Finding Golf Courses: The Ultra High Tech Approach

Neal R. Harvey, Simon Perkins, Steven P. Brumby, James Theiler, Reid B. Porter, A. Cody Young, Anil K. Varghese, John J. Szymanski and Jeffrey J. Bloch

Space and Remote Sensing Sciences Group,
Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Abstract. The search for a suitable golf course is a very important issue in the travel plans of any modern manager. Modern management is also infamous for its penchant for high-tech gadgetry. Here we combine these two facets of modern management life. We aim to provide the cutting-edge manager with a method of finding golf courses from space!
In this paper, we present GENIE: a hybrid evolutionary algorithm-based system that tackles the general problem of finding features of interest in multi-spectral remotely-sensed images, including, but not limited to, golf courses. Using this system we are able to successfully locate golf courses in 10-channel satellite images of several desirable US locations.

1 Introduction

There exist huge volumes of remotely-sensed multi-spectral data from an ever-increasing number of earth-observing satellites. Exploitation of this data requires the extraction of features of interest. In performing this task, there is a need for suitable analysis tools. Creating and developing individual algorithms for specific feature-detection tasks is important, yet extremely expensive, often requiring a significant investment of time by highly skilled analysts. To this end we have been developing a system for the automatic generation of useful feature-detection algorithms using an evolutionary approach.

The beauty of an evolutionary approach is its flexibility: if we can derive a fitness measure for a particular problem, then it might be possible to solve that problem. Many varied problems have been successfully solved using evolutionary computation, including: optimization of dynamic routing in telecommunications networks [1], optimizing image processing filter parameters for archive film restoration [2], designing protein sequences with desired structures [3] and many others.

When taking an evolutionary approach, a critical issue is how one should represent candidate solutions in order that they may be effectively manipulated. We use a genetic programming (GP) method of representation of solutions, due to the fact that each individual will represent a possible image processing algorithm. GP has previously been applied to image-processing problems, including:

edge detection [4], face recognition [5], image segmentation [6], image compression [7] and feature extraction in remote sensing images [8–10]. The work of Daida et al. Brumby et al. and Theiler et al. is of particular relevance since it demonstrates that GP can be employed to successfully evolve algorithms for real tasks in remote-sensing applications.

2 System Overview

We call our feature detection system "GENIE" (GENetic Image Exploitation) [9, 10] GENIE employs a classic evolutionary paradigm: a population of individuals is maintained and each individual is assessed and assigned a fitness value. The fitness of an individual is based on an objective measure of its performance in its environment. After fitness determination, the evolutionary operators of selection, crossover and mutation are applied to the population and the entire process of fitness evaluation, selection, crossover and mutation is iterated until some stopping condition is satisfied.

2.1 Training Data

The environment for each individual in the population consists of a set of training data. This training data consists of a data "cube" of multi-spectral data together with some user-defined data defining "ground-truth". Ground-truth, in this context, is not what is traditionally referred to as ground-truth (this being in-situ data collected at, or as close as possible to, the time the image was taken). Here, ground-truth refers to what might normally be referred to as "analyst-supplied interpretation" or "training data". This training data for our system is provided by a human analyst, using a Java-based tool called ALADDIN. Through ALADDIN, the user can view a multi-spectral image in a variety of ways, and can "mark up" training data by "painting' directly on the image using the mouse. Training data is ternary-valued with the possible values being "true", "false", and "unknown". *True* defines areas where the analyst is confident that the feature of interest **does** exist. *False* defines areas where the analyst is confident that the feature of interest **does not** exist. Fig. 1 shows a screen capture of an example session. Here the analyst has marked out golf courses as of interest.

2.2 Encoding Individuals

Each individual *chromosome* in the population consists of a fixed-length string of *genes*. Each gene in GENIE corresponds to a primitive image processing operation, and so the whole chromosome describes an algorithm consisting of a sequence of primitive image processing steps.

