# MARS TERRAIN IMAGE CLASSIFICATION USING CARTESIAN GENETIC PROGRAMMING

**J. Leitner, S. Harding, A. Förster, J. Schmidhuber**

*Dalle Molle Institute for Artificial Intelligence (IDSIA), SUPSI and Università della Svizzera Italiana (USI), Lugano, Switzerland,* juxi@idsia.ch

## ABSTRACT

Automatically classifying terrain such as rocks, sand and gravel from images is a challenging machine vision problem. In addition to human designed approaches, a great deal of progress has been made using machine learning techniques to perform classification from images. In this work, we demonstrate the first known use of Cartesian Genetic Programming (CGP) to this problem.

Our CGP for Image Processing (CGP-IP) system quickly learns classifiers and detectors for certain terrain types. The learned program outperforms currently used techniques for classification tasks performed on a panorama image collected by the Mars Exploration Rover *Spirit*.

## 1. INTRODUCTION

As more and more sophisticated robots are sent to Mars to explore, the bandwidth required to send data back to Earth becomes a bigger issue. In space all communication is limited by high-latency, low-bandwidth channels to Earth. Therefore nowadays all spacecraft need to utilise autonomy when possible, these include usually tasks like antenna pointing, star tracking, and failure recovery. The first extra-terrestrial surface exploration in the 1970s done by the Soviet rovers exploring the Moon used a 5-man team of "drivers" on Earth to tele-operate the robot [1]. In 1997 the *Sojourner* rover became the first robot to drive on another planet using semiautonomous, purely-reactive terrain navigation [2]. The current Mars Exploration Rovers (MER) were given incremental software updates during their (exceptionally long) stay on Mars, providing also more intelligence on-board. This shows that to increase the scientific return from such missions on-board autonomy is applied more often.

Despite this Artificial Intelligence and Machine Learning are still only in very limited use in space exploration. Recently autonomous science detection has become more interesting and was applied on the MER [3]. This approach was especially useful during dust devil detection, where on-board image analysis was used to detect and photograph relevant science events[4].

For missions such as MER, an important step is to allow navigational autonomy, and for this robust detection of the terrain needs to be performed. Our goal is to build a learning system that can classify various terrains based on visual information. This ability is of importance for such tasks as autonomies exploration, navigation, mobility but also autonomous science operations. We believe that the requirements of systems such as MER, align closely with capabilities of our machine learning based image classification system.

Cartesian Genetic Programming is an evolutionary algorithm that is able to produce compact, fast, robust and human readable computer programs. We show that this approach can accurately classify terrain, and that it can learn robust solutions from a small number of training examples. This has two main benefits: not only does it reduce the effort required by domain experts to provide labelled learning examples, it also reduces the computational effort required to perform the learning. Genetic Programming is amenable to the insertion of domain specific knowledge. For our image processing system, CGP-IP, we exploit the functionality of the OpenCV [5] machine vision library. As the programs generated by CGP-IP are based on OpenCV, the evolved code is familiar to and understandable by humans. Further, CGP-IP produces compact programs that are highly suited for use in embedded systems, where there are considerations for limited processing power and memory.

To demonstrate our system, we classify terrain using data from the *McMurdo* panorama image taken by the Mars Exploration Rover (MER-A) *Spirit* (see Fig. 1). In this work, we demonstrate the first known use of CGP to the terrain classification problem.

## 2. PREVIOUS WORK

In mobile robotics terrain classification is of interest to allow for more autonomous decision making about traversing. Automatically classifying terrain such as rocks, sand and gravel has been researched previously, both in terrestrial and extra-terrestrial robotic applications. It is especially interesting for robotic space exploration,

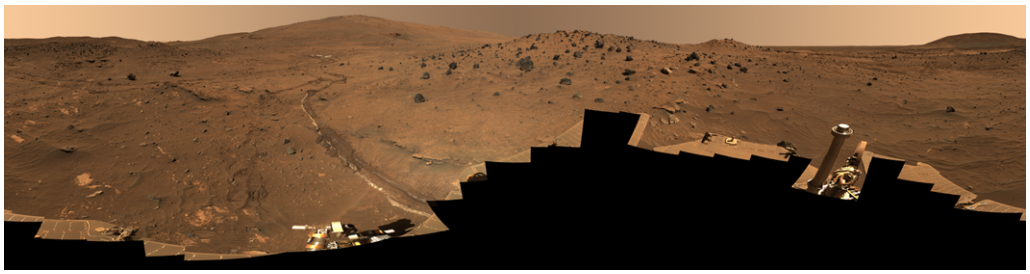

*Figure 1. The* McMurdo *panorama image taken by the Mars Exploration Rover* Spirit *(MER-A) .*

where all communication is limited by high-latency, low-bandwidth channels to Earth. As an example, the current Mars Exploration Rovers (MER) are using fused information from various sensors to build up a map of terrain traversability. The vehicles repeat the process of detecting hazards while moving towards a goal [3].

