Computers, Environment and Urban Systems 33 (2009) 463–471

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/compenvurbsys

Evaluation of the use of spectral and textural information by an evolutionary algorithm for multi-spectral imagery classification

H.G. Momm[a,*], Greg Easson[b], Joel Kuszmaul[b]

[a] University of Mississippi Geoinformatics Center, 224 Lester Hall, University, MS, United States
[b] Department of Geology and Geological Engineering, 118 Carrier Hall, University, MS 38677, United States

ARTICLE INFO

Keywords:
Genetic programming
Remote sensing
Image texture
Evolutionary computation
Optimization

ABSTRACT

Considerably research has been conducted on automated and semi-automated techniques that incorporate image textural information into the decision process as an alternative to improve the information extraction from images while reducing time and cost. The challenge is the selection of the appropriate texture operators and the parameters to address a specific problem given the large set of available texture operators. In this study we evaluate the optimization characteristic of an evolutionary framework to evolve solutions combining spectral and textural information in non-linear mathematical equations to improve multi-spectral image classification. Twelve convolution-type texture operators were selected and divided into three groups. The application of these texture operators to a multi-spectral satellite image resulted into three new images (one for each of the texture operator groups considered). These images were used to evaluate the classification of features with similar spectral characteristics but with distinct textural pattern. Classification of these images using a standard image classification algorithm with and without the aid of the evolutionary framework have shown that the process aided by the evolutionary framework yield higher accuracy values in two out of three cases. The optimization characteristic of the evolutionary framework indicates its potential use as a data mining engine to reduce image dimensionality as the system improved accuracy values with reduced number of channels. In addition, the evolutionary framework reduces the time needed to develop custom solutions incorporating textural information, especially when the relation between the features being investigated and the image textural information is not fully understood.

1. Introduction

Automated and semi-automated techniques to obtain useful information from remotely sensed imagery have been researched since the early 70s as a possible alternative to reduce (or at least minimize) human interaction (Baumgartner, Steger, Mayer, Eckstein, & Ebner, 1999 and Quackenbush, 2004). The quality of the results obtained by automated and semi-automated techniques are often measured on the basis of how well they replicate the results produced through human interpretation (either from field data or from visual inspection). The development of automated and semi-automated techniques to extract information from imagery that mimics the human interpretation is a complex problem. This lies in the fact that humans tend to interpret imagery using information such as context, shape, edge, color, and texture; while most automated and semi-automated techniques are based solely on spectral information (Jensen, 1996). Research is now being conducted to recreate the human ability to classify and discriminate image features using more sophisticated computer programs designed to incorporate both spectral values (color) and image texture information into the decision-making process.

Image texture definition varies greatly among disciplines (Russ, 1999). However, there is a scientific consensus defining texture as the frequency of tonal changes or, in simple terms, as the variations in brightness values (Lillesand & Kiefer, 2000). These variations repeat in a periodical or quasi-periodical pattern. Several image texture operators were developed over the years, but unfortunately image textural analysis is an area in image processing that still lacks fundamental knowledge (Jahne, 2002). The problem is in the selection of the most appropriate textural operator (along with its parameters) to solve a particular problem. This requires the choice of domain (spatial or frequency), type of textural operator (edge detection, convolution, first, second, or third order textural statistics), window size, direction of offset (orientation), offset distance, which spectral channel to run, and others (Hall-Beyer, 2007). The selection of custom-tailored image textural operators is commonly performed using a trial-and-error approach, where the success of the final solution relies on the analyst's ability to use

* Corresponding author. Tel.: +1 662 915 5201; fax: +1 662 915 5998.
E-mail addresses: hmomm@olemiss.edu (H.G. Momm), geasson@olemiss.edu (G. Easson), kuszmaul@olemiss.edu (J. Kuszmaul).

0198-9715/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compenvurbsys.2009.07.007

iteratively commercial software packages or by developing new computer programs. In either case a skilled user is needed and the entire process can become time consuming due to the large number of possible texture parameters available.

Alternatively, the search for the optimal texture operator (or sequence of texture operators) can be viewed from an optimization standpoint, and therefore a candidate application for evolutionary computation techniques (Fogel, 2000) such as genetic algorithms (Mitchell, 1997) and genetic programming (Koza, 1992). The strength of evolutionary computation algorithms is the fact that if one can define a fitness measure for a set of candidate solutions, then the problem might be solved by evolutionary computation (Brumby et al., 1999). This provides a desirable alternative solution for the selection of the textural operator because the trial-and-error process can be optimized and the need to develop tailored solutions can be significantly reduced.

This study addresses the challenge of selecting problem-specific image texture operators by evaluating an evolutionary framework in the task of incorporating spectral and textural information for improved information extraction from multi-spectral remotely sensed imagery. The proposed framework combines evolutionary computation with standard image classification algorithms. The optimization characteristic of the evolutionary component is used as a data mining engine to select and combine through mathematical operations, the most appropriate subset of textural operators (along with its parameters) from a set of available texture operators. In this process, the search for the most favorable solution in the solution space is optimized, and, the overhead needed to develop customized solutions is reduced. The objective of this study is to evaluate the performance of the proposed evolutionary framework in evolving custom-tailored solutions, represented by mathematical operations of image textural and spectral channels, to improve the classification results of standard image classification algorithms.

2. Related work

Evolutionary algorithms have been applied successfully to a wide range of image processing problems, such as image classification (Brumby et al., 1999; Korczak & Quirin, 2003; Pal, Bandyopadhyay, & Murthy, 2001; Perkins et al., 2000; Agnelli, Bollini, & Lombardi, 2002; Lin & Bhanu 2005; and Bandyopadhyay & Maulik, 2002), feature extraction (Daida, Hommes, Ross, & Vesecky, 1995 and Krawiec & Bhanu, 2005), and inverse modeling (Momm, Easson, & Kuszmaul, 2007; Chang, Liu, & Wen, 2007; Chen, 2003; Fonlupt & Robilliard, 2000; Fang, Liang, & Kuusk, 2003; Ines & Honda 2005; and Drunpob, Chang, Beaman, Wyatt, and Slater (2005). There is also work dedicated to the use of evolutionary algorithms to incorporate image textural information into the image processing framework, as an alternative to developing custom-tailored solutions for particular applications (Daida, Bersano-Begey, Ross, & Vesecky, 1996; Harris & Buxton, 1996; Poli, 1996a; and Bhandarkar, Zhang, & Potter, 1994).

In the work described by Poli (1996a, 1996b), genetic programming was used to discover complex and problem-specific filters to solve problems of image enhancement in the medical field. The objective of Poli's work was to discover effective filters capable of accentuating specific characteristic of the image. The image filters were formed by averaging functions of local "window" operators (convolution-type operations) of different sizes. Poli justified the use of genetic programming as a unique tool to combine all available pieces of information in a non-linear and complete way. Daida et al. (1996) used genetic programming as an optimization tool to develop image processing rules based on spectral arithmetic and textural information to extract desired pattern from ERS SAR data products. In that research, four channels were added to the original dataset: two mean images with window size of 3×3 and 5×5 and two other channels of edge detection operators (Laplacian image with kernel 5×5 and Laplacian image of the mean 3×3 image also with kernel 5×5). Harris and Buxton (1996, 1997) proposed the use of genetic programming to develop one-dimensional (1D) edge detector functions for real and synthetic generated images; and Bhandarkar et al. (1994) used GA to develop an (2D) edge detector.