Genes and Chromosomes A single gene consists of an operator name, plus a variable number of input arguments, specifying where input is to come from;

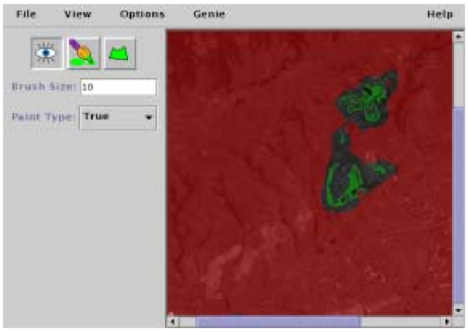


Fig. 1. GUI for Training Data Mark-Up. Note that ALADDIN relies heavily on color, which does not show up well in this image. The light colored patches in the center-right and upper-right parts of the image are two golf courses that have been marked up as "true". Most of the rest of the image has been marked up as "false", except for a small region around the golf courses which has been left as "unknown".

output arguments, specifying where output is to be written to; and operator parameters, modifying how the operator works. Different operators require different numbers of parameters. The operators used in GENIE take one or more distinct image planes as input, and generally produce a single image plane as output. Input can be taken from any data planes in the training data image cube. Output is written to one of a small number of *scratch planes* — temporary workspaces where an image plane can be stored. Genes can also take input from scratch planes, but only if that scratch plane has been written to by another gene positioned earlier in the chromosome sequence.

The image processing algorithm that a given chromosome represents can be thought of as a directed acyclic graph where the non-terminal nodes are primitive image processing operations, and the terminal nodes are individual image planes extracted from the multi-spectral image used as input. The scratch planes are the 'glue' that combines together primitive operations into image processing pipelines. Traditional GP ([11]) uses a variable sized (within limits) tree representation for algorithms. Our representation differs in that it allows for reuse of values computed by sub-trees since many nodes can access the same scratch plane, i.e. the resulting algorithm is a graph rather than a tree. It also differs in that the total number of nodes is fixed (although not all of these may be actually used in the final graph), and crossover is carried out directly on the linear representation.

We have restricted our "gene pool" to a set of *useful* primitive image processing operators. These include spectral, spatial, logical and thresholding operators. Table 1 outlines these operators. For details regarding Laws textural operators, the interested reader is referred to [12, 13].

The set of morphological operators is restricted to function-set processing morphological operators, i.e. gray-scale morphological operators having a flat structuring element. The sizes and shapes of the structuring elements used by

Table 1. Image Processing Operators in the Gene Pool

Code	Operator Description	Code	Operator Description
ADDP	Add Planes	MEAN	Local Mean
SUBP	Subtract Planes	VARIANCE	Local Variance
ADDS	Add Scalar	SKEWNESS	Local Skewness
SUBS	Subtract Scalar	KURTOSIS	Local Kurtosis
MULTP	Multiply Planes	MEDIAN	Local Median
DIVP	Divide Planes	SD	Local Standard Deviation
MULTS	Multiply by Scalar	EROD	Erosion
DIVS	Divide by Scalar	DIL	Dilation
SQR	Square	OPEN	Opening
SQRT	Square Root	CLOS	Closing
LINSCL	Linear Scale	OPCL	Open-Closing
LINCOMB	Linear Combination	CLOP	Close-Opening
SOBEL	Sobel Gradient	OPREC	Open with Reconstruction
PREWITT	Prewitt Gradient	CLREC	Close with Reconstruction
AND	And Planes	HDOME	H-Dome
OR	Or Planes	HBASIN	H-Basin
CL	Clip Low	CH	Clip High
LAWB	Laws Textural Operator $S3^T \times L3$	LAWC	Laws Textural Operator $L3^T \times E3$
LAWD	Laws Textural Operator $E3^T \times E3$	LAWE	Laws Textural Operator $S3^T \times E3$
LAWF	Laws Textural Operator $L3^T \times S3$	LAWG	Laws Textural Operator $E3^T \times S3$
LAWH	Laws Textural Operator $S3^T \times S3$		

these operators is also restricted to a pre-defined set of primitive shapes, which includes, square, circle, diamond, horizontal cross and diagonal cross, and horizontal, diagonal and vertical lines. The shape and size of the structuring element are defined by operator parameters. Other local neighborhood/windowing operators such as mean, median, etc. specify their kernels/windows in a similar way. The spectral operators have been chosen to permit weighted sums, differences and ratios of data and/or scratch planes.

