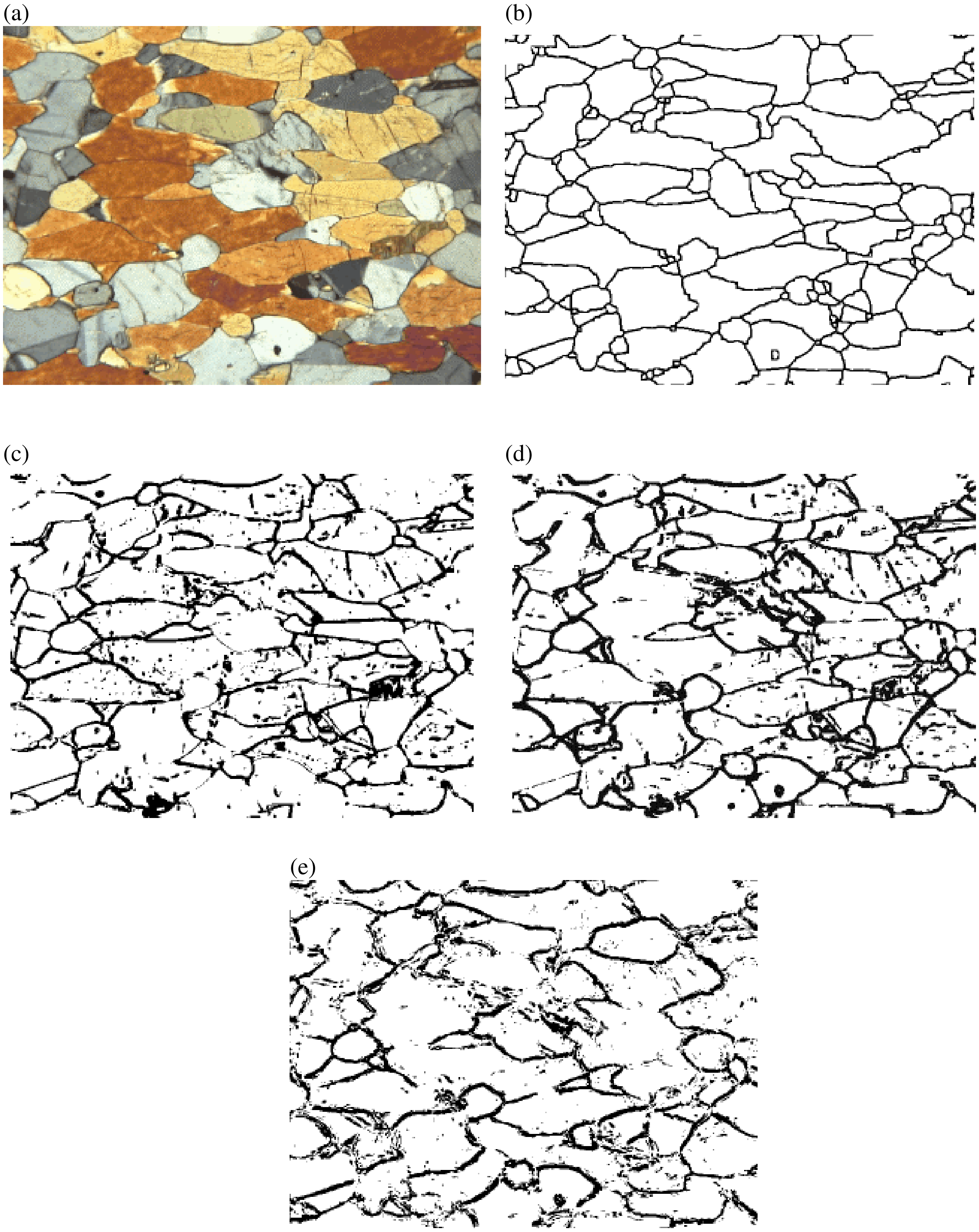

Figure 2: Results for experiment 2: (a) maximum intensity crossed; (b) solution edges; (c) GP edge detector; (d) GP edge detector from experiment 1; (e) neural network.

```
                    (/ p6 p8)
                    (avg p9 "9x9")
                    (amin p1 "7x7"))
                 (max (- p2 p9)
                      (+ (/ p6 p8) p1)))
                          0.84231) p1)
        (- (amin p1 "7x7")
           (sdev p1 "9x9")))
    (avg p1 "5x5"))
```