Research to investigate the use of evolutionary computation to select and to combine textural and spectral information (in a non-linear way) to address remotely sensed image classification problems is very limited. The textural operators in use represent only a small subset of the larger set of the available set of image texture operators. Furthermore, the majority of the previous investigations dealt with single-band images, whereas the use of textural information of multi-spectral remotely sensed images offers an extra level of difficulty. Most remotely sensed datasets are composed of multi-spectral (or hyper-spectral) images and therefore the texture operators are applied to each individual spectral band. As a result, the search space (originally made by the set of textural operators) is now multiplied by the number of spectral bands making the task of selecting the most appropriate combination of spectral and textural information more complicated.

3. Evolutionary framework

3.1. Background

3.1.1. Evolutionary computation

Evolutionary computation is a stochastic approach based on the concept of biological evolution first introduced by the work of Charles Darwin and Alfred Russel Wallace in 1858. The classic Darwinian evolutionary theory, when combined with selectionism theories and with genetics has become the Neo-Darwinian theory (Fogel, 2000). In this theory, the enormous variety of organisms living on Earth is the result of processes acting on populations of organisms and their genetic codes (Hoffman, 1989, p. 39). These processes are: reproduction, mutation, competition, and selection. According to the biological evolutionary process, over a large period of time and many generations, the individuals are able to adapt to the environment that they live in will have a greater chance to survive and therefore pass their genetic code to the next generation. Conversely, the individuals that are unable to cope with the environment that they live in will most likely die out and therefore their genes disappear (Koza, 1992).

The attempt to simulate biological evolution through computer programs comprises the field of evolutionary computation. Different algorithms were developed over the years. The four most common types of algorithms based on the evolutionary principles are evolutionary programming, evolutionary strategies, genetic algorithms, and genetic programming (De Jong, 2006). The evolutionary framework described in this manuscript uses genetic programming as the optimization and data mining engine.

3.1.2. Genetic programming

Genetic programming (GP) algorithms are designed to automatically generate computer programs through the process of *natural selection* also know as *survival of the fittest* (Koza, 1992). The overall evolutionary process is very similar to the one used in genetic algorithms; however, GP differs from genetic algorithms and other types of machine learning algorithms in its solution representation. In GP, candidate solutions for the problem are computer programs, often times expressed as mathematical expressions, internally represented as hierarchical tree structures rather than a fixed size strings in the case of genetic algorithms or a vector of weights in the case of artificial neural networks. Fig. 1 illustrates this representation. This data structure increases the complexity of

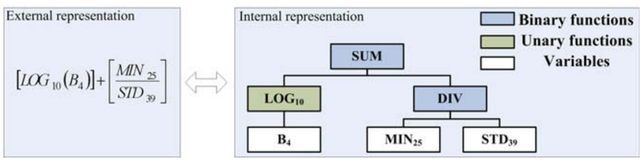

Fig. 1. Illustration of the candidate solution representation used by genetic programming. In genetic programming a candidate solution is characterized by a computer program that is internally represented by a hierarchical tree structure. In this tree representation, the nodes contain basic functions (colored squares) and the leaves the arguments used by these functions (white squares).

the system because these hierarchical trees constantly change size and shape when the structure undergoes adaptation.

The main components of a GP system are: a set of functions, a set of parameters, and a fitness function. The set of functions contains the basic building blocks which are assembled by GP to form different candidate solutions during the evolutionary process. The set of parameters are pre-defined arguments to be used by the evolved computer programs. The fitness function is problem-specific user-defined function designed to provide the means to compare and to rank individual candidate solution.

GP operates in an iterative mode by constantly updating the set of candidate solutions (also referred to as population) until the most fit solution (individual) is found. Initially, the first population is random generated and after each iteration, each candidate solution in the set of possible candidate solutions is evaluated based on the fitness function and sorted accordingly. Only the solutions with the highest fitness are then selected to be carried forward to form the next new set of candidate solutions (new generation). Some of the candidate solutions are carried forward with no change (replication) while others are susceptible to genetic operations such as mutation and crossover. Mutation is often considered a secondary operation and its main purpose is to reintroduce diversity into the population (Koza, 1992). In GP, mutation is commonly implemented by an operation that selects one solution from the set of candidate solutions with the highest fitness values, randomly selects a node of the tree (hierarchical tree representation show in Fig. 1), removes whatever is below the selected node, and inserts a new randomly created sub-tree at that point (Koza, 1992). In the crossover operation, two solutions are randomly drawn from the set of candidate solution with the highest fitness values. These two solutions are referred to as parents. One node in each parent is then randomly selected. The tree parts composed of the selected node toward the terminals of each tree are then switched and two new candidate solutions formed (children). When the stopping criteria are met the process comes to an end and the *most fit* individual is outputted as a computer program.

3.2. Framework description

The evolutionary framework (EF) is designed to evolve problem-specific computer programs, formed by mathematical expressions of the image's original channels, to maximize the ability of standard image classification algorithms to separate the target feature from the remaining image background. In this approach, EF combines genetic programming, standard image processing algorithms, and accuracy assessment algorithms with standard image classification algorithms to form a hybrid methodology that searches for the optimal computer program in a learn-from-examples schema (Momm, Easson, & Wilkins, 2006). Fig. 2 is a schematic describing the key components of the evolutionary framework.

In this system, the user provides three datasets: a multi-channel image, reference data indicating image locations where the feature of interest is present and image locations where the feature of interest is not present, and the parameters controlling the evolutionary process. The parameters include the number of image channels, number of iterations (generations), population size, percentage of crossover, percentage of mutation, restarting threshold, and stopping criteria.

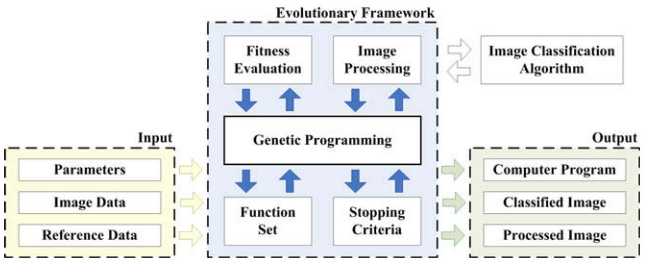

Fig. 2. Conceptual diagram illustrating the main components of the evolutionary framework and its integration to standard image classification algorithms.

In the first iteration (generation), the genetic programming algorithm randomly generates a set *P* of candidate computer programs, referred to as candidate solutions for the problem being investigated. This set of candidate solutions are individually applied to the user-provided multi-channel image, producing a new set of single channel images, called processed images. This new set of processed images is then passed to the image classification algorithm to individually classify them into binary classified images containing only two classes, one representing the class where the target feature is found and another representing the class where the target feature is not found.

