Brock University

Department of Computer Science

Automatic Mineral Identification Using Genetic Programming

B.J. Ross[†], F. Fueten[‡], D.Y. Yashkir[†]
Technical Report # CS-99-04
December 1999

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

Automatic Mineral Identification Using Genetic Programming

B.J. Ross[1], F.Fueten[2], D.Y. Yashkir[1]
[1]Department of Computer Science [2]Department of Earth Sciences
Brock University
St. Catharines, Ontario, Canada L2S 3A1
ph: (905) 688-5550 ext. 4284
fax: (905) 688-3255
email: bross@cosc.brocku.ca

Abstract - Automatic mineral identification using evolutionary computation technology is discussed. Thin sections of mineral samples are digitally photographed using a computer-controlled rotating polarizer stage on a petrographic microscope. A suite of image processing functions is applied to the images. Filtered image data for identified mineral grains is then selected for use as training data for a genetic programming system, which automatically synthesizes computer programs that identify these grains. The evolved programs use a decision tree structure that compares the mineral image values with one other, resulting in a thresholding analysis of the multi-dimensional colour and textural space of the mineral images.

Index terms - mineral classification, genetic programming, feature space thresholding.

1 Introduction

Most rocks are composed of microscopic-sized minerals that can be identified by a variety of methods. The most common method of manual mineral identification involves placing a thin section, a 30 µm thick slice of the rock mounted on a plate of glass, under a specialized microscope. At that thickness light will pass through the rock and a geologist can examine the minerals under different lighting conditions, involving one or two polarizing filters. Competent mineral identification is a task normally requiring years of expertise. An expert will use various visual cues, for example, colour, texture, and the interaction of the crystal lattice with different directions of polarized light. A geologist experienced in microscopic mineral identification can identify grains with a great deal of accuracy. While a number of other techniques for mineral identification are available (e.g. microprobe, X-ray diffraction), the manual use of thin sections is still in many cases the fastest and cheapest method available.

Given the challenges inherent in mineral identification, the use of computer-aided tools for automatic mineral identification is worth consideration. This research will illustrate how computer vision and artificial intelligence technologies can be used to automatically identify mineral grains from digital images. The fields of image processing, computer vision, and artificial intelligence have reached the point of maturity where they can offer practical solutions to many significant real-world problems.

Automatic mineral identification is a multiple stage process. Thin sections of mineral samples are digitally photographed using a computer-controlled rotating polarizer petrographic microscope. These images are processed through a series of image processing filters, which result in a set of filtered images that encapsulate the original image features in more compact and convenient representations. One of these filtered images consists of grain edges, in which a border graphically surrounds each grain. When these grain boundaries are overlaid onto the other filtered images, it is immediately apparent that identifiable combinations of filtered image features characterize particular types of mineral specimens. These features are similar to those used by geologists when manually identifying a mineral from a thin section. Therefore, once the image features are obtained for the grains in a sample, that information should be sufficient for identifying the type of minerals comprising those grains.

Once the image data for a set mineral images is extracted and filtered, it is put into a database of grain specimens. The problem then becomes one of characterizing minerals based on their data within this database. This is a classical problem for artificial intelligence, and a variety of AI technologies are pertinent to its solution. First, it can be viewed as a classification problem, in which the features of a presented object are examined and used to assign the object to one of a predefined set of classes (Berry and Linoff 1997). Classification problems typically use well-defined descriptions of the classes, and a training set of examples of these classes. With respect to mineral identification, a set of grain data from the database can be used to derive descriptions of grain characteristics, for use in deriving a model for mineral identification.

The problem can also be viewed as one in inductive concept learning, in which a generalization is derived from a finite set of examples (Luger and Stubblefield 1998). The task for inductive learning algorithms is that useful generalizations be derived from the training examples. Over-generalization means that imprecise and inaccurate descriptions will result, while under generalization will derive models that are too tightly bound to the training set's idiosyncrasies to be of general use for other mineral instances. Much machine learning research has been undertaken in inductive learning, the most famous example being Quinlan's ID3 inductive learning algorithm (Quinlan 1986).

Another pertinent machine learning paradigm is evolutionary computation and genetic algorithms (GA) (Holland 1992). Machine learning problems are characterizable as search problems, in which a possibly infinite space of model descriptions is searched in an attempt to find a description that best characterizes a set of training data. The evolutionary computation approach to machine learning uses a simulation of natural Darwinian evolution to efficiently search a space of model descriptions in quest for one that satisfies the problem requirements. Due to a GA's unique characterization of search, such as a highly compositional genomic representation of models and reproduction operators which derive new hypotheses by combining features of existing hypotheses, the resulting search is highly parallel and very efficient.

This research solves the mineral identification problem using a specialized branch of GA called genetic programming (GP) (Koza 1992). GP is essentially a GA in which models are denoted as conventional computer programs. As the search proceeds, increasingly more useful programs are built, until hopefully one evolves that solves the problem specification to a desired level of performance. Because of this denotation of hypotheses as programs, GP is a true instance of automatic programming, in which computers program themselves without human intervention to solve some problem of interest.

