Reliable Rock Detection and Classification for Autonomous Science

Scott Niekum
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
sniekum@andrew.cmu.edu

Abstract -- Current planetary rovers rely heavily on explicit instructions given by scientists on Earth, which requires extensive communication and frequent command cycles. Onboard science autonomy systems will enable rovers to reason about their science goals, so that they may make informed decisions, allocate bandwidth intelligently, and respond to discoveries. A reliable rock detector is the first step in interpreting local geology and enabling onboard science data understanding. The detection of objects in natural scenes is a problem that has been addressed by computer vision and machine learning researchers; it involves a difficult combination of image processing, the use of subtle visual cues, and domain knowledge. An architecture is presented that handles data from multiple sources and combines computer vision and machine learning techniques to achieve segmentation, detection, and classification of rocks. Results using data from the Atacama Desert and associated ground-truth validation confirm a successful approach.

1. Introduction

The detection of objects in natural scenes is a problem that has been addressed both by computer vision and machine learning researchers. It involves a difficult combination of image processing and acquired domain knowledge of the scene. Often, the problem is not trivial for a human looking at an image and there does not seem to be one dominant strategy for going about the task. The problem becomes much more tractable when conditions are fixed, or one type of object in particular is being sought. We are concerned with the problem of rock detection in natural scenes, specifically in desert environments. Although this provides a static environment and a specific class of object, there are many difficulties to be overcome. The types of rocks, terrain, and lighting conditions vary greatly across the desert environment. Also, there is no one quality that defines what a rock is, and it is even less clear how to differentiate rocks from soil, especially if the rock is partially buried. We will address these problems later. There has been documented success with rock detectors under controlled conditions such as white rocks on a black conveyor belt with controlled lighting. However, a generalized rock detector is a much more difficult task.

We will define rock detection as successfully finding pixel-resolution outlines of the rocks in an image. A ‘rock’ will be defined as any rock shown in the image that is larger than 5 by 5 pixels in area, whether it is above ground, partially buried, or occluded. This methodology can be extended to soil patches, layers, and other features of geologic interest. The information gained from this can be tallied and further analyzed to give useful statistics about the rocks and other features in the image. The rocks can also be passed to a classifier to give more information about each rock and rock distributions in the image.

Rock detection has many scientific applications, but this research was specifically motivated by the investigation of life and geologic habitats in the Atacama Desert. The Atacama project employs an autonomous science rover, Zoë. Zoë is a next-generation rover prototype that will overcome limitations in the relatively small amount of information that can be sent back to Earth during a Mars mission. Since current rovers have no on-board science capabilities, many images have to be sent back to scientists on Earth, who manually count rocks, calculate distributions, and determine where to send the rover next. If this scientific summary and decision making could be computed on-board the rover, vast amounts of time could be saved, and bandwidth would be freed up for scientifically relevant discoveries. Reliable rock detection and classification is one capability that such a system would require. A prototype system has been developed and tested in the Atacama Desert, a nearly lifeless Mars-like environment in northern Chile.

Two major functionalities are proposed that will demonstrate an effective on-board science autonomy system. The first is the capability of the rover to obtain science information from raw sensor data. The most relevant use of this functionality is the ability to give scientists compact, useful summaries of observations. This comes from statistics, analysis, and allocating image bandwidth based on scientist preferences and what was actually seen in the field. The second capability is to make intelligent decisions about actions in the field. An ability to make science decisions on the fly creates active responses to what is seen in the field, allowing the rover to opportunistically sample anomalies, or to change the path in order to adapt to local geology and better satisfy scientist goals.

In order to tackle a problem with as large of a scale as rock detection, a systematic methodology was required. Many different computer vision and machine learning algorithms were surveyed over the course of the project, but in the end, one method proved most effective. In the beginning, loose evaluation criteria were established so that the judging of the algorithms could be somewhat systematic and less subjective. Over time these criteria became more mathematical and solid, especially after a full ground truth dataset was collected in the Atacama Desert. These criteria will be discussed in greater detail later. Early in the research, it became clear that no single image processing algorithm would be able to capture something as abstract as the concept of a rock. A hybrid computer vision and machine learning approach has been taken to help in this manner. The final algorithm contains segmentation, detection, and classification stages, each of which relies on multiple algorithms and features to capture the full richness of the problem.