In computer vision this problem has been investigated from various aspects. Helatci et al. [6] have previously published work on classifying terrains for planetary rovers from images and also from vibrations sensed at the vehicle. There aim was to classify three separate types of terrain during a field test: rocks, beach grass and sandy terrain. Both Low level and high level features were used for classification. Using the vision approach alone they achieve an average accuracy of 77%, which was increased to 84% by fusing the vibration data. Their classification is reported to take approximately 29 seconds per $512 \times 512$ pixel frame on a Pentium 1.8 Ghz desktop PC.

Shang and Barnes [7, 8] investigated terrain classification from visual information only using fuzzy-rough feature selection together with support-vector-machines. The 360-degree view provided by the MER Spirit, the *McMurdo* panorama image. It is a composed image presented in approximately true colour [1], consisting of 1449 separate images and representing a raw data volume of nearly 500 megabytes. It shows the surroundings of the rover and includes dark, porous-textured volcanic, as well as, brighter and smoother-looking rocks. These are embedded in a sandy terrain with ripples and gravel (mixture of small stones). In their paper several classes of terrain were defined and hand labeled, then a low-level feature extraction was applied to generate 36 features per pixel. An SVM is then used to classify the pixel based on those features. With their technique a classification of up to 92% was achieved on hand-selected parts of the panorama. No runtime information is given for generating those 36 local features and classifying the terrain.

Distinguishing these topologies from visual information only is a challenging machine vision problem. In addition to human designed approaches, a great deal of progress has been made using machine learning techniques to perform classification from images.

[1] Publicly available at `http://marswatch.astro.cornell.edu/pancam_instrument/mcmurdo_v2.html`

## 3. SEGMENTING OBJECTS IN IMAGES

Image segmentation is the process of separating foreground from background, or one object class from the background. Most of the previous approaches to this problem have been using features. Features are a brief yet comprehensive representation of the image or scene that possess the property of being interesting and repeatable. Local image features (also called local descriptors) are the entities that capture the information of region around identified points called interest points. Local image features have proven to be resistant to occlusions, local deformations of the objects, variation in illumination conditions, and background clutter [9].

The images are then classified based on the features found. Usually by clustering and matching algorithms. Instead of a feature-based approach we use machine learning in a way that does not need to select certain points of interest but operates on the full input image.

## 4. CARTESIAN GENETIC PROGRAMMING

Machine learning has been used in Computer Vision previously but has mainly focussed on techniques such as Support Vector Machines (SVM), k-Nearest Neighbour (kNN) and Artificial Neural Networks (ANN). Herein we use Genetic Programming (GP) which is a search technique inspired by concepts from Darwinian evolution [10]. It can be used in many domains, but is most commonly used for symbolic regression and classification tasks. GP has also been used to solve problems in image processing, however previous attempts typically use a small set of mathematical functions to evolve kernels, or a small number of basic image processing functions (such as erode and dilate). Previously Spina used GP to evolve programs performing segmentation based on features calculated from partially labelled fore- and background [11].

Given the maturity of the field of image processing, it should be possible to construct programs that use much more complicated image operations and hence incorporate domain knowledge. Shirakawa evolved segmentation programs that use many high-level operations such as mean, maximum, Sobel, Laplacian and product [12].

Herein we are using Cartesian Genetic Programming (CGP), in which programs are encoded in partially connected feed forward graphs [13, 14]. The genotype, given by a list of nodes, encodes the graph. For each node in the genome there is a vertex, represented by a function, and a description of the nodes from where the incoming edges are attached.

The basic algorithm works as follows: Initially, a population of candidate solutions is composed of randomly generated individuals. Each of these individuals, represented by its genotype, is tested to see how well it performs the given task. This step, known as evaluating the 'fitness function', is used to assign a numeric score to each individual in the population. Generally, the lower this error, the better the individual is at performing the task.