In this paper, GP is used to automatically synthesize computer programs that identify minerals from their grain features. The programs to be evolved will take as input the filtered image data for a grain, examine and process that data, and then respond with an identification of the mineral represented by that grain data. More precisely, each program synthesized is specialized for identifying one specific mineral that can appear in a thin section sample. When one of these programs is supplied a set of data for a grain, that program will positively or negatively identify if that grain corresponds to the mineral the program is engineered to recognize. Therefore, all the grains in a thin section can be identified if the grain data is given to each identification program in succession.

An outline of the paper is as follows. Section 2 is a background discussion of the mineral identification problem and the rotating polarizer stage microscope. Section 3 discusses the image processing performed on the polarized images. The essentials of genetic programming are reviewed in section 4. The application of genetic programming towards mineral identification is discussed in section 5. This includes a discussion of evolutionary strategies and parameters used, and the results obtained. A critical discussion concludes the paper in section 6, which includes a comparison to other research employing neural networks for mineral identification.

2 Mineral Identification

The petrographic microscope is a commonly used tool for manual mineral identification in thin sections; however, its use for image processing has been limited (Fabbri 1984; Launeau et al. 1990; Pfleiderer et al. 1992; Petruk 1989; Starkey and Samantary 1993). Minerals are observed primarily under two different lighting conditions. In plane light a single polarizing filter is placed below the thin section, while under cross polarized illumination one polarizer is placed above and one below the sample, with their polarizing directions at 90° to each other. While a number of specialized tests can be performed on the sample, an experienced operator can often identify minerals by the nature of the change in appearance of the mineral as the as the sample is rotated with respect to the fixed polarizers. The use of optical automated mineral identification systems has been limited is due to several problems. In plane-polarized light many minerals are colourless, which also makes it impossible to distinguish grain boundaries between two adjacent colourless grains. Similarly, in cross-polarized light the colour, referred to as interference colour, displayed depends on the mineral type, the orientation of the crystal lattice of the grain with respect to the polarizers and the thickness of the thin section. Hence, the appearance of a single grain may vary from black to brightly coloured, depending on the orientation of its lattice to the polarizers. This problem of is overcome by rotating the microscope stage with respect to the fixed polarizers and observing the full range of colours a grain undergoes. 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. Hence automated mineral identification systems (Marschallinger 1997; Launeau et al. 1994) are based on scanned images and use the natural colour of the mineral.

The rotating polarizer microscope stage (Fueten 1997) was specifically designed as an addition to the standard petrographic microscope, to overcome some of its inherent problems. The stage works in conjunction with a video capture board and allows the thin section to remain fixed while the polarizers are rotated. This greatly enhances the usage of standard image processing techniques on thin sections for the segmentation, measurement and mineral identification.

3 Image Processing of Polarized Thin Section Images

Sampling of minerals is performed by repeatedly grabbing a frame, extracting data, and rotating the polarizers. The normal sampling procedure rotates the polarizers from 0° to 180° in 0.9° increments (200 steps) under both plane and crossed polarized light. A composite data set is then constructed from this information. An example thin section image is in Fig. 1, which has the maximum plane intensity for a specimen.

3.1 Data Collected Using Crossed-Polarized Light

The maximum intensity image contains the maximum intensity values as defined in Hue, Saturation, and Intensity (HSI) space. These correspond to the maximum interference colour every pixel attained during a 180° rotation of the polarizers. Intensity variations are directly related to the orientation of the crystal lattice with respect to the plane of the thin section. Variations in intensity due to the orientation of the polarizers, which are seen in single images, have been eliminated by the sampling procedure. The maximum position image is a grayscale image (with a 0 – 200 intensity range) indicating the polarizer position at which each pixel reached the maximum intensity. Position values record the orientation of the polarizing filters, ie. the step number, when a pixel reaches its maximum value. The gradient image is a representation of the accumulated sum of the maximum difference in intensity between the horizontal and vertical directions. The image is then rescaled to a 0-255 range (for display purposes) and serves as the input of the segmentation routine as presented by (Goodchild and Fueten 1998).

3.2 Data collected using Plane Polarized Light

A maximum intensity image is obtained under plane polarized light. In addition, a minimum intensity image and a minimum position image are constructed similarly to the maximum intensity and position images under crossed-polarized light. The advantage to having both a maximum and minimum intensity images is that they can be used to obtain a measure of pleochroism, the variation in colour of a mineral during plane light rotation.

3.3 Parameter extraction

It is relatively easy for the human brain to distinguish between different minerals by looking at qualitative features, but it is not obvious which numerical measurements can accomplish the same task. While their shape can on occasions identify minerals, this is not generally the case. Hence, it was decided that identification should restrict itself to visual parameters, such as colour and texture that can be extracted from each mineral. Mineral identification has to proceed on a grain by grain rather than a pixel-by-pixel basis, hence the images were segmented and the parameters were calculated on the basis of the pixels contained within a grain. A variety of colour and texture parameters were computed from the data set.