We use a notation for genes that is most easily illustrated by an example: the gene `[ADDP rD0 rS1 wS2]` applies pixel-by-pixel addition to two input planes, read from data plane 0 and from scratch plane 1, and writes its output to scratch plane 2. Any additional required operator parameters are listed after the input and output arguments.

Note that although all chromosomes have the same fixed number of genes, the *effective size* of the resulting algorithm graph may be smaller than this. For instance, an operator may write to a scratch plane that is then overwritten by another gene before anything reads from it. GENIE performs an analysis of chromosome graphs when they are created and only carries out those processing steps that actually affect the final result. Therefore, in some respects, we could refer to the fixed length of the chromosome as a "maximum" length.

2.3 Backends

Complete classification requires that we end up with a single binary-valued output plane from the algorithm. It would be possible to treat, say, the contents of scratch plane 0 after running the chromosome algorithm, as the final output from the algorithm (thresholding would be required to obtain a binary result). However, we have found it to be of great advantage to perform the final classification using a non-evolutionary algorithm.

To do this, we first select a subset of the scratch planes and data planes to be *answer planes*. Typically in our experiments this subset consists of just the scratch planes. We then use the provided training data and the contents of the answer planes to derive the *Fisher Discriminant*, which is the linear combination of the answer planes that maximizes the mean separation in spectral terms between those pixels marked up as "true" and those pixels marked up as "false", normalized by the "total variance" in the projection defined by the linear combination. See [14] for details of how this discriminant works.

The output of the discriminant-finding phase is a gray-scale image. This is then reduced to a binary image by using Brent's method [15] to find the threshold value that minimizes the total number of misclassifications (false positives plus false negatives) on the training data.

2.4 Fitness Evaluation

The fitness of a candidate solution is given by the degree of agreement between the final binary output plane and the training data. This degree of agreement is determined by the Hamming distance between the final binary output of the algorithm and the training data, with only pixels marked as true or false contributing towards the metric. The Hamming distance is then normalized so that a perfect score is 1000. To put this in a more formal/mathematical context. Let H be the Hamming distance between the final binary output of the algorithm and the training data, with only pixels marked as true or false contributing towards the metric, let N be the number of classified pixels in the training image (i.e. pixels marked as either "true" or "false") and let F be the fitness of the candidate solution.

$$F = (1 - (H/N)) \times 1000 \tag{1}$$

2.5 Software Implementation

The genetic algorithm code has been implemented in object-oriented Perl. This provides a convenient environment for the string manipulations required by the evolutionary operations and simple access to the underlying operating system (Linux). Chromosome fitness evaluation is the computationally intensive part of the evolutionary process and for that reason we currently use RSI's IDL language and image processing environment. Within IDL, individual genes correspond to single primitive image operators, which are coded as IDL procedures, with a chromosome representation being coded as an IDL batch executable. In the present implementation, an IDL session is opened at the start of a run and communicates with the Perl code via a two-way unix pipe. This pipe is a low-bandwidth connection. It is only the IDL session that needs to access the input and training data (possibly hundreds of Megabytes), which requires a high-bandwidth connection. The ALADDIN training data mark-up tool was written in Java. Fig. 2 shows the software architecture of the system.

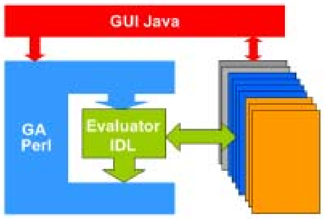


Fig. 2. Software Architecture of the System Described. Note that the feature depicted on the right of this diagram represents the input data, training data and scratch planes