In the next step of the algorithm, a new population of individuals is generated from the old population. This is done by taking pairs of the best performing individuals and performing functions analogous to recombination and mutation. These new individuals are then tested using the fitness function. The process of test and generate is repeated until a solution is found or until a certain number of individuals have been evaluated.

CGP offers some nice features, for instance, not all of the nodes of a solution representation (the genotype) need to be connected to the output node of the program. As a result there are nodes in the representation that have no effect on the output, a feature known in GP as 'neutrality'. This has been shown to be very useful [15] in the evolutionary process. Also, because the genotype encodes a graph, there can be reuse of nodes, which makes the representation distinct from a classically tree-based GP representation. An example is shown in Figure 2.

## 5. CARTESIAN GENETIC PROGRAMMING FOR IMAGE PROCESSING (CGP-IP)

Our implementation CGP for Image Processing (CGP-IP) draws inspiration from much of the previous work in the field (see e.g. [16]). It uses a mixture of primitive mathematical and high level operations. It's main difference to previous implementation is that it encompasses domain knowledge, i.e. it allows for the automatic generation of computer programs using a large subset of the OpenCV image processing library functionality [5]. With over 60 unique functions, the function set is considerably larger than those typically used with Genetic Programming. This does not appear to hinder evolution, and we speculate that the increased number of functions provides greater flexibility in how the evolved programs can operate.

Executing the genotype is a straightforward process. First, the active nodes are identified to perform the genotype-phenotype mapping. This is done recursively, starting at the output node, and following the connections used to provide the inputs for that node. In CGP-IP the final node in the genotype is used as the output. Next, the phenotype can be executed on an image. The input image (or images) are used as inputs to the program, and then a forward parse of the phenotype is performed.

The efficacy of this approach was shown for several different domains (including basic image processing, medical imaging, terrain classification, object detection in robotics and defect detection in industrial application) by Harding et al. [17].

### 5.1. Fitness Function

Depending on the application, different fitness functions are available in CGP-IP. The thresholded output image of an individual is compared to a target image using the Matthews Correlation Coefficient (MCC) [18, 19], which has previously been observed to be useful for classification problems solved using CGP [20]. The MCC is calculated based on the 'confusion matrix', which is the count of the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), as follows:

$$c = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (1)$$

A coefficient of 0 indicates that the classifier is working no better than chance. A score of 1 is achieved by a perfect classifier, $-1$ indicates that the classifier is perfect, but has inverted the output classes. Therefore, the fitness of an individual is given by:

$$fitness = 1 - |c| \quad (2)$$

with values closer to 0 being more fit.

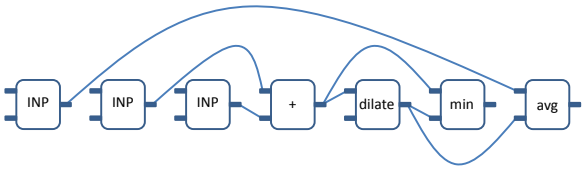


*Figure 2. Example illustration of a CGP-IP genotype. Internally each node is represented by several parameters. In this example, the first three nodes obtain the image components from the current test case (i.e. a grey scale representation and the red and green channels). The fourth node adds the green and red images together. This is then dilated by the fifth node. The sixth node is not referenced by any node connected to the output (i.e. it is neutral), and is therefore ignored. The final node takes the average of the fifth node and the grey scale component from the current test case.*

### 5.2. High Level Operations

Previous work on imaging processing with GP can be divided into two groups. The first group operates on a convolutional approach. Here, a program is evolved that operates as a kernel. For each pixel in an image, the kernel operates on a neighbourhood and outputs a new pixel value. This is also the typical approach when other machine learning approaches are applied to imaging. In GP, the kernel is typically an expression composed from primitive mathematical operators such as $+, -, \times$ and $\div$. For example, this approach was used in [21, 22, 23] to evolve noise reduction filters. In [24], many different image processing operations (e.g. dilate, erode, edge detection) were reverse-engineered. The second group operates on a functional level, where image operations such as dilate and erode are applied to an entire image. This is an obvious method to insert domain knowledge into evolved programs, as now operations that we know are useful can be used without having to re-evolve the same functionality.

CGP-IP combines both these methods. The function set not only contains high-level image processing functions, but also primitive mathematical operations. A complete list can be found in [17].

The primitive operators also work on entire images i.e. addition will produce a new image adding the values of corresponding pixels from two input images. However, this method does not directly allow for kernel-style functionality to be found. Instead, GP has to use a combination of shifts and rotations and other existing kernel methods to get information about a pixel's neighbourhood. This is similar to the methods proposed in [24] to allow for efficient parallel processing of images on Graphics Processing Units (GPUs).