3.3.1 Colour Parameters

Each image is captured using red, green and blue (RGB) components. However, experimentation determined the RGB colour model was very sensitive to fluctuations of the light source. Converting the RGB components to hue, saturation and intensity space (HSI) detaches the intensity component from the colour information and reduces the effects of variable lighting. The hue component provides a measurement that directly determines the colour of the grain, while the saturation yields the depth of that colour.

Pleochroism is a fundamental mineral characteristic that provides important information during manual mineral identification. Hence, a pleochroism parameter was defined as the difference between the modes of the intensity of the maximum intensity and the minimum intensity of the plane light images. For simplicity sake, any changes in the hue or saturation between the two plane light intensity images were ignored.

3.3.2 Texture parameters

Colour is not a uniquely identifying characteristic of minerals alone. Several characteristics of minerals require special tests, which involve the insertion of a special lens or filter. However an experienced petrologist can in many cases identify minerals using subtle colour-independent features such as undulatory extinction or small amounts of alteration without having to resort to specialized tests. Undulatory extinction which results from a bend lattice and results in a series of dark bands crossing the crystal during rotation would not manifest themselves in colour images, but would appear in position images and can be quantified by textural parameters successfully used texture parameters to classify rock textures for a variety of purposes (Autio et al. 1999).

The intensity component of the colour images and the values of position images were used to calculate four standard texture parameters. Texture parameters are calculated using a co-occurrence matrix that is calculated for each grain. A co-occurrence matrix, representing a two dimensional histogram, defines a P[i,j] value which corresponds to the number of pixels with the values i and j separated by the displacement vector (1,1). This displacement expressed as an image mask is the 2x2 identity matrix. Using this normalized matrix (P) the following measurements are calculated (Jain et al. 1995):

$$\text{Contrast} = \Sigma\Sigma\, (i - j)^2\, P[i,j]$$
$$\text{Entropy} = -\Sigma\Sigma\, P[i,j] \log(P[i,j])$$
$$\text{Energy} = \Sigma\Sigma\, P^2[i,j]$$
$$\text{Homogeneity} = \Sigma\Sigma\, P[i,j] / (1 + | i - j |)$$

A total of 35 colour and texture parameters (Table 1) were extracted from the data set.

4 Genetic Programming

Genetic programming (Koza 1992; Banzhaf et al. 1998) is an evolutionary computation paradigm in which programs are evolved using a genetic algorithm (Holland 1992). The basic GA is given in Fig. 2. GA and GP are characterized by the following:

(i) An initial population of random individuals, which in the case of GP are randomly constructed programs;

(ii) A finite number of generations, each of which results in a new or replenished population of individuals;

(iii) A problem dependent fitness function, which takes an individual and gives it a numeric score indicative of that individual's ability to solve a problem at hand;

(iv) A fitness proportional selection scheme, in which programs are selected for reproduction in proportion to their fitness;

(v) Reproduction operations, usually the crossover and mutation operations, which take selected program(s) and generate offspring for the next generation.

The essential difference between GP and GA is the denotation of individuals in the population. A pure GA uses genotypes that are fixed-length bit strings, and which must be decoded into a phenotype for the problem being solved. A GP uses a variable-length computer program genotype, which is directly executable by some interpreter (typically LISP). The use of programming code as the genotype is powerful, as its generality permits a vast number of problems to be solved. It also means that GP is the closest realization yet for true automatic programming, in which computers program themselves after being given a description of a problem in which to solve.

The two main reproduction operators used in GP are crossover and mutation. Crossover is the more important of the two operators, as it permits the genetic combination of program code from programs into their offspring, and hence acts as the means for inheritance of desirable traits during evolution. Crossover takes two selected programs, finds a random *crossover point* in each program's internal representation (normally a parse tree), and swaps the subtrees at those crossover points. The crossover points must be selected so that the resulting offspring are syntactically correct. With the program structure in (Koza 1992), this means that if a subtree with a nonterminal root (function label) is selected in one program, it is swapped with a subtree with a nonterminal root from the other program.

Mutation is not as commonly performed in GP. When it is used, a random mutation point is found in a selected program, and the subtree at that node is replaced with a new, randomly generated tree. This is the means by which new genetic traits can be introduced into the population during evolution. Although crossover and mutation preserve the grammatical integrity of programs, the user must ensure *closure* - that the resulting programs are always executable. So long as closure is maintained, all programs derivable by the GP system will be executable by the fitness function, and hence their fitness will be ascertainable.

5 Evolution of Mineral Identification Programs

5.1 Data Preparation

As discussed in section 3, the raw polarized images from the rotating polarizer stage microscope are processed through a number of filters, resulting in a set of 35 distinct data parameters. These parameters are computed for grain within the images. In order to extract the training data, a geologist manually inspected and identified a series of minerals from a set of thin sections. The data vectors of different minerals and their identification are then saved in a grain database. This information is used by the GP fitness evaluation procedure.