The classification accuracy assessment algorithm computes the fitness values of each candidate solution (computer program) by comparing the binary classified image generated by the candidate solution to the user-provided reference data. Classification accuracy measurements are derived from the contingency table (Lillesand & Kiefer, 2000) in the form of overall accuracy or Kappa coefficient of agreement (Cohen, 1960).

All candidate solutions are then sorted by fitness values and then the algorithm checks the stopping criteria represented by the maximum number of iteration and the fitness threshold. If any of these conditions are achieved, the EF outputs the "most fit" candidate solution of the last iteration in the form of a computer program, a classified binary image, a processed image generated by the computer program and the accuracy value yielded.

Conversely, if none of the stopping criteria are met, the iterative process continues following the principles controlling the biological evolution theory where only the *fittest* candidate solutions are carried forward to form the next generation. Only the candidate solution with the highest fitness values (in this work the upper 30%) are carried forward to the next generation after genetic operations such as replication, crossover, and mutation are performed. The newly evolved set of candidate solutions are then applied to the multi-channel image and the entire process is repeated until one of the stopping criteria is reached.

3.3. Implementation

The entire system was implemented in C and C++ computer programming languages as a result of the necessity to have new and highly customizable tools combined with high computational performance offered by both programming languages. The evolutionary framework was implemented as distinct modules (dynamic libraries) and linked together during run time. This modular architecture allows for modifications and/or additions of individual modules without changing the entire system. The following are examples of evolutionary framework modules: genetic programming, image texture, image processing, image statistics, fitness function, function set, fitness computation, input preparation, and others. Furthermore, the selected architecture facilitates the integration of the evolutionary framework with different external image processing algorithms, such as the image classification module.

4. Methods

4.1. Texture operators

Twelve image texture operators were selected and divided into three groups: first order image texture statistics, secondary image textural operators, and Haralick texture operators (Haralick, Shanmugam, & Dinstein, 1973). All of the operators chosen are based on the concept of convolution function (often referred to as neighboring operations). Convolution functions are a filtering process in which mathematical local operations are performed on a pre-defined size and shape window that moves across the image. An operation is performed using the pixels within the window as arguments for a function and the resulting value is put in the output image which usually has the same number of pixels as the original image. The center of the window in the original image and the output pixel in the output image has the same coordinates (Schowengerdt, 1997). Convolution operations are performed on individual image channels. Fig. 3 depicts the convolution concept.

4.1.1. First order image texture statistics

In this group are the most basic convolution functions, formed by descriptive statistics such as: *average*, *variance*, *standard deviation*, and *entropy*. These descriptive statistics are by definition invariant on any permutation of the pixels, and therefore they are rotation and scale independent (Jahne, 2002). The following equations were used to calculate average, standard deviation, and entropy images.

$$\text{Average}\quad(\text{AVE}) = \left(\frac{1}{W}\right)\sum_{i=0}^{q_k}(i * f_i) \tag{1}$$

$$\text{Standard Deviation}\quad(\text{STD}) = \sqrt{\left(\frac{i}{W}\right)\sum_{i=0}^{q_k}(i - AVE)^2 * f_i} \tag{2}$$

$$\text{Entropy}\ (\text{ENT}) = \sum_{i=0}^{q_k}\left(\frac{f_i}{W}\right)\ln\left(\frac{f_i}{W}\right) \tag{3}$$

In these equations, the term f_i represents the frequency of a pixel value i within the window considered. The term q_k represents the quantification of the image channel k (radiometric resolution). The total number of pixels in the window considered is represented by W.

4.1.2. Secondary image textural operators

The second set of textural operators, referred to as secondary image textural operators, is formed by a set of texture filters. Four texture operators are included in this set: minimum (MIN), maximum (MAX), minimum–maximum difference (MMD), and texture unit number (TUN). The MIN and MAX functions output the minimum and maximum, respectively, of the pixels within the considered window. The MMD textural operator returns the difference between the MAX and the MIN values. The TUN is based on the idea of quantifying the ordering of the pixel values in a clockwise way (Jensen, 1996). In a 3 × 3 window size, for example, let the pixel values be represented as $V = \{V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8\}$ where V_0 is the window's central pixel. Another set, referred to as the temporary texture unit, contains $\text{TTU} = \{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8\}$ where each element is computed as:

$$E_i = \left\{\begin{array}{l} 0, \text{if } V_i < V_0 \\ 1, \text{if } V_i = V_0 \\ 2, \text{if } V_i > V_0 \end{array}\right\} \quad \text{for } i = 1\text{–}8 \tag{4}$$

Because each element can have only one of the possible three values the TUN can be computed as following:

$$\text{TUN} = \sum_{i=1}^{W-1} 3^{i-1}E_i \tag{5}$$

4.1.3. Haralick textural operators

The third group of texture operators was originally proposed by Haralick et al. (1973). This group of texture operators is based on the concept of the gray level co-occurrence matrix (GLCM) which takes into consideration the relationship between groups of two neighboring pixels within the window being considered. The GLCM is computed by counting the number of distinct pair of pixel values according to a pre-defined pixel *separation* and *orientation*. In this study the GLCM matrix was added to its transpose to obtain a symmetric matrix. This operation was performed to reduce the number of possible orientations. Adding the GLCM matrix to its transpose causes the opposite orientation to be also considered (Hall-Beyer, 2007). For example, by adding the GLCM matrix obtained by considering an East orientation (0°) to its transpose results in an East–West matrix. Finally the GLCM matrix was normalized to express its values as probability.

$$\text{Probability}\ (P_{i,j}) = \frac{V_{i,j}}{\sum_{i,j=0}^{q_k} V_{i,j}} \tag{6}$$

The term $V_{i,j}$ represents the gray level co-occurrence matrix where i and j represent the sample and line numbers, respectively. Five textural operators were selected from the 14 originally proposed by Haralick:

$$\text{Contrast}\ (\text{CON}) = \sum_{i,j=0}^{q_k}\left[P_{i,j}(i-j)^2\right] \tag{7}$$

$$\text{Dissimilarity}\quad(\text{DIS}) = \sum_{i,j=0}^{q_k}[P_{i,j}|i-j|] \tag{8}$$

$$\text{Homogeneity}\ (\text{HOM}) = \sum_{i,j=0}^{q_k}\left[\frac{P_{i,j}}{1+(i-j)^2}\right] \tag{9}$$

$$\text{Angular Second Moment}\quad(\text{ASM}) = \sum_{i,j=0}^{q_k}[P_{i,j}^2] \tag{10}$$

$$\text{Entropy}\quad(\text{ENT}) = \sum_{i,j=0}^{q_k}[P_{i,j}[-\ln(P_{i,j})]] \tag{11}$$

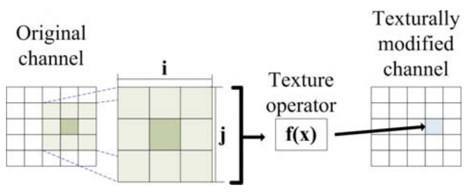

Fig. 3. Illustration of the convolution function (neighboring operations) concept.