### 5.3. Using Well Known Primitive Operations

Working at the functional level allows for the easy inclusion of many standard image processing operations. In CGP-IP a large number of commands from the OpenCV library are available to GP. Additionally, higher level functions, such as, Gabor filtering are available. Using OpenCV we can also be confident about using high quality, high speed software. In CGP-IP, individuals are evaluated at the rate of 100s per second on a single core. This makes it both efficient to evolve with, but also means that the evolved filters will run quickly if deployed.

Much of the previous work on imaging with GP has focused on the use of grey scale images. Often this is for performance considerations. But also this is out of consideration for how the images will be handled within the program. In CGP-IP, all functions operate on single channel images. The default treatment for colour images is to separate them into both RGB (red, green and blue) and HSV (hue, saturation and value) channels, and provide these as available inputs. A grey scale version of the image is also provided. Each available channel is presented as an input to CGP-IP, and evolutions selects which inputs will be used.

### 5.4. Pruning Generated Programs

Our implementation of CGP-IP generates human readable C# or C++ code based on OpenCV. Although CGP offers bloat free evolution, it often appears to leave redundant nodes in the evolved program. Examples of these include operations that add 0 to every value, or produce a uniformly black image. Whilst the neutrality is important in the genotype, in the phenotype (i.e. the evolved program) it is undesirable.

To optimise the generated code, we implemented a method of pruning that identifies unnecessary operations. The process of pruning a program is as follows: for each active node in the CGP program, replace it with a NOP and re-run the fitness evaluation. If the fitness does not change (or improves), leave the modification in place, and move to the next node. If the fitness degrades, replace the instruction with a node that generates a completely black image, and retest. Again, keep the change if it does not affect the program's output. If it does alter the fitness, a final substitution attempt is made with a white image. If all of these changes degrade performance, the original operation is restored.

It is typically found that this process reduces the number of used instructions, and hence reduces the execution time of the evolved program.

### 5.5. CGP Parameters

As with other CGP implementations, CGP-IP does not require many parameters to be set. The main parameters are:

- Graph length (i.e. the number of nodes in the genotype), which is set to 50 in this case.
- Mutation rate, 10% of all genes in the graph are mutated when an offspring is generated. The threshold parameter is mutated with a probability of 1%.
- Size of mutations
- Number of islands, this depends on the available computing resources. CGP-IP has been tested successfully from 1 to 24 islands.
- Number of individuals per island, which is set to 5 in keeping with the typical 1+4 evolutionary strategy used with CGP.
- Synchronisation interval between islands. Here each island compares their best individual to the servers individual every 10 generations.

*Table 1. Parameters of CGP-IP encoded at every node.*

| Parameter | Type | Range |
|---|---|---|
| Function | Int | # of functions |
| Connection 0 | Int | # of nodes and inputs |
| Connection 1 | Int | # of nodes and inputs |
| Parameter 0 | Real | no limitation |
| Parameter 1 | Int | $[-16, +16]$ |
| Parameter 2 | Int | $[-16, +16]$ |
| Gabor Filter Frequ. | Int | $[0, 16]$ |
| Gabor Filter Orient. | Int | $[-8, +8]$ |

It is important to note that in the work presented here the parameters have not been optimized other than by casual experimentation. It may be possible to improve the performance of CGP-IP by more carefully selecting these parameters. In particular, we would expect the mutation rate, genotype size and number of islands to be the most important parameters to adjust.

CGP-IP needs a number of additional parameters encoded in each node, compared to classical CGP. They are listed in Table 1. These are needed because often the functions used require additional parameters, with specific requirements as to their type and range. Connection 0 and 1 contain the relative address of the node used as first and second input. If a relative address extends beyond the extent of the genome it is connected to one of the inputs. Specialised 'Input' functions are also provided (e.g. INP, SKIP), which manipulate a pointer that indexes the available inputs and return the currently indexed input. A full description can be found in [25]. An illustrative example is shown in Fig. 2. In addition to the graph representing the program, the genotype also includes a value used for thresholding the output. All parameters are kept constant throughout the experiments presented below. Again, whilst it may be possible to improve performance for a given problem by optimising the parameters, we believe that the apparent parameter robustness is an important feature of this technique.