The grain data is taken from a set of 44 petrographic thin sections. Some of the sparser minerals on the thin sections could not be used, since there were not enough examples to process by the GP system. The minerals species and number of each examples of each mineral examined here are:

1. Quartz (575)
2. K-spar (369)
3. Plag (358)
4. Biotite (74)
5. Muscovite (36)
6. Horneblende (86)
7. CPX (81)
8. Olivine (685)
9. Garnet (311)
10. Calcite (520)
11. Opaque (189)

Some of these minerals had fewer example grains as others. This affects the ability to run the GP system on them for larger sets of training examples.

5.2 Target Languages

An important preparation step for GP is to identify a suitable target language in which to evolve programs. The success of evolution is dependent upon the use of a language that can adequately represent a solution to the problem being solved. Should an inadequate language be used, there will be no means for the GP system to evolve a solution. In addition, higher-level languages can sometimes enhance evolution efficiency, since higher-level features can promote faster evolution of better quality programs. This must be balanced with efficiency of execution, however: high-level functions may demand computational overhead, and this will slow the GP system considerably.

Two different languages were used for evolving mineral identification programs. The Backus-Naur grammars for the languages are in Fig. 3. These grammars are translated into a LISP-like notation for use by the GP system. The first language, L1, is a nested decision tree language, consisting of nestable *If-Then* statements, a relational operator “<” over the 34 parameter values, and the logical constants *true* and *false*. Both conditional *if* and relational “<” statements are used, as the latter permits decision trees to be leaner in size than if only nested “if”’s were allowed. As can be seen in the grammar, an L1 program will performing a relational test on the parameter data, and will return a logical value as a result. The intention is for a mineral identification program written in L1 to return true if the grain parameters being analyzed conform to the particular mineral that program is specialized to identify, and false otherwise.

The second language, L2, is a more complex language. It is a superset of L1, introducing mathematical expressions over parameter values. Most of L2 is self-explanatory. The *min* and *max* operators return the minimal and maximal value of their pairs of arguments. All equality tests within relational expressions work with a degree of error within 10% of the left-hand parameter. For example, in the expression "a = b", if b is within 10% of a's value, the expression returns true. All the operators must observe closure: every operator should execute with any possible argument value. The only operator this affects is the "/" division operator, which must handle zero-value denominators. It is coded to return '0' in these instances. Ephemeral constants are floating point numbers between 0.0 and 1.0, which are initially generated with a random number generator, but thereafter retain their initialized values throughout their existence.

5.3 Fitness Strategy

Each program to be evolved is to identify one of the mineral types resident on the thin section images. Let this mineral be *M*. Once a mineral is selected as a goal for evolution, the grain database is segregated into two sets of data: training data and testing data. Training data are the grain feature vectors (henceforth called "grains"), which will be used during GP evolution by the fitness evaluator. Let N be the total number of minerals. The training set for a mineral consists of a random selection of K grains corresponding to the mineral being identified, plus a random selection of K grains for each of the other minerals. This yields a total of N*K grains to be used as training examples.

The fitness function used during evolution is the following:

$$\frac{1}{2}\left(\frac{\#\,\text{hits correct}}{K}+\frac{\#\,\text{misses correct}}{(N-1)K}\right)$$

A *hit* is when the program-identifying mineral M identifies a grain as belonging to M, or in other words, the program returns *true*. Similarly, a *miss* is when the program returns *false*. This formula therefore computes the percentage of correctly identified grains of mineral M, and the percentage of grains correctly identified as not being mineral M. The sum of these terms is scaled by 1/2, to keep the range of values between 0 and 1. This implies that a value of 1 means a program has perfectly identified all the grains in the training set, while 0 implies that no grains were correctly identified. Note that the equation is weighted towards awarding correct hits. This is because there are far fewer positive examples than negative examples in the training set. Otherwise, if all the grains were evaluated with the same weight, a program that returned false for all grains would have a relatively high score of (N-1)/N percent, which negatively affects evolution.

The testing set is used to measure the performance of programs on new data not seen during training. The testing set is a good way to ensure that a program has not been over trained to recognize only its training data, and also to verify that the training data is representative of the problem at hand. The testing set is defined to be the rest of the grain database not used for training. The testing formula is similar to the fitness formula:

$$\frac{1}{2}\left(\frac{\#\,\text{hits correct}}{P}+\frac{\#\,\text{misses correct}}{Q}\right)$$

P is the total number of grains for mineral M in the testing set, and Q is the total number of other grains. Again, this formula is weighted towards favouring correct hits, because there may be fewer grains for M than the rest of the minerals. Also note that the testing set does not necessarily have a balanced number of grains for different minerals, since it is defined to be whatever is left in the database after the training set has been extracted. Since some mineral specimens are far more rare than others, the testing measurements for various minerals should not be directly compared with one another. The important relationship to note is how well the testing and training values correlate with each another.