To test the rock detector and evaluate performance, a ground truth data set was necessary. We traveled to the Atacama Desert and collected various types of data sets, such as periodic panoramas and image sequences over multiple traverses. Later, we ground-truthed these data sets to obtain accurate positions of the rocks in the image using software that was carefully developed to remove the bias of the coder. Upon evaluation, we found that the detector showed both high precision and recall, especially for large rocks. Room for improvement exists with respect to boundary detection, small rocks, and shadows. Classification and a science preference interface are also in their early stages.

The background of both computer vision and machine learning work in the field is reviewed. A wide survey of algorithms is also discussed, outlining the many approaches that were considered and often experimented with. This is followed by an in-depth description of the final 3-stage hybrid architecture that was implemented. Successful results from our experiments in the Atacama are shown, followed by a discussion of suggested improvements and further work.

2. Background

We all know a rock when we see it, but the idea of rock detection is actually an abstract and vaguely defined goal. Let us define a rock as a hard, solid mass of matter, of composition different from soil, sand, or other background material. Once a boundary is detected for a rock, we define a bounding box around it, as to accommodate the entire region. A rock will be considered "detected" if the center of the bounding box intersects a rock in the image. The term "well localized" will refer to a region whose area within the bounding box we defined overlaps the true area of the rock in the image by at least 50%.

Isolating a single approach for creating a general rock detector is a daunting task. Mkwelo, De Jager, and Nicolls have done work in uncontrolled lighting conditions, but always looking straight down onto a conveyor belt with fairly uniform rocks and dirt [10]. Another attempt by Farfan, Salinas, and Cifuentes involved examining the fragmentation of rocks after blasting in mining situations. Though many different types of rocks were involved, lighting conditions, angle, and many other environmental factors were constant [7]. These projects used the Watershed algorithm for segmentation purposes, which although powerful often requires the use of markers or other *a priori* information to work correctly. Other projects have made attempts at a general rock detector, but have all fallen short of a general solution. The Nomad project in Antarctica used color information to separate meteorites from background ice, but this is more easily separable information than could be acquired in a desert scene [17]. Castaño et al used intensity data with an edge detector to find rock boundaries [3]. Unfortunately, due to shadows and other lighting phenomenon, intensity is not a very reliable source of information in our environment of interest. The OASIS project has used stereo data, since any deviation from the ground plane is usually a rock [8]. Unfortunately, stereo often fails at long distance, and irregular terrain can make segmentation of such data difficult. Some research has combined multiple of the above techniques, but all such methods share the common assumption that there is a single or fixed set of features that can uniquely express what is and is not a rock. The approach taken in this paper is one combining machine learning and computer vision techniques such that example based analysis is used on image regions found to be homogenous in any given feature space. A 3-stage hybrid system is used to segment these regions and detect and classify rocks using a flexible feature set.

From the perspective of a human, we take many factors into consideration when we seemingly automatically identify rocks in an image. Though the problem seems intuitively easy, it is because there is much going on behind the scenes that we do not see. Visual cues, both instinctual and learned, are used heavily in any object recognition problem. Humans are incredibly skilled at integrating cues such as shading, occlusion, texture, size in the visual field, perceived distance, motion, and many others. The hardware on the average human is also excellent – high resolution color stereo cameras with built-in preprocessing. Then, this low-level information somehow gets combined with high-level domain knowledge about what rocks generally look like (a rough sense of a "template"), what dirt and soil look like, and a sense of what seems realistic or physically possible. Much of this information appears to be processed in parallel in addition to the serial sections of the process. And even with all this knowledge, humans can still be tricked fairly easily. When given candidate rock regions without context information (a surrounding area of the image), most people have great difficulty saying whether the region is a rock or not. On top of all the other difficulties of the process, detection of rocks is extremely subjective. It was found that even geologists do not always agree on what is and is not a rock in an image.