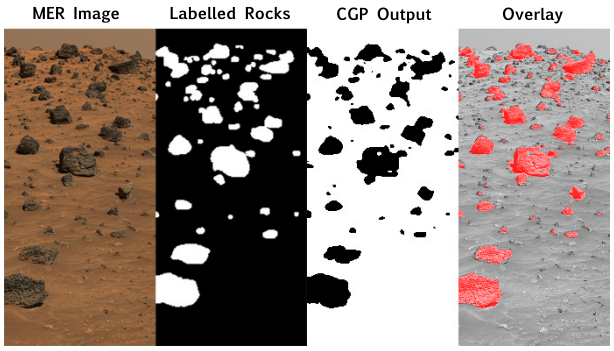


*Figure 3. A brief overview of CGP-IP: From the input (here a MER Image) together with a hand-labelled segmentation, a program is evolved to perform the same segmentation. The output of this program is shown in the third column and is used as an overlay, for quick visual inspection, in the last column.*

### 5.6. Training

A handful of examples, in which the object of interest has been correctly segmented, are used as a training set for CGP-IP. The selection of this training set is of importance for the capabilities of the found solution. If chosen correctly, the resulting programs are very robust filters. In Fig. 3 the various stages are shown. The original image is used to hand-label points of interest (in this case rocks). This is then used as a training set for CGP-IP. The output of the filter is shown in the third column and is used again, just for visualisation, in the last column as an overlay for the input image (in grey). .

An example of the human-readable output, in the form of a C++ computer program, is shown in Listing 1. On a single core of a modern desktop PC, the evolved programs run quickly and as these are largely linear in form, the memory requirements are low. Speed and simplicity of processing is important in many embedded systems, making this approach suitable for implementation in constrained computing environments.

## 6. EXPERIMENTS

### 6.1. Learning to Detect Rocks

This experiment shows how our CGP-IP system can be applied to assist autonomous navigation. We try to detect all rocks (larger than a certain size) to calculate a simple traversability map. The steps to learn a classification are shown in Fig. 3. In one section of the panorama image all rocks are hand-labelled and are used to learn a CGP-IP program for this task.

The evolutionary search finds a useable program after just 8000 evaluations (takes about a minute on a standard desktop PC). The runtime of the program classifying the content of the training image ($347 \times 871$ pixels) is around 400ms. On the scaled down version of the panorama ($8939 \times 2308$) the classification takes about 30 seconds[2]. Fig. 4 shows the result for a different section

[2]All tests were performed and runtimes reported for a 2.1 Ghz Intel Core i5 laptop with 4GB of RAM.

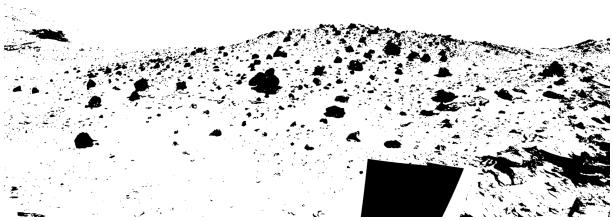

*Figure 4. Rocks detected by an evolved CGP-IP program in a part of the* McMurdo *panorama.*

```
icImage* RockDetector::runFilter() {
    icImage *node0  = InputImages[6]->gauss(3);
    icImage *node1  = node0->sqrt();
    icImage *node9  = InputImages[5];
    icImage *node12 = node9->unsharpen(13);
    icImage *node15 = node1->mulc(7.00936886295676);
    icImage *node24 = node15->SmoothBilateral(9);
    icImage *node31 = node24->Normalize();
    icImage *node33 = node12->mulc(4.03286868333817);
    icImage *node35 = node33->add(node31);
    icImage *node49 = node35->SmoothBilateral(11);

    // evolved threshold value = 177.2417
    icImage *out = node49->threshold(177.2417f);

    // cleanup of memory ...

    return out;
}
```

*Listing 1. The generated code from CGP-IP for detecting a specific type of rock. `icImage` is a wrapper class to allow portability within our framework [26].*

of the panorama. The C++ code is shown to perform this segmentation is shown in Listing 1.

The output can then be used as a simple traversability measure by simply summing up the pixels (which are either 0 or 1) over image parts.

### 6.2. Detecting Specific Rocks

The second experiment aims to help autonomous science detection on space robots. Rocks that are of similar visual appearance as a labelled example are detected, as they, like the hand-labelled rock, might be of scientific interest. In contrast to Fig. 3, where all rocks (larger than a certain size) were to be detected, the aim here is to detect specific rocks of interest.