5.4 Other Experiment Details

Table 2 lists other parameters for the GP experiments. Most of these parameters are well known from the GP literature, and will not be discussed further. The terminals and nonterminals depend upon whether language L1 or L2 is used (see Fig. 3).

Two sets of experiments were undertaken. The first set of runs evolved identification programs for all the minerals, and used a minimal training set of 34 positive examples and 340 negative examples per run. A set of training data for a single run was extracted pseudo-randomly from the database. The second set of runs doubled the number of positive examples to 70. In doing this, four of the eleven minerals (biotite, muscovite, horneblende, CPX) had to be excluded, because they insufficient numbers of positive examples in the grain database to permit accurate training and testing.

The GP system used is the lil-gp 1.1 system (Zongker and Punch 1995). The version used is a patched one by Sean Luke, which adds strong typing. Typing permits a more constrained search during evolution, as well as less stringent requirements for closure for languages L1 and L2 (Montana 1995). The system is compiled with GNU C on the Silicon Graphics IRIX 6.3 operating system. This C implementation of GP was chosen for its efficiency for processing the large example sets in this application.

5.5 Results

Table 3 shows the final results of the GP runs. There were 20 runs undertaken for each GP experiment (mineral type, L1 or L2, 34 or 70 positive grain examples). For comparison, testing performance values for the same data by a neural network (Thompson et al. 1999) is given in the last.

The more complex language L2, with its additional floating-point arithmetic, did not lend any great advantages to the quality of solutions evolved in the simpler decision language L1. Although 7 of the best mineral identifiers in the table are written in L2, their scores are not outstandingly better than the equivalent L1 solutions. L2 was in fact more expensive to run during the GP runs, as the floating-point arithmetic is computationally more expensive to execute. Hence L2 runs took between two to three times longer to execute than L1 runs.

Using more positive grain examples was normally advantageous for the average quality of solutions obtained. However, in most of the cases for L1, the solutions obtained for K=34 were marginally better than those for K=70. This is probably a combination of the particular samples of grain used in these cases being more reflective of the entire grain set as a whole, as well as inherent statistical margins of error. This phenomenon did not occur with L2. In any case, the differences in errors between the values for K=34 and K=70 are not exceptional, which implies that the examples selected for K=34 were representative of the database as a whole.

Another factor affecting the overall performance is that the grain database itself had errors. A geologist manually identified the thousands of grains in this database, and some grains may have been misidentified. When such an erroneous grain happens to be selected for training, it can introduce noise to the evolution process. This is especially acute when such a misidentified grain is blatantly different in visual characteristics to the rest of the mineral for which that grain belongs. Obviously, this problem is more pronounced when smaller training sets are used.

One phenomenon observed was that the runs using K=34 often suffered from over training effects. In Fig. 4, it can be seen that the testing fitness declines after approximately generation 15, while the training fitness continue to improve. This implies that the population is not correctly generalizing from the training set. Over training did not occur for the runs with K=70.

A variety of different styles of programs were evolved. One example L2 program (K=70) that identified garnet with a testing performance of 97.6% is the following simple program:

```
p3 < (p16-p21)*p16
```

This program has 7 nodes and a depth of 3 (see Table 1 for the parameters denoted in the expression). This program is unusually concise compared to others in the runs for this set of experiments, which normally had tree sizes averaging 222 nodes (the largest tree had 500 nodes). For example, another program in this set of runs with 93 nodes and a testing performance of 98.0% is:

```
if ((((0.71899/p35)-(0.71899/p35)) < (p16*p14+p21))
                and ((p3+p17) < (p16*p14+max(p25,p32)))
                      and ((p5+p10) < (max(p26,p7)-(p34-p23)))
            or (not (False or True)))
then (max((min((p10-p26),(p3+p10+min(p2,p26)))+(p5+p10)*p6*p6),
               min(p24,p12)) > p8)
         and (min(p2,p26) < min(p9,p8))
else (((p3+p10+min(p2,p26))-(p26*p6*p6)) < (p33/p7))
```

As is often to be expected with GP results, it is difficult to intuit the logic used in this program. Also note that there is some intron material ("junk DNA") that can be simplified. For example, the expression "(not (or False True))" simplifies to "False", which in turn can be used to simplify its parent expression.

Intron material is often prevalent in some solutions, which is evident when a superfluous number of unnecessary tests and terms exist. Consider the following 57-node L1 solution for identifying K-spar (K=34, testing fitness 90.1%):

```
if p3 < p2
         then if p3 < p2
                  then (p21 < p33)
                  else if p26 < p26
                           then (p2 < p25)
                           else (p21 < p33)
         else if p25 < p2
                  then False
```

```
        else if p23 < p2
            then (if p4 < p16 then False else False)
            else if p3 < p2
                then (p9 < p21)
                else if p2 < p26
                    then if p26 < p26
                        then (p19 < p14)
                        else if p24 < p6
                            then False
                            else if p25 < p2
                                then False
                                else True
                    else (p9 < p21)
```

Removing redundant tests and simplifying expressions can remove 34 nodes:

```
if p3 < p2
    then (p21 < p33)
    else if p25 < p2
        then False
        else if p23 < p2
            then False
            else if p2 < p26
                then (p6 < p24)
            else (p9 < p21)
```

It is possible to implement an automatic simplifier that does the above transformation.