4.2. Texture information as image channels

One common difficulty when using evolutionary computation to solve image processing problems is the significant computational overhead (CPU time and large memory requirements) that can be generated (Daida et al., 1996). A typical run of evolutionary computation, addressing remote sensing problems, can easily take minutes to several hours to complete. This is due to the fact that each individual candidate solution is formed by a sequence of operations (equations) where the attributes consist of images and during the evolutionary process these operations are performed thousands (or even millions) of times during the fitness evaluation depending on the population size and number of iterations. The situation is aggravated when image textural operators are incorporated. Due to the computational load and memory size required to compute texture images, when added to the already complex programs evolved by the evolutionary framework, the incorporation of textural information can increase the computational time by many orders of magnitude (Poli, 1996a, 1996b).

One possible alternative is to preprocess the textural information prior to the evolutionary process and store it as additional channels to the original datasets. In this approach, the textural information is used by the evolutionary algorithm as a set of variables (terminal set) instead of functions (function set) and thus being computed only once for the entire process.

Table 1 lists four distinct images with varying number and types of image textural operators. The first image does not have textural channels and therefore only the image's original spectral channels are considered. The second, third, and fourth images are formed by the image's original spectral channels plus the textural channels resulted from the application of the texture operators from the first order image texture statistics group (Section 4.1.1), secondary image textural operators group (Section 4.1.2), and Haralick's textural operators (Section 4.1.3), respectively. Each variable is identified by two indices: the first index denotes the spectral band in which the texture operator was applied while the second index denotes the window size (3 for 3×3, 5 for 5×5, etc.). Image 4 in Table 1, was generated considering an East–West orientation with a separation of one pixel. Fig. 4 shows examples of selected textural channels for images 2–4.

5. Experimental results

This work is based on the premise that image texture contains important information for the problem being investigated, and thus, the classification of images with additional textural channels should yield improved results than those without them. The following experiment was designed to quantitatively examine the overall performance of standard image classification algorithms in integrating spectral and textural information with and without the aid of the evolutionary framework. The image classification algorithm selected is the K-Means algorithm (Tou & Gonzalez, 1974) due to its simplicity and low computational cost. However, it is important to mention that other image classification algorithms could had been used instead.

The K-Means algorithm was set to classify the provided images into a two class thematic images. One class represents the feature of the interest while the other represents the remaining image background. This constitutes a challenge for the classification algorithm because it creates clusters based on statistical groupings of the spectral/textural information of each channel, which may or may not correspond to the desired feature (Aronoff, 2005). Yet, this scenario was chosen as a test bed to evaluate the ability of the evolutionary framework to select a subset of textural and spectral channels from a larger set and to combine them in a non-linear fashion to improve the user-defined classifier's performance.

The problem selected was designed to assess the classifier's ability to separate features with similar spectral signatures but distinct textural pattern. This was addressed by investigating a problem which the goal was to discern planted evergreen tree fields from natural evergreen fields using high spatial resolution commercial imagery. The spectral similarities between these two fields combined with the limited spectral resolution offered by the selected sensor (four spectral channels: blue, green, red, and infrared) limits the ability of classifiers to rely solely on spectral information. In addition, selection of the appropriate texture oper-

Table 1
Different number and types of image textural channels of the four images considered in the evaluation of the evolutionary framework.

Image 1 (4 channels)				Image 2 (52 channels)				Image 3 (68 channels)				Image 4 (84 channels)			
B_1	B_2	B_3	B_4	B_1	B_2	B_3	B_4	B_1	B_2	B_3	B_4	B_1	B_2	B_3	B_4
				AVE$_{13}$	AVE$_{23}$	AVE$_{33}$	AVE$_{43}$	MIN$_{13}$	MIN$_{23}$	MIN$_{33}$	MIN$_{43}$	ASM$_{13}$	ASM$_{23}$	ASM$_{33}$	ASM$_{43}$
				AVE$_{15}$	AVE$_{25}$	AVE$_{35}$	AVE$_{45}$	MIN$_{15}$	MIN$_{25}$	MIN$_{35}$	MIN$_{45}$	ASM$_{15}$	ASM$_{25}$	ASM$_{35}$	ASM$_{45}$
				AVE$_{17}$	AVE$_{27}$	AVE$_{37}$	AVE$_{47}$	MIN$_{17}$	MIN$_{27}$	MIN$_{37}$	MIN$_{47}$	ASM$_{17}$	ASM$_{27}$	ASM$_{37}$	ASM$_{47}$
				AVE$_{19}$	AVE$_{29}$	AVE$_{39}$	AVE$_{49}$	MIN$_{19}$	MIN$_{29}$	MIN$_{39}$	MIN$_{49}$	ASM$_{19}$	ASM$_{29}$	ASM$_{39}$	ASM$_{49}$
				STD$_{13}$	STD$_{23}$	STD$_{33}$	STD$_{43}$	MAX$_{13}$	MAX$_{23}$	MAX$_{33}$	MAX$_{43}$	ENT$_{13}$	ENT$_{23}$	ENT$_{33}$	ENT$_{43}$
				STD$_{15}$	STD$_{25}$	STD$_{35}$	STD$_{45}$	MAX$_{15}$	MAX$_{25}$	MAX$_{35}$	MAX$_{45}$	ENT$_{15}$	ENT$_{25}$	ENT$_{35}$	ENT$_{45}$
				STD$_{17}$	STD$_{27}$	STD$_{37}$	STD$_{47}$	MAX$_{17}$	MAX$_{27}$	MAX$_{37}$	MAX$_{47}$	ENT$_{17}$	ENT$_{27}$	ENT$_{37}$	ENT$_{47}$
				STD$_{19}$	STD$_{29}$	STD$_{39}$	STD$_{49}$	MAX$_{19}$	MAX$_{29}$	MAX$_{39}$	MAX$_{49}$	ENT$_{19}$	ENT$_{29}$	ENT$_{39}$	ENT$_{49}$
				ENT$_{13}$	ENT$_{23}$	ENT$_{33}$	ENT$_{43}$	MMT$_{13}$	MMT$_{23}$	MMT$_{33}$	MMT$_{43}$	CON$_{13}$	CON$_{23}$	CON$_{33}$	CON$_{43}$
				ENT$_{15}$	ENT$_{25}$	ENT$_{35}$	ENT$_{45}$	MMT$_{15}$	MMT$_{25}$	MMT$_{35}$	MMT$_{45}$	CON$_{15}$	CON$_{25}$	CON$_{35}$	CON$_{45}$
				ENT$_{17}$	ENT$_{27}$	ENT$_{37}$	ENT$_{47}$	MMT$_{17}$	MMT$_{27}$	MMT$_{37}$	MMT$_{47}$	CON$_{17}$	CON$_{27}$	CON$_{37}$	CON$_{47}$
				ENT$_{19}$	ENT$_{29}$	ENT$_{39}$	ENT$_{49}$	MMT$_{19}$	MMT$_{29}$	MMT$_{39}$	MMT$_{49}$	CON$_{19}$	CON$_{29}$	CON$_{39}$	CON$_{49}$
								TUN$_{13}$	TUN$_{23}$	TUN$_{33}$	TUN$_{43}$	DIS$_{13}$	DIS$_{23}$	DIS$_{33}$	DIS$_{43}$
								TUN$_{15}$	TUN$_{25}$	TUN$_{35}$	TUN$_{45}$	DIS$_{15}$	DIS$_{25}$	DIS$_{35}$	DIS$_{45}$
								TUN$_{17}$	TUN$_{27}$	TUN$_{37}$	TUN$_{47}$	DIS$_{17}$	DIS$_{27}$	DIS$_{37}$	DIS$_{47}$
								TUN$_{19}$	TUN$_{29}$	TUN$_{39}$	TUN$_{49}$	DIS$_{19}$	DIS$_{29}$	DIS$_{39}$	DIS$_{49}$
												HOM$_{13}$	HOM$_{23}$	HOM$_{33}$	HOM$_{43}$
												HOM$_{15}$	HOM$_{25}$	HOM$_{35}$	HOM$_{45}$
												HOM$_{17}$	HOM$_{27}$	HOM$_{37}$	HOM$_{47}$
												HOM$_{19}$	HOM$_{29}$	HOM$_{39}$	HOM$_{49}$