Science Autonomy is of current importance because of the accomplishment of planetary science rovers and the opportunity to overcome current limitations. Castaño et al performed a high-level investigation of the machine learning problems facing autonomous science rovers, including the translation of science goals into algorithmic tasks, validation plans, and risk-analysis [1]. NASA's OASIS project investigated the more specific issue of increasing total mission science return by detecting and downlinking only high science-yield data. They use a multi-stage system with several levels of autonomy that can perform tasks ranging from selective data return to opportunistic science and autonomous goal adjustment [2]. A similar team thoroughly investigated the rock detection and analysis segment of such a system. Using image intensity and height map data, they segmented targets for further analysis in categories such as angularity and sphericity. These features were used to understand the geology and geometry of the rocks in the scene [8]. Wagstaff et al took a different approach to the limited bandwidth problem by researching new image compression techniques. The specialized methods were based on using variable compression on various parts of an image so that scientifically interesting areas of the image could be reconstructed with greater precision than irrelevant areas [18]. Other researchers, such as Peters, have focused on the problems of planning and executing science goals in uncertain natural environments. A system is proposed that would be simultaneously able to adjust its behavior on the fly, allow for opportunistic science, and be robust [13]. Finally, Estlin et al have pushed the frontiers of rover science in proposing a system that coordinates and integrates multiple rover planning and scheduling [6].

A systematic and organized approach is necessary to tackle such a large problem. It was decided that a broad overview of computer vision and machine learning algorithms would need to be tested on sample images before any sort of architecture could be discussed. As each algorithm was tested and documented, it became clear that a hybrid approach would necessary and that evaluation metrics would be needed to select optimal (and flexible) algorithms, features, and parameters for the task.

3. Algorithm Survey

An attempt was made to survey the breadth of computer vision techniques that others have used in the past for various forms of rock detection. Many had very little promise and were discounted almost immediately and others were more thoroughly tested until one stood out superior to all others.

Edge detection was one of the first classes of algorithms examined. Simple edge detectors using gradient like Prewitt, Sobel, and Roberts operators were practically useless, picking up a huge amount of noise in the desert environment in addition to the rocks. Preprocessing steps like Gaussian convolution and Bilateral filtering [16] did little to help this problem. Using the Laplacian of Gaussian helped deal with the noise problem, but edges were usually not contiguous and it had to be run multiple times at different scales. A Canny detector was used to try to deal with the problem of continuous lines around rocks. Unfortunately, even with hysteresis, the edges of the rocks were still not contiguous, many rocks were missed, noise was still a problem, and there were many parameters that were difficult to set. Discounting all other problems, since there was no clear way to define possible rock regions in terms of non contiguous edges, edge detection was deemed unfit for the task.

Methods using the geometric, directional, and texture properties of rocks were also tested. The simplest of these is the 2-d Fourier Transform. After trying many modes of analyzing the FFT data from tiles in the image, such as histograms and filtering, it was found that there was not a large enough statistical difference in frequency space to differentiate rock from background in the dataset. Similar tile-based methods were used with Gabor wavelets which work at different scales and also have a directional component. This method ran into similar problems as the Fourier analysis. A geometric approach using the generalized Hough transform was proposed. An implementation was never attempted though, because it became clear that there was no good way to parameterize what a rock looks like – they are certainly not always round, or oval, or regular in any way. Finally, stereo images were experimented upon, clearly displaying large rocks in a height map after a ground plane was fit to the data. Though this did not yield a total solution since it could not find small or distant rocks, it became an important piece in the final system.

Region based methods were examined next to try to fix the failings of the previous categories of algorithms. They produce contiguous regions, unlike an edge detector, and can operate on any attribute or channel of an image. The Watershed transform [10] is one such algorithm that showed a lot of promise. It basically floods an image from its minima and makes "watershed lines" where the different waters meet. Many papers have been written about its successful use in detecting many types of objects, including rocks. Unfortunately, without tremendous amounts of preprocessing, the watershed transform grossly over-segments an image. It also is ill suited for high noise situations like natural scenes – researchers have had much more luck with it in more stable conditions like rocks on a conveyor belt. Region merging, a technique involving the shattering of an image and the merging of homogenous regions, was tried next. It worked very well when homogeneity was defined in terms of intensity, but also worked well (and for different rocks) when the same algorithm was run on the hue and saturation channels of the same image. Therein lies one of the great benefits of region merging; homogeneity can be arbitrarily defined to work with any image channel or even other sensors. The only downside to this algorithm is that it does not find rocks, but only homogenous regions, so some other algorithm has to sort them out. This disadvantage, however, turned out to be a strength, because it allowed machine learning to be used in the problem. It also solves the problem of no one feature being able to describe a rock. Only the fact that it is homogenous and different from its background separates it. This was the main algorithm decided upon for the task. Other variants of region merging like region splitting and split/merge were attempted, but were less effective for various reasons, such as the square region constraint on splitting. A seeded version of region merging was also tested, but there was no good way to find seeds for region growth, since there was no single attribute that would tip us off to where a rock may be.