6 Discussion

This research successfully used genetic programming to evolve mineral identification programs for use with images obtained with the rotating polarizer microscope. The overall performance of the best mineral identifiers ranged from between 86% to 98%. These programs will be incorporated into an interactive workstation environment, which will automatically identify grains as selected by the user.

The GP performance is comparable to that obtained with a neural network for mineral identification (Thompson et al. 1999). A direct comparison of these two different approaches is difficult to do, given the differences inherent between neural nets and genetic programs. The GP identifiers are specialized for each mineral, and hence are evolved separately. On the other hand, the NN is trained to identify all the minerals with one single network. The NN takes the 35 image parameters as input, as well as a code that identifies the mineral specimen for that grain. It also has 16 outputs, with one output per mineral specimen. The weights generated on these outputs identify the type of mineral corresponding to the input pattern. Hence the neural network is a "universal identifier" for the entire set of minerals.

To determine whether a universal mineral identifier could be evolved with GP, another GP experiment was undertaken. Rather than generate a logical true or false if a grain parameter matches a particular mineral, as was done earlier, these new programs would generate a label corresponding to the mineral matching the parameter data. A set of terminals corresponding to the set of mineral names was given to the system, and a training strategy similar to that used before was tried. These runs were unsuccessful in producing good quality identifiers, and the best solution obtained had a testing performance of 56.4% (training=46.5%). This is still better than what a random identifier might be expected to perform (testing=9%), but not nearly as accurate as the dedicated mineral identifiers. Part of the problem with this "universal identifier" approach is that it is difficult for evolution to ensure that all the mineral labels themselves are retained in the single program. During many runs, many mineral labels were lost throughout the population, which automatically prohibited the identification of those minerals by the programs in later generations. When this is added to the basic problem of evolving a useful identification strategy, the search becomes too challenging for GP. Hence the strategy undertaken in this paper of evolving individual mineral identifiers is the most sensible one for GP.

To our knowledge, this is the only work that uses evolutionary computation to identify minerals. The use of a neural network for mineral identification was described earlier (Thompson et al. 1999). There are many examples of work elsewhere applying GP to pattern classification (Zhao et al. 1996; Abramson and Hunter 1996; Gray et al. 1996; Ryu and Eick 1996) and inductive learning (Freitas 1997; Siegel 1994).

Acknowledgements
B. Ross and F. Fueten are supported by NSERC operating grants. D. Yashkir is supported by an NSERC undergraduate research grant.

References

Abramson M, Hunter L (1996) Classification using cultural co-evolution and genetic programming. In: Koza JR et al. (ed) Genetic Programming 1996: Proceedings of the First Annual Conference. MIT Press, Cambridge, MA, pp. 249-254.
Autio J, Rantanen L, Visa A, Lukkarinen S (1999) The classification and characterisation of rock using texture analysis by co-occurrence matrices and the Hough transform. Proceedings Geovision 99: International Symposium on Imaging Applications in Geology. University of Liege, Belgium, pp. 5-8.
Banzhaf W, Nordin P, Keller RE, Francone FD (1998) Genetic Programming – An Introduction. Morgan Kaufmann, San Francisco.
Berry MJA, Linoff G (1997) Data Mining Techniques. John Wiley & Sons, New York.
Fabbri AG (1984) Image processing of geological data. Van Nostrand-Reinhold, Wokingham, UK.
Freitas AA (1997) A genetic programming framework for two data mining tasks: Classification and generalized rule induction. In: Koza JR et al. (ed) Genetic Programming 1997: Proceedings of the Second Annual Conference. Morgan Kaufmann, San Francisco, pp. 96-101.
Fueten F (1997) A computer controlled rotating polarizer stage for the petrographic microscope. Computers and Geoscience. 23:203–208.
Goodchild JS, Fueten F (1998) Edge detection in petrographic images using the rotating polarizer stage. Computers and Geosciences. 24:745-751.
Gray HF, Maxwell RJ, Martinez-Perez I, Arus C, Cerdan S (1996) Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra. In: Koza JR et al. (ed) Genetic Programming 1996: Proceedings of the First Annual Conference. MIT Press, Cambridge, MA, p. 424.
Holland JH (1992) Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA.
Jain R, Kasturi R, Schunk BG (1995) Machine Vision. McGraw Hill.
Koza JR (1992) Genetic Programming – On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA.
Launeau P, Bouchez J-L, Benn K (1990) Shape preferred orientation of object populations: automatic analysis of digitized images. Tectonophysics. 180(2-4):201-211.
Launeau P, Cruden CA, Bouchez J-L (1994) Mineral recognition in digital images of rocks: A new approach using multichannel classification. The Canadian Mineralogist. 32:919-933.
Luger GF, Stubblefield WA (1998) Artificial Intelligence – Structures and Strategies for Complex Problem Solving. Addison-Wesley, Reading, MA.
Marschallinger R (1997) Automatic mineral classification in the macroscopic scale. Computers and Geosciences. 23:119-126.
Montana DJ (1995) Strongly typed genetic programming. Evolutionary Computation. 3(2):199-230.
Petruk W (1989) Shortcourse on image analysis applied to mineral and earth sciences. Mineralogical Association of Canada Shortcourse Handbook. no 16.
Pfleiderer S, Ball DGA, Bailey RC (1992) AUTO: A computer program for the determination of the two-dimensional auto-correlation function of digital images. Computers and Geosciences. 19(6):825-829.
Quinlan JR(1986) Induction of Decision Trees. Machine Learning. 1(1):81-106.