In a first step the rock (or multiple rocks) of interest is labeled in an image. The section was cropped from the original panorama image and resized to $539 \times 471$ pixels. Figure 5 shows it together with the hand-labelled rock for this experiment. The rock labelled has a different visual appearance than most of the others in this section, similar to porous, volcanic rock. Using this input a learned CGP-IP program then performs its classification. Fig. 6 shows the segmentation performed by CGP-IP in various section of the panorama image. It can be seen that although only this one input was used other rocks that are visually similar were detected.

Speed and simplicity of processing is important in many embedded systems. On a single core of a modern desktop PC, the evolved filters run in about $150ms$ on image sections of $512 \times 512$ pixels, and as the program is largely linear in form, the memory requirements are low. This may make this approach highly suitable for implementing in constrained computing environments.

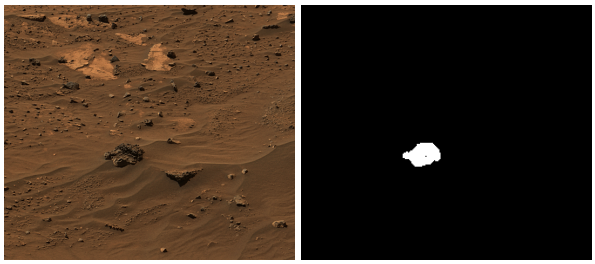

*Figure 5. A part of the panorama showing a rock of interest. The hand-labelled segmentation on the left is used for training.*

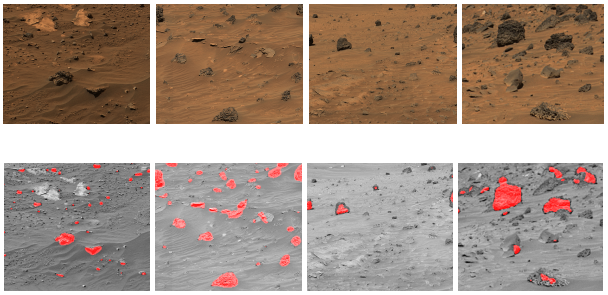

*Figure 6. A selection of scenes from the panorama with the rock segmentation using a single learned program at the bottom.*

### 6.3. Learning to Classify Terrain From Images

To classify the terrain from visual inputs we use the same 5 segmentation classes as used by Shang et al. [7]. Instead of hand-labelling all test images we reuse their results to train our classifiers with. The slices, taken from the full panorama, used in their publication vary in size from $400 \times 400$ to $600 \times 600$ pixels.

Fig. 7 and Fig. 8 show the segmentation performed by CGP-IP in two sections of the panorama. Gravel is shown in yellow, sand in blue. Various types of rocks are detected and shown in green, red and purple. Our approach is quite fast as each class needs about $50 - 150ms$ per slice. To speed up execution those could be performed in parallel. In comparison Helatci et al. [6] require 29 seconds for a $512 \times 512$ image frame.

In [7] 816 labelled feature patterns are needed to perform the classification and no runtime information is given. Herein we use only 7 hand-labelled slices as inputs to learn the segmentation. We believe that an increase in number of hand-labelled data will also allow for higher accuracy. Since there is no dataset with 'ground-truth' available it is therefore hard to objectively compare the accuracy of the various methods. Especially the definition of sand vs. gravel vs. small rock can be very tricky. This is visible in the two figures, where the areas of yellow and blue vary between the two solutions.