Finally, a clustering approach was taken to try to solve the problem better and also deal with stereo data, where some data may be missing. A spatially constrained fuzzy c-means algorithm was implemented, which is similar to the more familiar k-means clustering algorithm. The main point of difference between this and k-means clustering is that each point probabilistically belongs to all the classes at once, instead of one class at each update step. Also, spatial distance in the image was part of the distance function in addition to distance in the channel (intensity, hue, saturation, etc.). Using $c = 2$, this algorithm proved to be a good tool for segmenting stereo data, since it can easily handle missing data points, but for the standard channels, region merging still performed better.

A number of preprocessing algorithms were also surveyed to try to improve the quality of the image before the above algorithms were run. Gaussian filters were used to reduce noise in the image, and were fairly effective in doing so, but also blurred over some of the weaker edges. The bilateral filter originally posited by Tomasi and Manduchi was an effective remedy for this problem [16]. By using range as well as domain information, a bilateral filter blurs over noise while preserving strong edges. Unfortunately, it is also computationally expensive, but some good approximations were found that sufficed for our purposes. Another set of algorithms were implemented to deal with lighting in the image and other range issues. Plane and polynomial fit algorithms were tested to correct for lighting gradients in the image, but were found to be expensive and ineffective in improving the end segmentation result. Contrast stretching was found to be useless because of its linear nature, but the non-linearities of histogram equalization proved to help in some of the low contrast images. Finally, color calibration was implemented using a target mounted on the rover to make results more consistent, especially in the later classification stage. The last class of preprocessing algorithms deal with extrema in the image. Adaptive thresholding and extended minima algorithms were used to try to discount certain areas of the image before segmentation began. Since rocks cover the entire spectrum from light to dark, this proved ineffective, even at removing shadows. Therefore, some more specific techniques aimed at only removing shadows were attempted based on hue and texture information, but these failed as well since the entire visible face of a rock was often in deep shadow.

After this rather comprehensive survey, a region merging algorithm used in conjunction with a discriminating algorithm seemed to be the clear choice for a segmentation strategy. To improve results, color calibration, histogram equalization, and an approximation to the bilateral filter were used. This approach uses strong edge information in conjunction with an abstract idea of homogeneity to create the best possible conditions for finding and identifying rock regions. However, the segmentation step outputs a list of homogenous regions that still must be detected as rock or non-rock and classified into geologic sub-types. This need called for a multi-stage architecture.

4. Three Stage Approach

We chose to break the stages up into segmentation, detection, and classification. Segmentation is the process described earlier, of breaking an image into homogeneous regions from multiple image/sensor channels. Detection takes these regions and makes a decision on whether they are rock, soil patch, shadow, or sky. Classification takes each of the rock regions and classifies them into auto-generated and example-driven clusters. This 3-stage architecture creates a very simple flow of data in the system, provides for a very flexible, customizable system, and easily handles new and missing data sources.

Segmentation

Preprocessing is the first major step in the segmentation stage of the code. First, if data is available, color calibration takes place. This is a very simple algorithm in which a periodic image of a color target mounted on the rover is compared with an image taken of the same target under optimal lighting conditions. The proportional differences of the red, green, and blue channels of the two images are used to correct all images within the timeframe of the current periodic image. Next, histogram equalization is run on the image to stretch intensity and enhance edges in washed out (or uniform intensity) images. This is also a very straightforward algorithm that simply uses a non-linear remapping of the image’s intensity values so that their density distribution is as uniform as possible across all possible values. Finally, a bilateral filter is run to blur out some of the small noise in the image, while keeping the edges sharp. This operation is more complex than the others and merits further explanation.