Ryu T-W, Eick CF (1996) MASSON: discovering commonalities in collection of objects using genetic programming. In: Koza JR et al. (ed) Genetic Programming 1996: Proceedings of the First Annual Conference. MIT Press, Cambridge, MA, pp. 200-208.
Siegel EV (1994) Competitively evolving decision trees against fixed training cases for natural language processing. In: Kinnear KE Jr (ed) Advances in Genetic Programming. MIT Press, Cambridge, MA.
Starkey J, Samantary AK (1993) Edge detection in petrographic images. Journal of Microscopy. 172(3):263-266.
Thompson S, Fueten F, Bockus D (1999) Mineral identification using artificial neural networks and the rotating polarizer stage. Submitted for publication.
Zhao J, Kearney G, Soper A (1996) Emotional expression classification by genetic programming. Late Breaking Papers at the Genetic Programming 1996 Conference. pp. 197-202.
Zongker D, Punch B (1995) lil-gp 1.0 User's Manual. Michigan State University, Department of Computer Science.

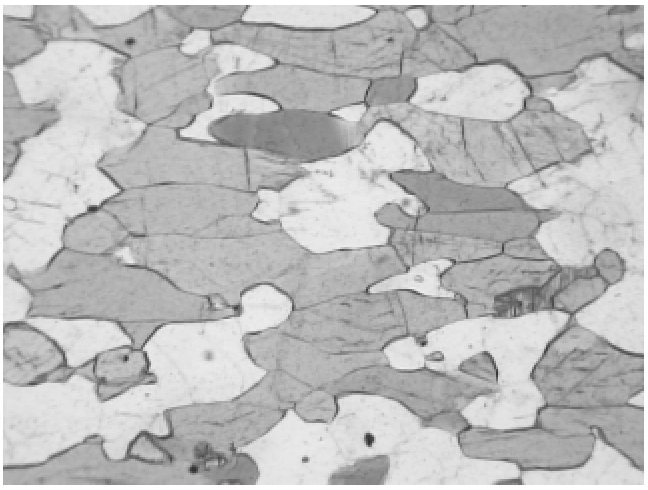

Fig. 1: Maximum Plane Intensity

Maximum crossed intensity:
- p1. hue
- p2. saturation
- p3. intensity
- p4. contrast
- p5. entropy
- p6. energy
- p7. homogeneity

Maximum plane intensity:
- p8. hue
- p9. saturation
- p10. intensity
- p11. contrast
- p12. entropy
- p13. energy
- p14. homogeneity

Minimum plane intensity:
- p15. hue
- p16. saturation
- p17. intensity
- p18. contrast
- p19. entropy
- p20. energy
- p21. homogeneity

Maximum crossed position:
- p22. contrast
- p23. entropy
- p24. energy
- p25. homogeneity

Minimum Plane Position:
- p26. contrast
- p27. entropy
- p28. energy
- p29. homogeneity

Gradient:
- p30. intensity
- p31. contrast
- p32. entropy
- p33. energy
- p34. homogeneity

Pleochroism:
- p35. pleochroism

Table 1: Image parameters

```
1. Initialize: Generate initial population.
2. Evolution:
Loop while current generation < maximum generations
    and fitness of best individual not considered a solution {
    Loop while new population size < maximum population size {
        - Select a genetic operation probabilistically:
        --> Crossover:
            - Select two individuals based on fitness.
            - Perform crossover.
            - Evaluate the fitness of the offspring.
            - Add offspring to new population.
        --> Mutation:
            - Select one individual based on fitness.
            - Perform mutation.
            - Evaluate the fitness of the offspring.
            - Add offspring to new population.
    }
    - Increment generation counter.
}
3. Output: Print best solution obtained.
```

Fig. 2: Genetic algorithm

```
L1:  Bool     ::= if Param < Param then Bool else Bool | Param < Param
                    | true | false
     Param    ::= p1 | p2 | ... | p35 | <ephemeral const>

L2:  Bool     ::= if Bool then Bool else Bool | Bool and Bool | Bool or Bool
                    | not Bool | E BoolRel E | true | false
     BoolRel  ::= < | <= | > | >= | =
     E        ::=  E Op E | max(E1, E2) | min(E1, E2) | Param
     Op       ::= + | - | / | *
     Param    ::= p1 | p2 | ... | p35| <ephemeral const>
```