3 Why Golf Courses?

The usefulness of devising algorithms for the detection of golf courses may not, at first, seem apparent (except to a manager, perhaps!). However, due to the nature of golf courses and their characteristics in remotely-sensed data, they are of great use in testing automatic feature-detection systems, such as described here. They possess distinctive spectral and spatial characteristics and it is the ability of feature-detection algorithms to utilize both these "domains" that we seek to test. It is also useful that there exists a great deal of "ground truth" data available: a great many golf courses, for the benefit of low-tech managers, are marked on maps. In addition, golf courses usually possess a well-known, particular type of vegetation and it is rare to find information regarding specific vegetation types on maps. Fig. 3 (a) shows a map of NASA's Moffet Field Air Base, clearly showing the position of a golf course. Fig. 3 (b) shows a false col-

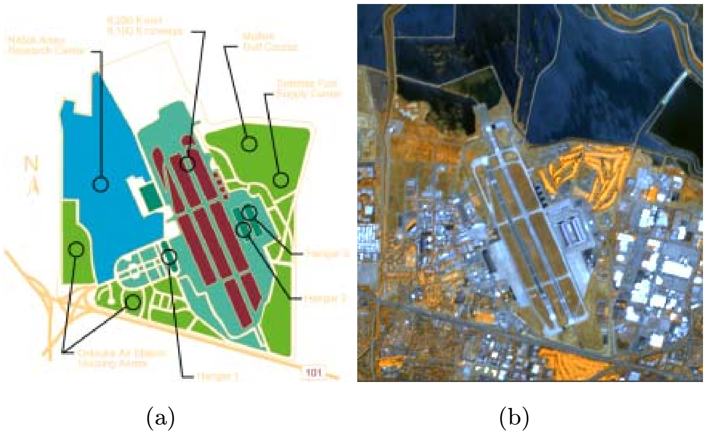

NASA Ames Research Center
9,200 ft and 8,100 ft runways
Moffett Golf Course
Defense Fuel Supply Center
Hangar 2
Hangar 3
Onizuka Air Station Housing Areas
Hangar 1
101
(a)
(b)

Fig. 3. (a) Map of NASA's Moffet Field Air Base, showing a golf course (available at http://george.arc.nasa.gov/jf/mfa/thesite2.html) (b) Image from remotely-sensed data of NASA's Moffet Field Air Base our image of some remotely sensed data of the same region. The airfield and golf course are clearly visible.

4 Remotely-Sensed Data

The remotely-sensed images referred to in this paper are 10-channel simulated MTI data, produced from 224-channel AVIRIS data, each channel having 614×512 pixels. The images displayed are false-color images (which have then been converted to gray-scale in the printing process). The color mappings used are the same for all images shown (an exception being Fig. 1 where the false-color image has had a red and green overlay, corresponding to "false" and "true" pixels, as marked by the human analyst). The particular color mappings used here involve averaging bands A and B for the blue component, bands C and D for the green component and bands E and F for the red component. In addition, the images have been contrast enhanced. The choice of color mappings was arbitrary, in that it was a personal decision made by the analyst, made in order to best "highlight" the feature of interest, from his/her perspective and thus enable him/her to provide the best possible training data. This choice of color-mappings, together with a contrast-enhancement tool, are important and very useful features of ALADDIN. Table 2 provides details about MTI data.

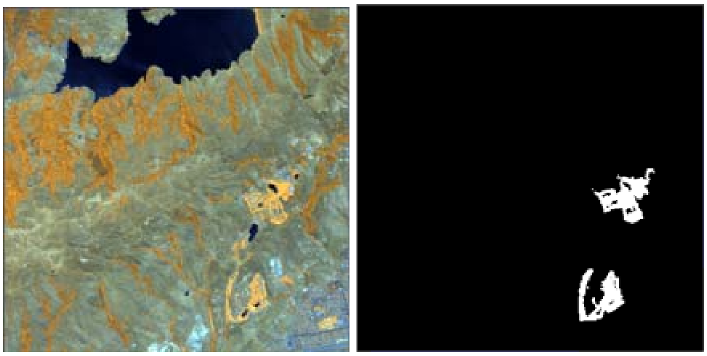
Table 2. MTI Band Characteristics

Band	Wavelength (μm)	Color	SNR	Ground Sample Distance
A	0.45-0.52	blue/green	120	5m
B	0.52-0.60	green/yellow	120	5m
C	0.62-0.68	red	120	5m
D	0.76-0.86	NIR	120	5m
E	0.86-0.89	NIR	500	20m
F	0.91-0.97	NIR	300	20m
G	0.99-1.04	SWIR	600	20m
H	1.36-1.39	SWIR	4	20m
I	1.55-1.75	SWIR	700	20m
O	2.08-2.35	SWIR	600	20m
J	3.50-4.10	MWIR	250	20m
K	4.87-5.07	MWIR	500	20m
L	8.00-8.40	LWIR	800	20m
M	8.40-8.85	LWIR	1000	20m
N	10.2-10.7	LWIR	1200	20m