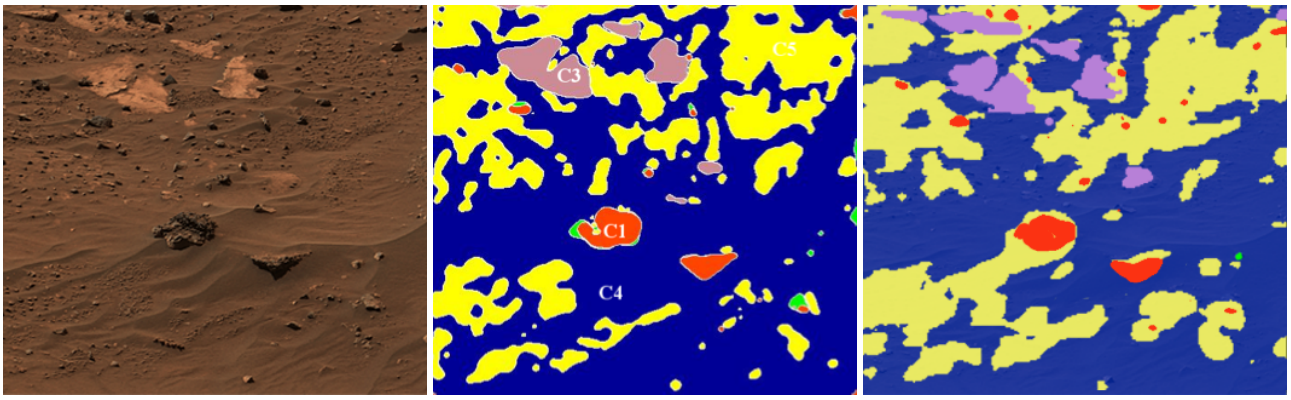


*Figure 7. The right is taken from Shang et al. [7], the left is the classification performed by CGP-IP.*

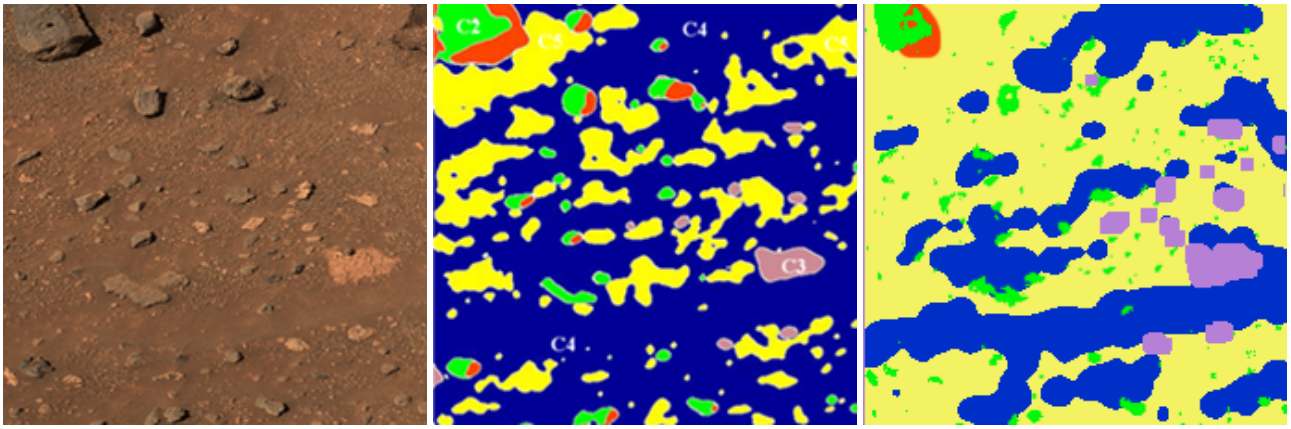


*Figure 8. A second area taken from Shang et al. [7] and compared to our CGP-IP classification.*

## 7. CONCLUSIONS

Automatically classifying terrain such as rocks, sand and gravel from images is a challenging machine vision problem. In addition to human designed approaches, a great deal of progress has been made using machine learning techniques to perform classification from images.

Herein we demonstrate our system, named Cartesian Genetic Programming for Image Processing or CGP-IP. It is the first known use of Cartesian CGP to the terrain classification problem. Herein we apply it to perform image segmentation and classification on Martian images. Data from the *McMurdo* Panorama Image taken by the Mars Exploration Rover (MER-A) *Spirit* (see Fig. 1) was used. Multiple experiments were presented. In the first we evolve a detector for rocks larger than a certain size to support the creation of traversability maps.

In the second our approach was used to learn a detector for rocks of scientific interest from visual input only. The third experiment compared the terrain classification capabilities of CGP-IP to previous approaches.

These results show that our technique allows classification of various types of terrain, such as rocks, sand and gravel. It does so after a short learning phase and creates programs that execute considerably faster then previous approaches.

## REFERENCES


[1] WT Huntress, VI Moroz, and IL Shevalev. Lunar and planetary robotic exploration missions in the 20th century. *Space science reviews*, 107(3):541–649, 2003.

[2] A.H. Mishkin, J.C. Morrison, T.T. Nguyen, H.W. Stone, B.K. Cooper, and B.H. Wilcox. Experiences with operations and autonomy of the mars pathfinder microrover. In *Aerospace Conference, 1998. Proceedings., IEEE*, volume 2, pages 337–351. Ieee, 1998.