The basic idea behind a local image filter is creating new values for each pixel in an image with some weighted combination of its local neighborhood. A bilateral filter is similar to a Gaussian filter, in that it gives higher weight to geometrically close pixels. A bilateral filter also adds pixel intensity as a factor in the weighting, combining both domain and range filtering. Bilateral filtering can be described by the equations [16]:

$$h(x) = k^{-1}(x)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\varepsilon)c(\varepsilon, x)s(f(\varepsilon), f(x))d\varepsilon$$

$$k(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} c(\varepsilon, x)s(f(\varepsilon), f(x))d\varepsilon$$

where:
f is an image
$c(\varepsilon, x)$ is the geometric closeness of the neighborhood center x and point ε
$s(f(\varepsilon), f(x))$ is the photometric similarity between x and ε
$k(x)$ is a normalizing factor

So in the Gaussian case,

$$c(\varepsilon, x) = e^{-\frac{1}{2}\left(\frac{\|\varepsilon - x\|}{\sigma_d}\right)^2}$$

$$s(f(\varepsilon), f(x)) = e^{-\frac{1}{2}\left(\frac{\|f(\varepsilon) - f(x)\|}{\sigma_r}\right)^2}$$

To give a concrete example, a Gaussian bilateral weighting neighborhood would look like the following if a strong edge was present:

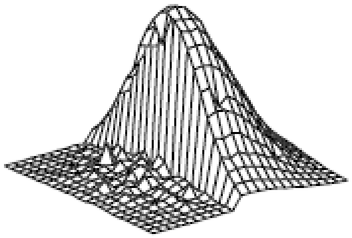

Figure 4.1 – Bilateral Weighting Kernel

Using the above equations, a discrete version of the filter was implemented. It is well known that a 2-d Gaussian convolution can be replicated as two 1-d Gaussian convolutions. This property holds for the bilateral filter as well, and was implemented as such. Also, instead of computing the expensive exponentiation e^x in the range equation (where x is short for the more complicated exponent in the s equation) the approximation 1-x is used. This is much faster and also a sufficiently accurate approximation for our purposes.

Region Merging is the second and main step in the segmentation stage of our system. First, the preprocessed channel (since region merging is run separately on each image and sensor channel) is divided into square sub-regions of an arbitrary size. It was found that 5x5 regions worked best as a compromise between speed, resolution, and statistical significance. Then initial statistics like pixel mean and variance are calculated for each of these regions. There is a data structure that is used to represent this initial region breakup to facilitate fast region merging. Each of the 5x5 sub-regions is actually represented as a single position in a matrix of Region objects. Each region object contains all the statistical information about the region, and also has a parent pointer that begins as null. As region merging occurs and neighboring regions combine, each region only contains one "parent" node that the rest of the member regions point to with their parent pointer – either directly, or through a chain of parent pointers. This parent node is the only one that contains the updated information about the statistics of the regions. This data structure had many benefits. First, it can support arbitrarily shaped regions and quickly access and update the statistical information in one source, without ever having to recompute earlier computations. Secondly, once you follow a pointer chain to the parent node of a region, you can update the parent pointer to point directly at the parent so any future accesses are in constant time. Finally, when a merge occurs, one parent simply needs to point to the parent of the region it merged with. Figure 4.2 illustrates this structure.

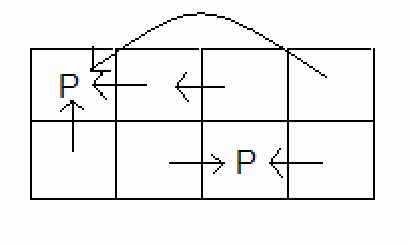

Figure 4.2 – Region Data Structure

After the data structure is initialized, the iterative region merging process begins. Each region is iterated through and is a candidate to merge with any neighboring regions (4-connectivity). So for each candidate merge, a simple distance metric is used to determine how similar the regions are to each other. Difference of means is an example of a criterion that can be used. Then, if the regions are similar enough within a threshold, the regions are merged together into one region. Iterative passes are made over all the regions in this manner until there is one full pass with no active merges. Surprisingly, this simple algorithm is usually sufficient to find all the homogenous regions in an image with reasonable accuracy. Unfortunately, there are some cases where the merging criterion is too simple, especially for large rocks with a lot of surface variance. To remedy this, the t-statistic from a Welch-modified t-test is used as the new merging criteria after normal region merging terminates. This is also run until no more merges occur. The Welch-modified t-test is designed to compare populations where the variance is not assumed to be the same. By using a more complex statistical analysis than our previous merging criterion, this test merges some similar areas that were passed over before. However, this test only works well as a second pass, because the regions are not large enough in the beginning to produce good statistics for the test. The Welch modified t-test can be described by the following equations:

$$df = \frac{v_1/n_1 + v_2/n_2}{(v_1/n_1)^2/(n_1-1) + (v_2/n_2)^2/(n_2-1)}$$

$$s_p = \sqrt{\frac{(n_1-1)\cdot v_1 + (n_2-1)\cdot v_2}{df}}$$

$$s_e = s_p \cdot \sqrt{v_1/n_1 + v_2/n_2}$$

$$t = \frac{|\bar{x}_1 - \bar{x}_2|}{s_e}$$

where:
n is the size of a population
v is the variance of a population
$\bar{x}$ is the mean of a population

After the t-test is complete, segmentation is finished. The homogenous regions are put into a list and given to the detection stage for further processing. Regions touching the sides of the image are thrown out because they either constitute background or a partially visible rock on the edge of the image, which we do not want to deal with. An example of segmentation follows in figure 4.3.

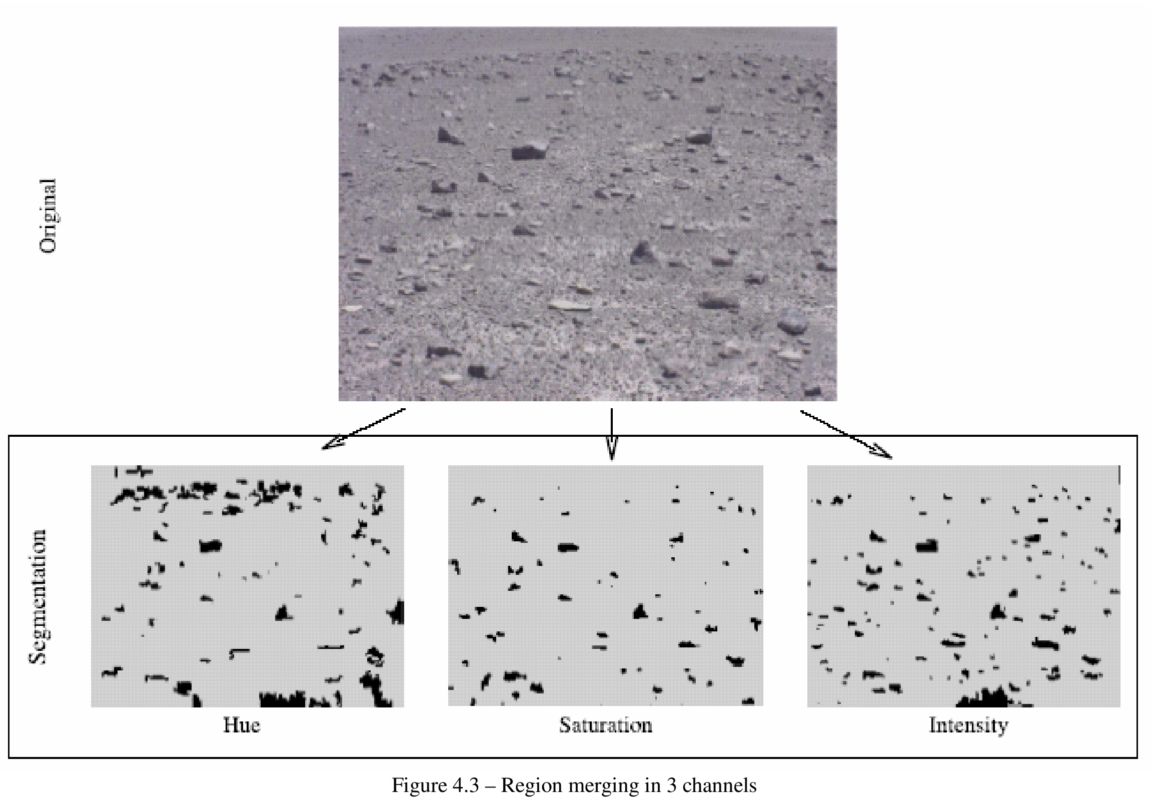

Figure 4.3 – Region merging in 3 channels

Detection

The detection stage of the system identifies regions as one of the following: rock, soil patch, shadow, or sky. A Bayesian belief network is used to make these classifications. Bayesian classification offers many features that lend themselves to our particular kind of task. Belief networks allow for the graceful and natural handling of missing data. So, for example, if there is no stereo or spectrometer data available for a particular image, it does not matter, though more data is better. Also, once trained, these networks only require a few mathematical operations to calculate a posterior probability, making them very fast. A Bayesian approach also allows a prior class probability to be specified, thereby enabling both high precision and recall behaviors based on preference. This will be important in future work done with scientist preference interfaces. Our Bayesian network relies on a set of discretized features drawn from the pixels belonging to each region and a rectangular set of context pixels around the region as shown in figure 4.4.

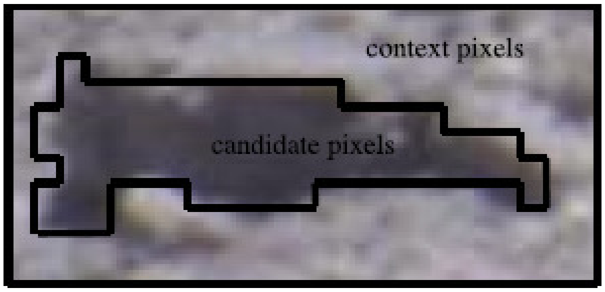

Figure 4.4 – Candidate and context pixels

For each region, a feature vector is calculated using the following features (some of these are used also/only for geological classification as opposed to detection) [15]:

Perimeter: The ratio of the region's squared perimeter to its pixel area. Non-rock artifacts often have long, spidery shapes while rocks tend to be more convex and ellipsoidal.

Relative Color: The absolute value of the difference in mean pixel hue, saturation, and intensity between the interior of the region and the context region. Fisher distance (which weights the score according to the variance of each sample) performed better as a difference metric than a simple comparison of means.

Relative Color Variance: The absolute value of the difference between the pixel variance of the interior region and the pixel variance of the context region. This functions as a simple measure of texture, helping to detect situations like a smooth rock sitting in rough, high-variance gravel or a pitted rock on a smooth background.