Figs. 3(a), 4(a) and 5(a) are data taken over an area of NASA's Moffet Field Air Base in California, USA. Fig. 3(a) is a sub-set of the data shown in Fig. 4(a). Figs. 3(a) and 5(a) are non-adjacent regions of the original data. These sub-sets of the data contain a lot of different features, but, of course, have a common feature of interest: golf courses.

5 Searching for Golf Courses

We reserve the data described above (Fig. 3(a)) for testing an evolved golf-course finder algorithm and set the system the task of finding a golf course on some other data. This data, showing the "truth" as marked out by an analyst, is shown in Fig. 1. The golf course area has been marked as "true" and most of the remaining data has been marked as "false". The system was run for 400 generations, with a population of 100 chromosomes, each having a fixed length of 20 genes. At the end of the run the best individual had a fitness of 966 (a perfect score would be 1000). This fitness score actually translates into a detection rate of 0.9326 and a false alarm rate of 0.00018. The results of applying the best overall algorithm found during the run to the data used in the training run are shown in Fig. 4.



(a) (b)

Fig. 4. (a) Image of training data (b) Result of applying algorithm found to training data

It can be seen that the algorithm has been able to successfully detect the golf course and has not detected any of the other features within the image.

In order to test the robustness of the algorithm found, it was applied to out-of-training-sample data, as described previously, and shown in Fig. 3 (b). The results are shown in Fig. 5.

It should be noted that the data shown in Fig. 5 covers a greater area than shown by the map in Fig. 3 (a). It can be seen that the algorithm has successfully found the golf course shown on the map. It can also be seen that the algorithm has detected other golf courses. On closer examination of the data, it would appear that further golf courses do, in fact, exist at those locations. It can also be seen that the algorithm has not found any spurious features.

The "short" (redundant genes stripped out) version of the chromosome found is detailed below.

[LAWG rD2 wS0] [OPREC rD3 wS3 5 1] [ADDP rS0 rS3 wS1] [ADDP rS1 rD6 wS1] [LAWE rD6 wS4] [LAWG rD6 wS0] [OPCL rS4 wS3 1 1] [DIL rS1 wS1 1 0] [OPREC rS1 wS1 5 0] [MEDIAN rS1 wS2 1] [LAWH rD2 wS4]

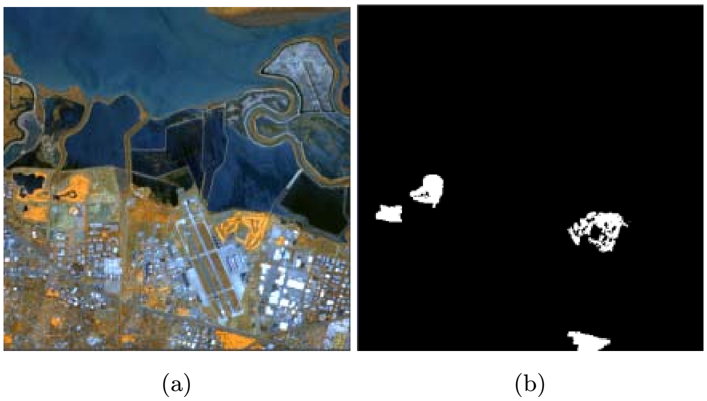


Fig. 5. (a) Image of out-of-training-sample data (b) Result of applying algorithm found to out-of-training-sample data

A graphical representation of the algorithm found is shown in Fig. 6. Note that the circles at the top of the graph indicate the data planes input to the algorithm (in this case only 3 data planes out of a possible 10 have been selected), the 5 circles in the center represent the scratch planes and the circle at the bottom represents the final, binary output of the overall algorithm. The operations above the line of scratch planes represent that part of the overall algorithm incorporated in the chromosome. The operations below the line of scratch planes represent the optimal linear combination of scratch planes and intelligent thresholding parts of the overall algorithm.