[3] M. Bajracharya, M.W. Maimone, and D. Helmick. Autonomy for mars rovers: Past, present, and future. *Computer*, 41(12):44–50, 2008.

[4] A. Castano, A. Fukunaga, J. Biesiadecki, L. Neakrase, P. Whelley, R. Greeley, M. Lemmon, R. Castano, and S. Chien. Automatic detection of dust devils and clouds on mars. *Machine Vision and Applications*, 19(5):467–482, 2008.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] I. Halatci, K. Iagnemma, et al. A study of visual and tactile terrain classification and classifier fusion for planetary exploration rovers. *Robotica*, 26(6):767–779, 2008.

[7] C. Shang, D. Barnes, and Q. Shen. Facilitating efficient mars terrain image classification with fuzzy-rough feature selection. *International Journal of Hybrid Intelligent Systems*, 8(1):3–13, 2011.

[8] C. Shang and D. Barnes. Classification of mars mcmurdo panorama images using machine learning techniques. *Acta Futura*, 5:29–38, 2012.

[9] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *ECCV*, pages 113–130, 2002.

[10] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[11] T. V. Spina, Javier A. Montoya-Zegarra, A. X. Falcao, and P. A. V. Miranda. Fast interactive segmentation of natural images using the image foresting transform. In *16th International Conference on Digital Signal Processing*, pages 1–8, July 2009.

[12] Shinichi Shirakawa and Tomoharu Nagao. Feed forward genetic image network: Toward efficient automatic construction of image processing algorithm. In *Advances in Visual Computing: Proceedings of the 3rd International Symposium on Visual Computing (ISVC 2007) Part II*, volume 4842 of *Lecture Notes in Computer Science*, pages 287–297, Lake Tahoe, Nevada, USA, 2007. Springer.

[13] Julian F. Miller. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1135–1142, Orlando, FL, Jul 1999. Morgan Kaufmann.

[14] Julian F. Miller, editor. *Cartesian Genetic Programming*. Natural Computing Series. Springer, 2011.

[15] J. F. Miller and S. L. Smith. Redundancy and computational efficiency in cartesian genetic programming. In *IEEE Transactions on Evoluationary Computation*, volume 10, pages 167–174, 2006.

[16] Lukas Sekanina, Simon L. Harding, Wolfgang Banzhaf, and Taras Kowaliw. Image processing and CGP. In Julian F. Miller, editor, *Cartesian Genetic Programming*, Natural Computing Series, chapter 6, pages 181–215. Springer, 2011.

[17] S. Harding, J. Leitner, and J. Schmidhuber. Cartesian genetic programming for image processing. In *Genetic Programming Theory and Practice X (in press)*. Springer, 2012.

[18] B W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta*, 405(2):442–451, 1975.

[19] Wikipedia. Matthews correlation coefficient — wikipedia, the free encyclopedia, 2012. [accessed 21-March-2012].

[20] Simon Harding, Vincent Graziano, Jürgen Leitner, and Jürgen Schmidhuber. Mt-cgp: Mixed type cartesian genetic programming. In *Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012.

[21] S. Harding and W. Banzhaf. Genetic programming on GPUs for image processing. *International Journal of High Performance Systems Architecture*, 1(4):231–240, 2008.

[22] Simon L. Harding and Wolfgang Banzhaf. Distributed genetic programming on GPUs using CUDA. In *Workshop on Parallel Architectures and Bioinspired Algorithms*, pages 1–10, Raleigh, NC, 13 September 2009.

[23] Tomás Martínek and Lukás Sekanina. An evolvable image filter: Experimental evaluation of a complete hardware implementation in fpga. In *6th International Conference Evolvable Systems: From Biology to Hardware*, pages 76–85, 2005.

[24] Simon Harding. Evolution of image filters on graphics processor units using cartesian genetic programming. In *IEEE World Congress on Computational Intelligence*, pages 1921–1928, Hong Kong, Jun 2008.

[25] Simon Harding, Wolfgang Banzhaf, and Julian F. Miller. A survey of self modifying cartesian genetic programming. In Rick Riolo, Trent McConaghy, and Ekaterina Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, chapter 6, pages 91–107. Springer, Ann Arbor, USA, 2010.

[26] J. Leitner, S. Harding, M. Frank, A. Förster, and J. Schmidhuber. icVision: A Modular Vision System for Cognitive Robotics Research. In *International Conference on Cognitive Systems (CogSys)*, 2012.