XXX$_{ij}$, where XXX represents the textural operator, i represents the image spectral channel, and j the convolution window (in pixels).
B_i represents the individual original spectral channels.

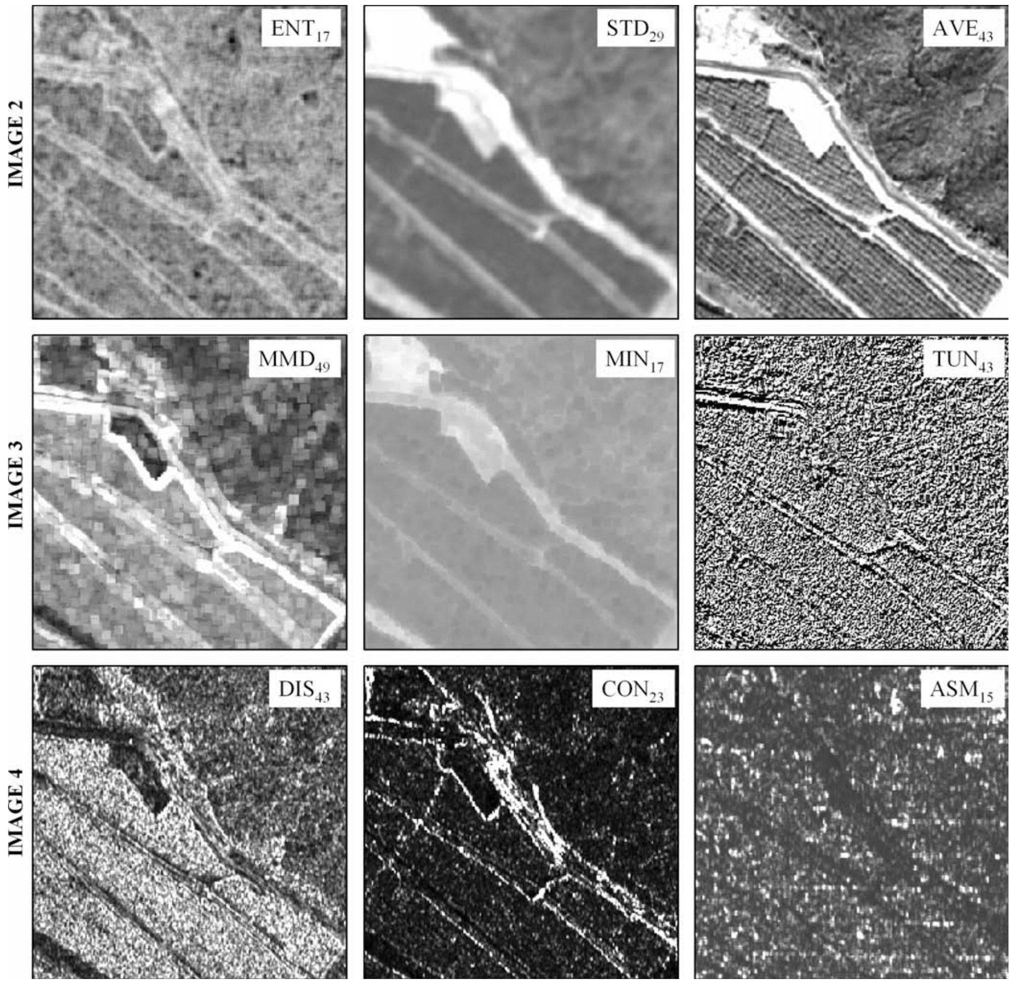

Fig. 4. Examples of additional image channels generated by the application of textural operators to individual original spectral channels of the satellite image. Image are identified by the texture operator (three letter acronym) followed by the indices representing the spectral channel used and the convolution window size. These images were used in the investigation of the evolutionary framework.

ator and its parameters represents an extra level of difficulty due to the large number of possibilities to choose from.

5.1. Data preparation

A scene acquired with the QuickBird sensor on March 2007 covering an area with 1.3 km^2 located in Oxford, Mississippi, USA was selected as the study site (Fig. 5A). The four multi-spectral original channels, blue (485 ηm), green (560 ηm), red (660 ηm), and near infrared (830 ηm), were converted from scaled radiance to absolute radiance using the commercial software package ENVI. Polygons were generated to be used as reference in the evolutionary framework by manually delineating of fields with planted and fields with natural evergreen trees. The fields intentionally included pure and mixed pixels containing tree canopy, grass land, and shadows. This approach was adopted to reinforce the algorithm's aptitude to use textural over spectral information. Fig. 5A illustrates the study site and the fields selected as reference. Four convolution window sizes were considered: 3×3, 5×5, 7×7, and 9×9.

The original multi-spectral image was loaded into the evolutionary framework and the texture module was used to compute the texture channels previously described in Section 5.1. Four images were then created by grouping the different texture channels. Table 1 shows each of the images created with their respective spectral and textural channels.

5.2. Methodology

Table 2 summarizes the parameters controlling the run of the evolutionary framework. These parameters were selected based on recommendations by Koza (1992) combined with the experience gained during the use of the evolutionary framework to address different remote sensing problems. Three independent realizations of the evolutionary framework were performed for each of the images considered and the realization with the highest fitness value was selected. This approach was adopted because the genetic programming algorithm begins with a random seed and converges to the optimal solution. Therefore, it is possible that important parts of the solution could be left out of the initial set of candidate solution. In those cases, the algorithm's convergence ability could be compromised leading to convergence to a local minimum rather than the desired global minimum. Uncertainty analysis of the variability of the system due to the random seed generation (from which the algorithm begins) indicated that the convergence of the algorithm is influenced by the diversity of the population (Momm, Easson, & Kuszmaul, 2008).