It is interesting to have some kind of objective measure of the algorithm's performance on the out-of-training-sample data. To this end an analyst marked up training data (i.e. true and false) for this data, with respect to the golf courses present. This enabled determination of a fitness for the algorithm on this data as well as detection and false alarm rates. The fitness of the algorithm was 926.6, the detection rate was 0.8532 and false-alarm rate was 3.000E-05.

6 Comparison with Other Techniques

In order to compare the feature-extraction technique described here to a more conventional technique, we used the Fisher discriminant, combined with the intelligent thresholding, as described previously, to try and extract the golf courses in the images shown/described. This approach is based purely on spectral information. On application to the data used in the training run (Fig. 4(a)), this "traditional" approach produced a result having a fitness of 757.228 (with respect to the training data/analyst-supplied interpretation), which translates into a detection rate of 0.5159 and a false-alarm rate of 0.00141. On application to the out-of-training-sample data, the result had a fitness of 872.323, which translates into a detection rate of 0.7477 and false-alarm rate of 0.00305. Both of these results are significantly below the performance of the results produced by the Genie system described here.

7 Conclusions

A system for the automatic generation of remote-sensing feature detection algorithms has been described. This system differs from previously described systems in that it combines a hybrid system of evolutionary techniques and more traditional optimization methods. It's effectiveness in searching for useful algorithms has been shown, together with the robustness of the algorithms discovered. It has also been shown to significantly out-perform more traditional, purely-spectral approaches.