The artificial neural network that produced the edge results in image (e) in Figures 1 and 2 uses a 3-layer feed forward architecture (Fueten and Thompson 2000). It was trained on the same image data as the GP experiments, using approximately 2% of the image data (6000 pixels) for 2000 iterations. The results shown are considered inferior to what it is normally capable of producing, should a more judicious selection of training data be undertaken.

5 CONCLUSION

The nature of petrographic images makes edge detection a particularly challenging problem as far as image processing problems are concerned. Natural phenomena such as microscopic images of mineral grains usually do not exhibit clean discrete signals, but rather, are replete with noise and other unavoidable artifacts. For example, grains often contain fractures, which visually appear to be edges, but are not considered to be such, since the same mineral grain resides on each side of a fracture. From this perspective, the edge detectors evolved in our experiments exhibit promising results. Because image data has been scarce up to now, more extensive training and testing will be undertaken as more data becomes available.

An important factor in determining the success of edge detection evolution is the training set. Our approach permitted the selection of the relative classes (edge, near-edge, grain, ...) of data to use, which does give some biased control of random sampling. What is more desirable, however, is for problematic areas of the image to be either manually or automatically identifiable, and then be used for additional sampling during training. It is typical to see different edge detectors having similar difficulties with specific portions of images. Future work will enhance the training procedure, for example, by permitting the user to mark image areas for concentrated sampling.

The solution edges used were manually massaged from the output from an edge algorithm using conventional filtering discussed below (Goodchild and Fueten 1998). These solutions contain errors which negatively affect training. For example, the solution edges are wide enough that they straddle grain boundaries. Thus there are implicit contradictions in the edge solution images, in which similar pixels are classified as both edges and non-edges in different portions of the image. Although the impact of these errors is statistically weakened with a large training set, solutions with more refined grain edges are desirable, and would encourage the evolution of better quality edge detectors.

Work has commenced on evolving noise filters, which attempt to correct the errors from the GP edge detectors. In a sense, a noise filter is like a second pass edge detector, except that it expects the majority of its input data to be correctly identified. Therefore, their evolution is simpler than that of edge detectors, because noise filters need only be trained on intermittent errors from edge detectors, and can ignore the majority of the image that is normally processed correctly. It is unknown whether general noise filters can be obtained, or if each edge detector will require its own specific filter. An alternative is to apply conventional filters to the results of the GP edge detectors in order to remove noise.

Earlier work in (Goodchild and Fueten 1998) extracted edges from the gradient image using a sequence of image processing filters. This edge extraction procedure used 10 steps and 7 separate routines to extract the edges, and is similar to the procedures implemented in the Canny edge detector (Canny 1986). Output of this procedure consists of closed edges only. Such edges are ideal input for calculations of grain size or shape which are easily calculated on closed shapes. Although most edges are determined accurately, the solution is not perfect. In clean thin sections, the algorithm produces edges which require little to no editing. The procedure fails, however, to determine the edges of small, narrow grains and produces unsatisfactory results in rocks that contain alterations or regions with a fine grained matrix. Boundaries must be either added or deleted by manual editing.

The results obtained so far with the genetic programming edge detectors are resolve some of the problems of those in (Goodchild and Fueten 1998). One improvement is that the GP edge detectors are able to find edges on narrow or fine grains, unlike the earlier approach. A disadvantage, however, is that the GP edge detectors are trained to work with specific thin section types. The filter sequence approach is generalized to work on any thin section, but with variable results. A future research project is to use GP to evolve edge clean-up programs which will take output from the edge detecting in (Goodchild and Fueten 1998), and remove erroneous or missed edges. This is a considerably easier task than evolving a complete edge detecting algorithm as done in this paper.