Even though no mutation operation was used, the diversity of the population was controlled by a procedure known as "restarting" or introduction of new genetic material. In this process new genetic material is introduced into the evolutionary process after a number of iterations without change in the fitness values of the "most fit" candidate solution. The restarting threshold of five

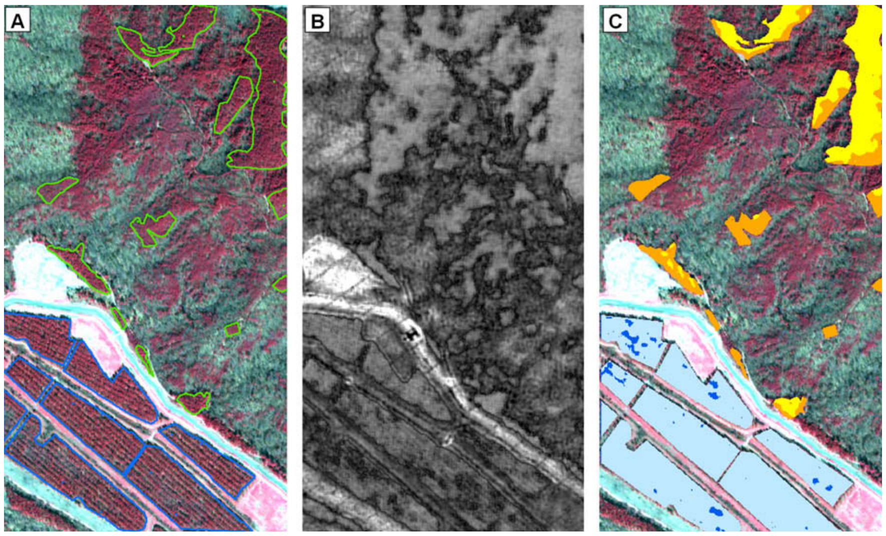

Fig. 5. Satellite image used in the evaluation of the evolutionary framework. (A) shows the scaled reflectance values of QuickBird sensor (spectral band combination 4-3-1). Overlaid are polygons representing manually classified fields with natural (green) and planted (blue) evergreen trees. (B) Example of a preprocessed image produced by the evolutionary framework. (C) Classified image generated with the evolutionary framework overlaid on the satellite image. Color symbology: light blue – true positive, dark blue – false positive, yellow – true negative, and orange – false negative. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Summary of the parameters used to control the run of the evolutionary framework.

Parameters	Description
1. Terminal Set	Image's channels
2. Function Set	SUM, SUB, DIV[1], MUL, SQRT[2], LOG[3], ABS
3. Fitness	Kappa coefficient of agreement
4. Population Size	40 Candidate solutions
5. Generations	91 (initial randomly generated plus one)
6. Crossover	30%
7. Stopping Criteria	91 Generations (iterations) or fitness threshold of 0.975
8. Restarting Threshold	5 Consecutives generations without change in the fitness value

1 – safe division, 2 – safe square root, and 3 – safe logarithm.

Table 3
Comparison of kappa coefficient of agreement values generated by the evolutionary framework using different levels of textural information.

Scenario number	Image number	Classification method	Total number of channels	Number of selected channels	Kappa coefficient of agreement
1	1	K-Means	4	N/A	0.058
2	1	K-Means+EF	4	3	0.427
3	2	K-Means	52	N/A	0.004
4	2	K-Means+EF	52	5	0.609
5	3	K-Means	68	N/A	0.020
6	3	K-Means+EF	68	5	0.600
7	4	K-Means	84	N/A	0.465
8	4	K-Means+EF	84	9	0.403

was selected and the standard deviation values of 0.027, 0.077, 0.090, and 0.027 were obtained for the three runs of the system for the scenarios 2, 4, 6, and 8, respectively. For further description of the restarting process please refer to the work of Momm et al., 2008 and Ines & Honda, 2005.

The output of each realization is composed of a processed image, a classified binary image, solution evolved (mathematical expression), and a kappa coefficient of agreement value (Cohen, 1960). Table 3 shows the accuracy results expressed as kappa coefficient of agreement and Table 4 displays the final solutions generated by the evolutionary framework. The processed image is the result of the application of the evolved candidate solution to the original image. Fig. 5B presents an example of a processed image generated by the evolutionary algorithm. The clustering of the processed image into two classes creates the classified binary image. Fig. 5C shows an example of a classified binary image overlaid on the original imagery. The kappa statistic value is then computed by comparing the final binary image with the user-provided reference data.

5.3. Discussion of results

Comparisons were made for the classification results of four different images obtained by using a standard classification algorithm with and without the aid of the evolutionary framework (Table 3). When the evolutionary framework was not involved, scenarios 1, 3, 5, and 7, the image is processed only with to the classification algorithm. In the scenarios that the classifier was assisted by the evo-

Table 4
Mathematical expressions developed by the evolutionary framework combining spectral and textural information.

Scenario number	Solution
2	$((\mathrm{SQRT}((\mathrm{SQRT}((B_3/B_4)) * ((B_3 + B_4) + (B_3 * (B_3 + B_4))))) - B_2)/\mathrm{ABS}(((\mathrm{SQRT}(((B_3/B_4)/B_4)) * ((\mathrm{SQRT}((B_3/B_4)) * ((B_4 + (B_3 * ((((\mathrm{SQRT}((B_3/B_4)) * ((((B_3 - B_4) + \mathrm{SQRT}((B_3/B_4))) + (B_4 + (B_3 * (\mathrm{SQRT}((B_2/B_4)) * (B_4 + (B_3 * (B_3 + B_3))))))) + (B_3 * ((B_3 - B_4) + \mathrm{SQRT}(((B_3 * \mathrm{SQRT}((B_3/B_4)))/B_4)))))) * B_2) - B_4) + B_4))) + (B_3 * \mathrm{SQRT}(((\mathrm{SQRT}((B_3/B_4)) * ((B_4 + (B_3 * ((((\mathrm{SQRT}((B_3/B_4)) * ((B_4 + (B_4 + (B_3 * (\mathrm{SQRT}((B_2/B_4)) * (B_4 + (B_3 * (B_3 + B_3))))))) + (B_3 * ((B_3 - B_4) + \mathrm{SQRT}((B_3/B_4)))))) * B_2) - B_4) + B_4))) + (B_3 * \mathrm{SQRT}((B_3/B_4))))) * B_2))))) * B_2)) * B_2)))$
4	$(B_4 - (((\mathrm{STD}_{39} * \mathrm{ENT}_{37}) + \mathrm{ABS}(\mathrm{AVE}_{29})) * \mathrm{ABS}(\mathrm{ENT}_{49})))$
6	$((\mathrm{MAX}_{19} + ((B_3 - \mathrm{MAX}_{19}) + ((\mathrm{MAX}_{23} + \mathrm{SQRT}\ (\mathrm{TUN}_{29})) + \mathrm{MMT}_{49}))) * \mathrm{LOG}_{10}\ (B_3))$
8	$(((\mathrm{HOM}_{23} + \mathrm{DIS}_{25}) + (\mathrm{ABS}(((((\mathrm{DIS}_{25}/\mathrm{ENT}_{15}) + ((((B_3/\mathrm{HOM}_{37})/\mathrm{ENT}_{15}) + (B_3 + \mathrm{SQRT}((\mathrm{DIS}_{19} - \mathrm{HOM}_{23})))) + \mathrm{DIS}_{25})) + (\mathrm{ABS}((((\mathrm{HOM}_{37} + \mathrm{ENT}_{15}) + (B_3/(\mathrm{HOM}_{37} + \mathrm{DIS}_{25}))) + \mathrm{SQRT}(\mathrm{HOM}_{23})))/\mathrm{HOM}_{23})) + (\mathrm{DIS}_{25}/\mathrm{HOM}_{23}))) + ((B_4 * ((B_3/\mathrm{HOM}_{37}) - \mathrm{HOM}_{23})) - \mathrm{ENT}_{15}))) + (((\mathrm{HOM}_{37} + \mathrm{ENT}_{15}) + (B_3/\mathrm{HOM}_{37})) + \mathrm{SQRT}(((B_3/\mathrm{HOM}_{37}) - (\mathrm{ABS}((((\mathrm{HOM}_{37} + \mathrm{ENT}_{15}) + (B_3/\mathrm{HOM}_{37})) + \mathrm{SQRT}(((B_3/\mathrm{HOM}_{37}) - \mathrm{HOM}_{23}))))/\mathrm{HOM}_{23})))))$