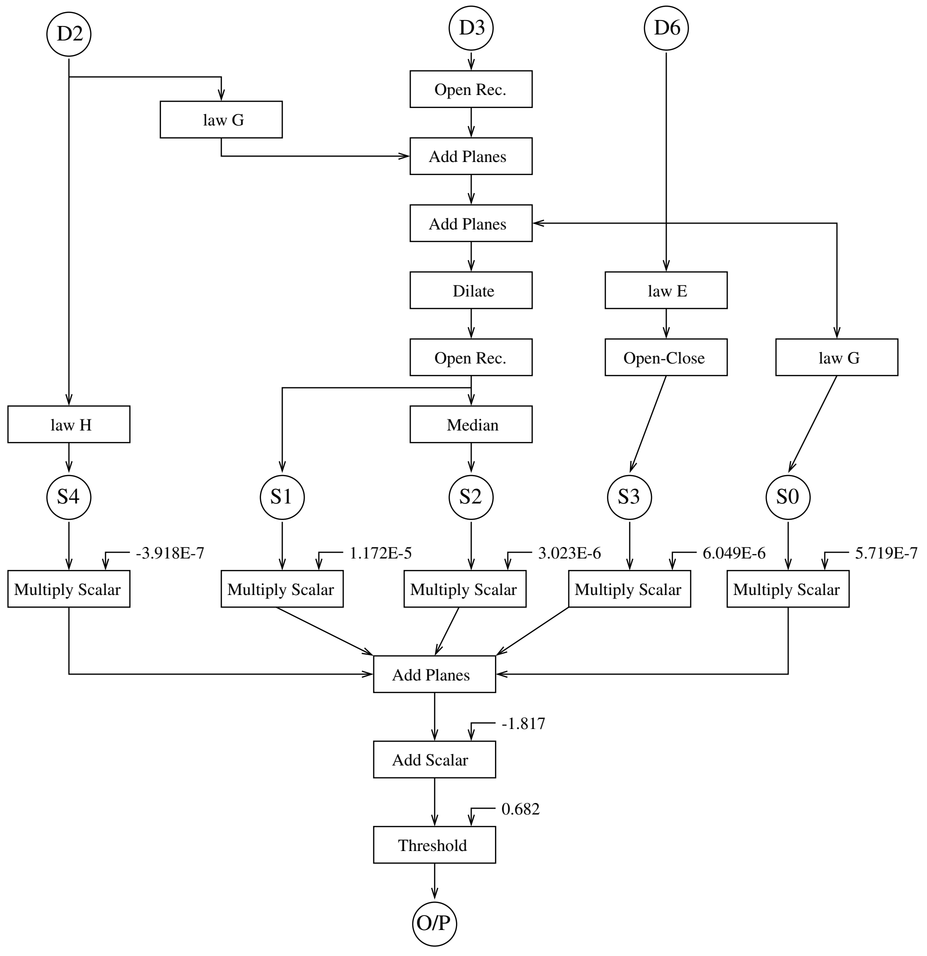


Fig. 6. Graphical representation of algorithm found

References

1. Cox, L.A., Jr., Davis, L., Qiu, Y.: Dynamic anticipatory routing in circuit-switched telecommunications networks, in Handbook of Genetic Algorithms, L. Davis, ed., pp. 124-143, Van Nostrand Reinhold, New York, 1991.
2. Harvey, N.R., Marshall, S.: GA Optimization of Spatio-Temporal Grey-Scale Soft Morphological Filters with Applications in Archive Film Restoration. In: Poli, R., Voigt, H.-M., Cagnoni, S., Corne, D., Smith, G.D., Fogarty, T.C. (eds.): Evolutionary Image Analysis, Signal Processing and Telecommunications (1999) pp. 31–45
3. Dandekar, T., Argos, P.: Potential of genetic algorithms in protein folding and protein engineering simulations, Protein Engineering 5(7), pp. 637-645, 1992.
4. Harris, C., Buxton, B.: Evolving edge detectors, Research Note RN/96/3, University College London, Dept. of Computer Science, London, 1996.
5. Teller, A., Veloso, M.: A controlled experiment: Evolution for learning difficult image classification, in 7th Portuguese Conference on Artificial Intelligence, Volume 990 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1995.
6. Poli, R., Cagoni, S.: Genetic programming with user-driven selection: Experiments on the evolution of algorithms for image enhancement, in Genetic Programming 1997: Proceedings of the 2nd Annual Conference, J. R. Koza, et al., editors, Morgan Kaufmann, San Francisco 1997.
7. Nordin, P., Banzhaf, W.: Programmatic compression of images and sound, in Genetic Programming 1997: Proceedings of the 2nd Annual Conference, J. R. Koza, et al., editors,, Morgan Kaufmann, San Francisco, 1996.
8. Daida, J.M., Hommes, J.D., Bersano-Begey, T.F., Ross, S.J., Vesecky, J.F.: Algorithm discovery using the genetic programming paradigm: Extracting low-contrast curvilinear features from SAR images of arctic ice, in Advances in Genetic Programming 2, P. J. Angeline and K. E. Kinnear, Jr., editors, chap. 21, MIT, Cambridge, 1996.
9. Brumby, S.P., Theiler, J., Perkins, S.J., Harvey, N.R., Szymanski, J.J., Bloch J.J., Mitchell, M.: Investigation of Image Feature Extraction by a Genetic Algorithm in Proc. SPIE 3812, pp. 24–31, 1999.
10. Theiler, J., Harvey, N.R., Brumby, S.P, Szymanski, J.J., Alferink, S., Perkins, S., Porter, R., Bloch, J.J.: Evolving Retrieval Algorithms with a Genetic Programming Scheme in Proc. SPIE 3812, in Press.
11. Koza, J.R.: Genetic programming: On the Programming of Computers by Means of Natural Selection MIT Press, 1992
12. Laws, K.I.: Texture energy measures in Proc. Image Understanding Workshop, Nov. 1979, pp. 47–51.
13. Pietikainen, M., Rosenfeld, A., Davis, L.S.: Experiments with Texture Classification using Averages of Local Pattern Matches IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-13, No. 3, May/June 1983, pp. 421–426.
14. Bishop, C.M.: Neural Networks for Pattern Recognition, pp. 105–112, Oxford University Press, 1995.
15. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C, 2nd Edition, Cambridge University Press, 1992, pp. 402–405..