Height Above the Ground Plane: While small rocks are generally below the noise threshold for our stereo system, height is a valuable attribute for detecting large rocks and excluding large non-rock regions.

Texture: We use a fractal dimension measure [4] of a binary intensity map to describe the detail of each region as resolution is increased. The result is an efficiently-computed value that corresponds somewhat to our intuitive notion of surface roughness. Like color variance, this helps to detect rocks in those cases where their roughness differs substantially from the background sediment.

Intensity Gradient: Rocks are three-dimensional protrusions that exhibit shading when illuminated by sunlight. We use least-squares regression to find the magnitude of the intensity gradient over the pixel surface of each region in the image, giving the overall strength of its shading.

Physical Location: While not implemented in the current field test, future versions will include an estimate of the rock's position in the world. A region's location should not affect its classification as a rock, but it should play into the autonomous geological classification—a familiar looking rock can still be geologically interesting if it is found in an unexpected place.

Absolute Color: Another attribute that informs geological classification. We exclude it from the attribute set for rock/non-rock classification in order to maintain generality.

Two additional attributes do not directly affect a region's chances of being a science target or its geological classification. Nevertheless, we include them due to their strong conditional dependence relationships with the other attributes. By considering these dependencies the Bayesian network computes a more accurate posterior probability.

Absolute Range: Many of the differences between rocks and non-rocks become less apparent as range increases. In particular, texture is more difficult to see even when there are many pixels representing the distant region.

Pixel Area: There are varying degrees of conditional dependence between most observed attributes and the regions' pixel area. These dependencies are due to the way regions are represented as a finite number of pixel "samples." For example, normalized perimeter is generally small for regions of small pixel area because rough borders become less noticeable when the number of pixels used to describe them decreases. Similarly, texture is hard to recognize with few pixel samples.

To train the network, we use an example based system where a graphical tool is used to define regions found by the segmentation algorithm as rock, soil patch, sky, shadow, or undefined. The feature vectors for each of these regions are then calculated and used to constitute an exemplar for that particular class. Figure 4.5 shows a picture of the detection training tool.