lutionary framework, scenarios 2, 4, 6, and 8, the mathematical solutions developed by the evolutionary framework (Table 4) are applied to the original image resulting into a new image (referred to as processed image) which is then passed to the classification algorithm. In both cases, the accuracy of the result of the classification is expressed as kappa statistics.

The kappa statistic (Cohen, 1960) was chosen as the accuracy metric because it accounts for the rate chance of agreement. It varies between −1 and +1. Negatives values indicate agreement poorer than the random chance of agreement whereas positive values indicate better agreements beyond the random chance of agreement. A kappa value of zero indicates agreement merely by chance. Landis and Kock (1977) proposed that kappa values greater than 0.75 indicates excellent agreement beyond the random chance of agreement, values between 0.4 to 0.75 indicates good agreement beyond the random chance of agreement, and values between 0 and 0.4 indicates poor agreement beyond chance.

Comparison of kappa coefficient values for images 1–3 shows higher values for the classification process aided by the evolutionary framework. These images changed from poor agreement to good agreement beyond chance. Images 2 and 3 changed to the upper part of the good agreement range. Conversely, the results from the classification of image 4 did not show improvement in kappa values with the aid of the evolutionary framework.

Image 4 is composed by a group of image textural operators referred to as second order statistics (Haralick et al., 1973). These statistical measurements were computed using the co-occurrence matrix, which considers the relationship of groups of two pixels in the image. In this process, each pair of pixels in the window considered is evaluated individually. The selection of the pair of pixels depends on the offset distance and the angular orientation chosen. Image 4 textural channels were computed using an offset of one (immediate neighbor) and an East–West orientation (0°). The introduction of these texture channels have provided improvements in the results (scenario 7) when compared to the image with spectral channels only (scenario 1). However, the assistance of the evolutionary framework yielded results in the same order of magnitude (scenario 8) than the scenario with no aid from the evolutionary framework (scenario 7). The lack of improvement in the accuracy of scenario 8 could be partially attributed to the choice of offset distance and orientation in the creation of the co-occurrence matrix combined with the large number of channels in image 4 (84). The increasing the number of channels also increases the complexity of the problem by enlarging the search space. It is possible that, with the larger search space (84 channels) the genetic programming algorithm was not able to converge to the optimal solution with the number of iterations considered (90).

Tables 3 and 4 also illustrate the ability of the evolutionary framework to reduce the dimensionality of the data set by selecting the most appropriate textural/spectral channels for the problem being investigated. In the scenarios number 4 and 6, higher accuracy results were obtained with non-linear combination of five channels out of 52 and 68, respectively. Even in the scenario number 8, kappa coefficient values in the same order of magnitude as the scenario number 7 were obtained with reduced number of channels.

6. Conclusions

The experiment focused on assessing the performance of the evolutionary framework to evolve custom solutions by combining spectral and textural information in a non-linear way. This was accomplished by an experiment to separate fields with similar spectral characteristics but distinct textural patterns.

The use of texture information as image channels represents a way to reduce the computational load by many orders of magnitude when compared to the option of using texture operators in the function set which requires the value to be computed during the evolutionary process. However, the second alternative should not be discarded. Complex convolution functions, such as the Haralick's textural operators, have many parameters to be defined. The orientation and the separation between pairs of pixels along with the convolution window size considered could be defined by the optimization algorithm according to the feature being investigated and/or the type of problem being addressed.

Based on the experimental results, further investigation should be performed to assess the impact of the dimensionality, represented by the total number of channels, on the algorithm's convergence ability. Images with large number of channels represent a more difficult task for the genetic programming algorithm because the probability of selection of each channel is considerably smaller. For example, for image 1 (four channels) the probability that each channel will be selected during the evolutionary process is 25% (1/4) where for image 4 (84 channels) is 1.1% (1/84). Higher dimensionality increases the chances that important channels necessary for the final solution may not be included in the initial pool of candidate solutions and/or the subsequent pool of candidate solutions used in the restarting process.

The research conducted has demonstrated that image texture contains significant information, when coupled with spectral information, improves the overall separability between the target feature and the remaining image background. The results demonstrated the ability of the evolutionary framework to combine spectral and textural information in a non-linear and complete way to form custom-tailored solutions to address specific problems. The mathematical functions developed by the evolutionary framework are tailored to the problem being investigated. Compared to alternative existing machine learning algorithms, such as ANN and genetic algorithms, these mathematical functions are easily interpreted by humans (Fig. 1). When applied to the original image these functions generate a single-band processed image designed to maximize the influence of the desired feature while minimizing the influence of the remaining image background.

Additionally, traditional trial-and-error processes used to select the type of textural operators and their parameters (window size, orientation, and separation) combined with the number of spectral channels found in multi-spectral images, can limit the use of textural information in supporting remote sensing applications. This iterative process can become labor intensive and time consuming. The use of optimization algorithms such as evolutionary computation provides an alternative capable of significantly reducing the time necessary to develop custom solutions incorporating texture. This is particularly true when the relation between the feature investigated and the image textural information is not fully understood.

Acknowledgements

The authors would like to acknowledge the University of Mississippi Geoinformatics Center for providing the funds, the QuickBird scene, and the image processing software necessary for this research. Thanks are also due to Mississippi Center for Supercomputing Research for allowing us to use their facilities to run our algorithms.

References

Agnelli, D., Bollini, A., & Lombardi, L. (2002). Image classification: An evolutionary approach. *Pattern Recognition Letters, 23*(1–3), 303–309.

Aronoff, S. (2005). *Remote sensing for GIS managers*. Redlands, California, USA: ESRI Press.

Bandyopadhyay, S., & Maulik, U. (2002). Genetic clustering for automatic evolution of clusters and application to image classification. *Parttern Recognition, 35*, 1191–1208.

Baumgartner, A. C., Steger, C., Mayer, H., Eckstein, W., & Ebner, H. (1999). Automatic road extraction based on multi-scale, grouping, and context. *Photogrammetric Engineering & Remote Sensing, 65*(7), 777–785.

Bhandarkar, S. M., Zhang, Y. Q., & Potter, W. D. (1994). An edge detection technique using genetic algorithm based optimization. *Pattern Recognition, 27*(9), 1159–1180.