Fig. 3: Grammars for GP Languages

Parameter	**Value**
Terminals, Nonterminals	(see Fig. 1)
Fitness function	Number of correct identifications
Training set size	(i) 34+, 340- (ii) 70+, 420-
Generation type	Generational
Selection scheme	Tournament, size=7
Population size	500
Maximum generations	50
Runs per experiment	20
Initial population tree size	Ramped half&half, $2 \leq size \leq 6$
Maximum tree size	17
Probability crossover	0.95
Probability mutation	0.05
Probability internal (external) crossover	0.1 (0.9)

Table 2: GP parameters

		L1				L2				NN
		K=34		K=70		K=34		K=70		K=74
Mineral		test	train	test	train	test	train	test	train	test
Quartz	best:	90.2	94.6	92.2	93.8	93.5	97.5	97.8	96.1	96.5
	avg:	84.7	90.9	85.7	88.9	88.3	93.7	90.1	92.5	
K-spar	best:	94.2	97.8	93.9	96.0	93.8	99.0	93.4	96.9	88.6
	avg:	89.7	95.9	91.0	93.5	90.7	97.8	90.8	93.9	
Plag	best:	86.2	91.8	84.7	93.8	83.3	95.7	85.4	94.3	89.1
	avg:	75.9	88.0	78.9	88.9	78.6	91.8	80.2	91.0	
Biotite	best:	98.1	98.1	–	–	96.6	99.4	–	–	97.3
	avg:	93.7	96.4			91.7	98.0			
Muscovit	best:	98.3	96.6	–	–	97.6	99.9	–	–	–
e	avg:	74.8	94.1			78.0	97.9			
Horne-	best:	91.4	97.7	–	–	92.8	98.5	–	–	95.4
blende	avg:	86.8	94.7			86.1	95.8			
CPX	best:	90.3	96.5	–	–	93.8	98.5	–	–	97.5
	avg:	81.0	88.1			84.9	92.8			
Olivine	best:	90.0	94.4	89.9	93.8	91.4	96.6	93.2	98.1	92.8
	avg:	83.0	88.6	86.6	90.0	84.4	92.6	89.4	95.1	
Garnet	best:	98.2	99.7	98.9	99.6	97.8	100	99.4	99.8	97.4
	avg:	96.0	98.3	97.6	98.1	95.8	98.9	97.1	98.6	
Calcite	best:	92.1	97.4	91.8	95.2	92.8	99.0	95.1	98.2	96.0
	avg:	84.4	93.6	88.5	93.0	86.2	96.3	91.2	95.9	
Opaque	best:	97.7	99.9	97.7	99.9	98.3	100	98.1	100	95.2
	avg:	93.6	98.9	96.6	99.4	95.2	99.3	96.5	99.7	

Table 3: Results (values are percentages)

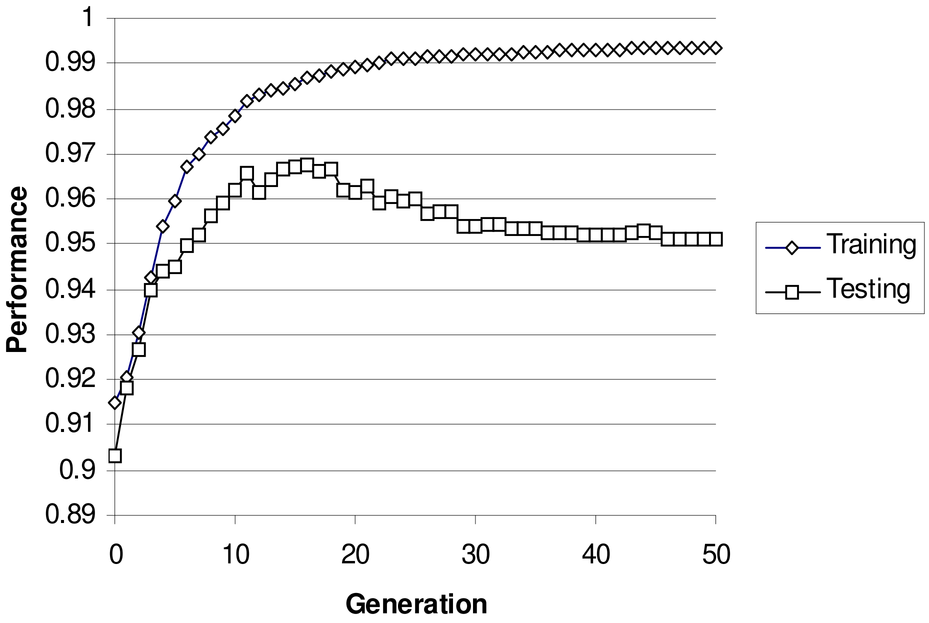

Fig. 4: Opaque evolution (L2, K=34, avg. 20 runs)