(Fueten and Thompson 2000) use artificial neural nets to evolve petrographic edge detectors, and two examples of that work are given in this paper. One advantage of the ANN approach is the ability to retrain a given ANN edge detector at any time on problematic areas of an image, hence improving its performance. This is impossible in GP: although it is difficult (but not impossible) to stop evolution in mid-stream and manually set a new training set, it is very difficult to rationalize what program or programs in the population to use as candidates for identifying problematic performance. The best K programs in the population may exhibit quite variable characteristics across an image, and reconciling their overall performance is difficult and arbitrary. With an ANN, there is one and only one trained net which must be retrained, and so it is a simple matter to incrementally supplement its training on successive sets of data.

Examples of other related research is the use of GP for 1-D edge detection (Harris and Buxton 1996), 2D edge detection (Lucier *et al.* 1998), and 2D feature detection (Daida *et al.* 1996, Poli 1996, Winkeler and Manjunath 1997). It would be interesting to compare our edge detectors with others (especially those in (Lucier *et al.* 1998)) on petrographic images.

Acknowledgement: This research is supported through NSERC Operating Grants 138467 and 0046210, and an NSERC undergraduate research grant.

References

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Learning* **8**(6), 679–698.

Daida, J.M., T. F. Nersano-Begey, S.J. Ross and J.F. Vesecky (1996). Computer-Assisted Design of Image Classification Algorithms: Dynamic and Static Fitness Evaluations in a Scaffolded Genetic Programming Environment. In: *Proc. Genetic Programming 1996* (J.R. Koza *et al.*, Ed.). MIT Press. pp. 279–284.

Fueten, F. (1997). A computer controlled rotating polarizer stage for the petrographic microscope. *Computers and Geosciences* **23**, 203–208.

Fueten, F. and S. Thompson (2000). Petrographic image edge detection using artificial neural nets. In preparation.

Gonzalez, R.C. and R.E. Woods (1992). *Digital Image Processing.* Addison-Wesley.

Goodchild, J.S. and F. Fueten (1998). Edge detection in petrographic images using the rotating polarizer stage. *Computers and Geosciences* **24**, 745–751.

Harris, C. and B. Buxton (1996). Evolving Edge Detectors with Genetic Programming. In: *Proc. Genetic Programming 1996* (J.R. Koza *et al.*, Ed.). MIT Press. pp. 309–314.

Koza, J.R. (1992). *Genetic Programming.* MIT Press.

Lucier, B.J., S. Mamillapalli and J. Palsberg (1998). Program Optimization for Faster Genetic Programming. In: *Proc. Genetic Programming 1998* (J.R. Koza *et al*, Ed.). Morgan Kaufmann. pp. 202–207.

Montana, D.J. (1995). Strongly Typed Genetic Programming. *Evolutionary Computation* **3**(2), 199–230.

Poli, R. (1996). Genetic Programming for Image Analysis. In: *Proc. Genetic Programming 1996* (J.R. Koza *et al.*, Ed.). MIT Press. pp. 363–368.

Ross, B.J. (2000). The Effects of Randomly Sampled Training Data on Program Evolution. In: *Proc. GECCO 2000* (D. Whitley *et al.*, Ed.). Morgan Kaufmann.

Ross, B.J., F. Fueten and D.Y. Yashkir (1999). Automatic Mineral Identification Using Genetic Programming. Technical Report CS-99-04. Brock University, Dept. of Computer Science.

Thompson, S., F. Fueten and D. Bockus (1999). Mineral identification using artificial neural networks and the rotating polarizer stage. In: *Proceedings Geovision 99: International Symposium on Imaging Applications in Geology.* pp. 225–228.

Winkeler, J.F. and B.S. Manjunath (1997). Genetic Programming for Object Detection. In: *Proc. Genetic Programming 1997* (J.R. Koza *et al*, Ed.). Morgan Kaufmann. Stanford University, CA, USA. pp. 330–335.

Zongker, D. and B. Punch (1995). *lil-gp 1.0 User's Manual.* Dept. of Computer Science, Michigan State University.