Brumby, S. P., Theiler, J., Perkins, S., Harvey, N., Szymanski, J., & Bloch, J. J., et al. (1999). Investigation of image feature extraction by a genetic algorithm. In Applications and science of neural networks, fuzzy systems, and evolutionary computation – SPIE 3812.

Chang, C. H., Liu, C. C., & Wen, C. G. (2007). Integrating semi analytical and genetic algorithms to retrieve the constituents of water bodies from remote sensing of ocean color. *Optical Society of America, 15*(2), 252–265.

Chen, L. (2003). A study of applying genetic programming to reservoir trophic state evaluation using remote sensor data. *International Journal of Remote Sensing, 24*(11), 2265–2275.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37–46.

Daida, J. M., Bersano-Begey, T. F., Ross, S. J., & Vesecky, J. F. (1996). Evolving feature-extraction algorithms: Adapting genetic programming for image analysis in geoscience and remote sensing. In *Proceedings of the 1996 international geoscience and remote sensing symposium, part 4 (of 4), May 28–31, 1996* (pp. 2077–2079). Lincoln, Nebraska, USA: Institute of Electrical and Electronics Engineers Computer Society.

Daida, J. M., Hommes, J. D., Ross, S. J., & Vesecky, J. F. (1995). Extracting curvilinear features from synthetic aperture radar images of arctic ice: algorithm discovery using the genetic programming paradigm. In Proceedings of the 1995 international geoscience and remote sensing symposium, part 1 (of 3), July 10–14 1995 (pp. 673–675).

De Jong, K. A. (2006). *Evolutionary computation – A unified approach*. Massachusetts, United States of America: Massachusetts Institute of Technology.

Drunpob, A., Chang, N. B., Beaman, M., Wyatt, C., & Slater, C. (2005). Seasonal soil moisture variation analysis using RADARSAT-1 satellite image in a semi-arid coastal watershed. In 3rd international workshop on the analysis of multi-temporal remote sensing images (pp. 86–190). Institute of Electrical and Electronics Engineers Computer Society, Biloxi, Mississippi, USA.

Fang, H., Liang, S., & Kuusk, A. (2003). Retrieving leaf area index using a genetic algorithm with a canopy radiative transfer model. *Remote Sensing of Environment, 85*, 257–270.

Fogel, D. B. (2000). *Evolutionary computation: toward a new philosophy of machine learning*. New York, USA: IEEE Press.

Fonlupt, C. W. B., & Robilliard, D. (2000). Genetic programming with dynamic fitness for a remote sensing application. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, & Juan Julian Merelo, et al. (Eds.), *Parallel problem solving from nature – PPSN 6th international conference*. Paris, France: Springer Verlag.

Hall-Beyer, M. (2007). GLCM texture: A tutorial, national council on geographic information and analysis remote sensing core curriculum. Version: 2.10 February 2007. <http://www.fp.ucalgary.ca/mhallbey/tutorial.htm>. Accessed August 2007.

Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. *EEE Transactions on Systems, Man and Cybernetics, 3*, 610–620.

Harris, C., & Buxton, B. (1996). *Evolving Edge Detectors with Genetic Programming. Genetic Programming: First Annual Conference*. California, USA: MIT Press, Stanford University. pp. 309–315.

Harris, C., & Buxton, B. (1997). Low-level edge detection using genetic programming: Performance, specificity and application to real-world signals. Gower Street, London, WC1E 6BT, UK.

Hoffman, A. (1989). *Arguments on evolution: A paleontologist's perspective*. New York: Oxford University Press.

Ines, A. V., & Honda, K. (2005). On quantifying agricultural and water management practices from low spatial resolution RS data using genetic algorithms: A numerical study for mixed-pixel environment. *Advances in Water Resources, 28*(8), 856–870.

Jahne, B. (2002). *Digital image processing*. Berlin, Germany: Springer-Verlag.

Jensen, J. R. (1996). *Introductory digital image processing: A remote sensing perspective*. Upper Saddle River, New Jersey, USA: Prentice Hall.

Korczak, J., & Quirin, A. (2003). Evolutionary approach to discovery of classification rules from remote sensing images. In *Lecture notes in computer science*. Berlin/UK: Springer/Heidelberg. pp. 388–398.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press.

Krawiec, K., & Bhanu, B. (2005). Visual learning by coevolution feature synthesis. *IEEE Transations on Systems Man and Cybernetics Part B, 35*(3), 409–425.

Landis, J. R., & Kock, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics, 33*(1), 159–174.

Lillesand, T. M., & Kiefer, R. W. (2000). *Remote sensing and image interpretation*. New York, USA: John Wiley and Sons Inc.

Lin, Y. Q., & Bhanu, B. (2005). Object detection via feature synthesis using MDL-based genetic programming. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, 35*(3), 538–547.

Mitchell, T. M. (1997). *Machine learning*. Singapore: The McGraw-Hill Company, Inc..

Momm, H. G., Easson, G., & Kuszmaul, J. (2007). Integration of logistic regression and genetic programming to model coastal Louisiana land loss using remote sensing. In Proceedings of the ASPRS 2007 annual conference, May 7–11, Tampa, Florida, USA.

Momm, H. G., Easson, G., & Wilkins, D. (2006). Genetic programming as a preprocessing tool to aid multi-temporal imagery classification. In Proceedings of the ASPRS 2006 annual conference, Reno, Nevada, USA.

Momm, H. G, Easson, G., & Kuszmaul, J. (2008). Uncertainty analysis of an evolutionary algorithm to develop remote sensing spectral indices, in image processing: Algorithms and systems VI. In Jaakko T. Astola, Karen O. Egiazarian, & Edward R. Dougherty (Eds.). *Proceedings of SPIE-IS&T electronic imaging* (Vol. 6812, pp. 68120A). SPIE.

Pal, S. K., Bandyopadhyay, S., & Murthy, C. A. (2001). Genetic classifiers for remotely sensed images: Comparison with standard methods. *International Journal of Remote Sensing, 22*(13), 2445–2569.

Perkins, S., Theiler, J., Brumby, S. P, Harvey, N. R, Porter, R., Szymanski, J. J, et al. (2000). GENIE: A hybrid genetic algorithm for feature classification in multi-spectral images. In *Applications and science of neural networks, fuzzy systems, and evolutionary computation III, July 31-August 1 2000* (pp. 52–62). San Diego, California, USA: Society of Photo-Optical Instrumentation Engineers.

Poli, R. (1996a). Genetic programming for image analysis. *Genetic programming: First annual conference*. Stanford University, California, USA: MIT Press. pp. 363–368.

Poli, R. (1996b). Genetic programming for feature detection and image segmentation. *Lecture Notes in Computer Science, 1143*, 110–125.

Quackenbush, L. J. (2004). A review of techniques for extracting linear features from imagery. *Photogrammetric Engineering & Remote Sensing, 70*(12), 1383–1392.

Russ, J. C. (1999). *The image processing handbook*. USA: CRC Press & IEEE Press.

Schowengerdt, R. A. (1997). *Remote sensing models and methods for image processing*. San Diego, California, USA: Academic Press.

Tou, J. T., & Gonzalez, R. C. (1974). *Pattern recognition principles*. Massachusetts, USA: Addison-Wesley Publishing Company, Inc.