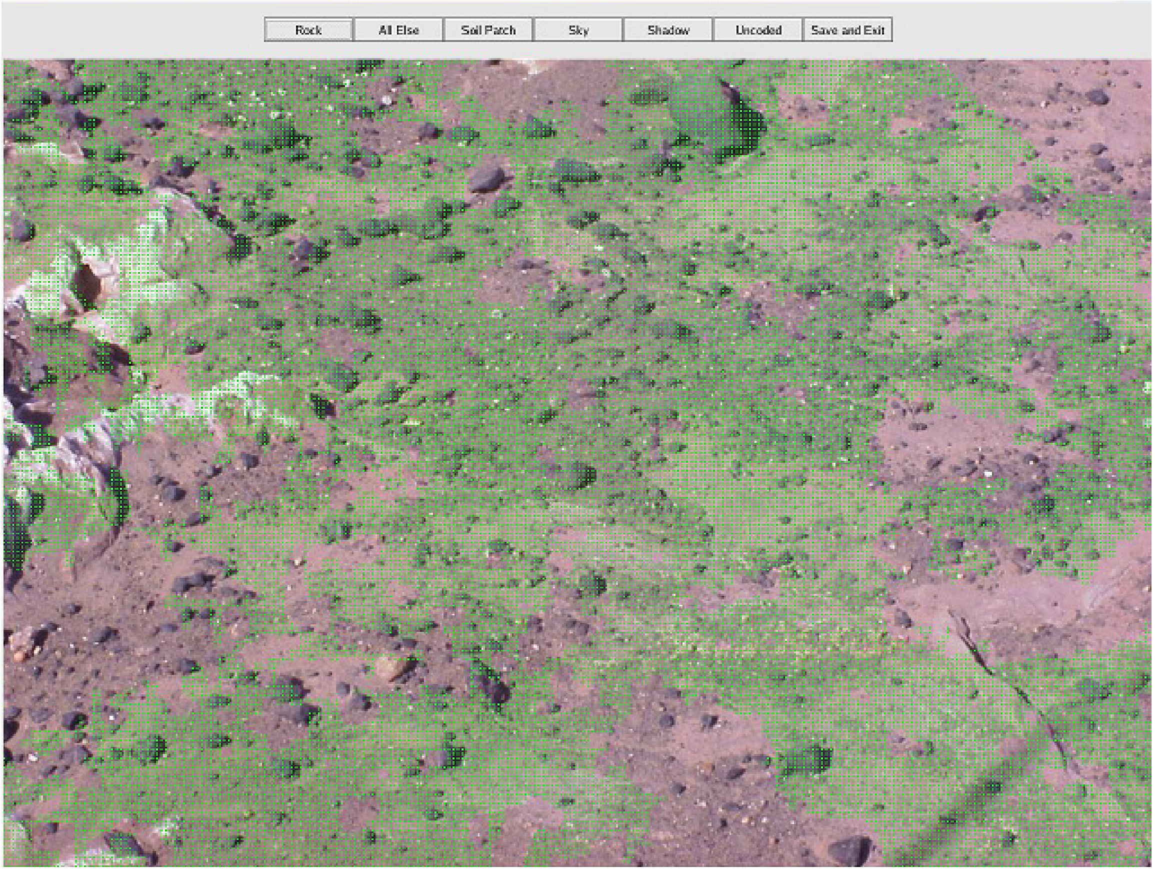

Figure 4.5 – Detection training tool

Once the network is trained, a feature vector $X = \langle x_1, x_2, ..., x_n \rangle$ is calculated for each region given to the detector and class probabilities are generated as follows:

$$P(C \mid X) = \alpha P(C) P(x_1 \mid C) P(x_2 \mid C) ... P(x_n \mid C)$$

where α is the prior probability of class C

This is the case of naïve Bayes where the largest class probability C is the detector's output class. But it turns out we can do better than this by using statistical dependencies between network features [11]:

$$P(C \mid X) = \alpha P(C) P(x_1 \mid C, x_2) P(x_2 \mid C) ... P(x_n \mid C)$$

would be an example if x_1 was dependent upon the value of x_2

Figure 4.6 shows a description of the full network topology.

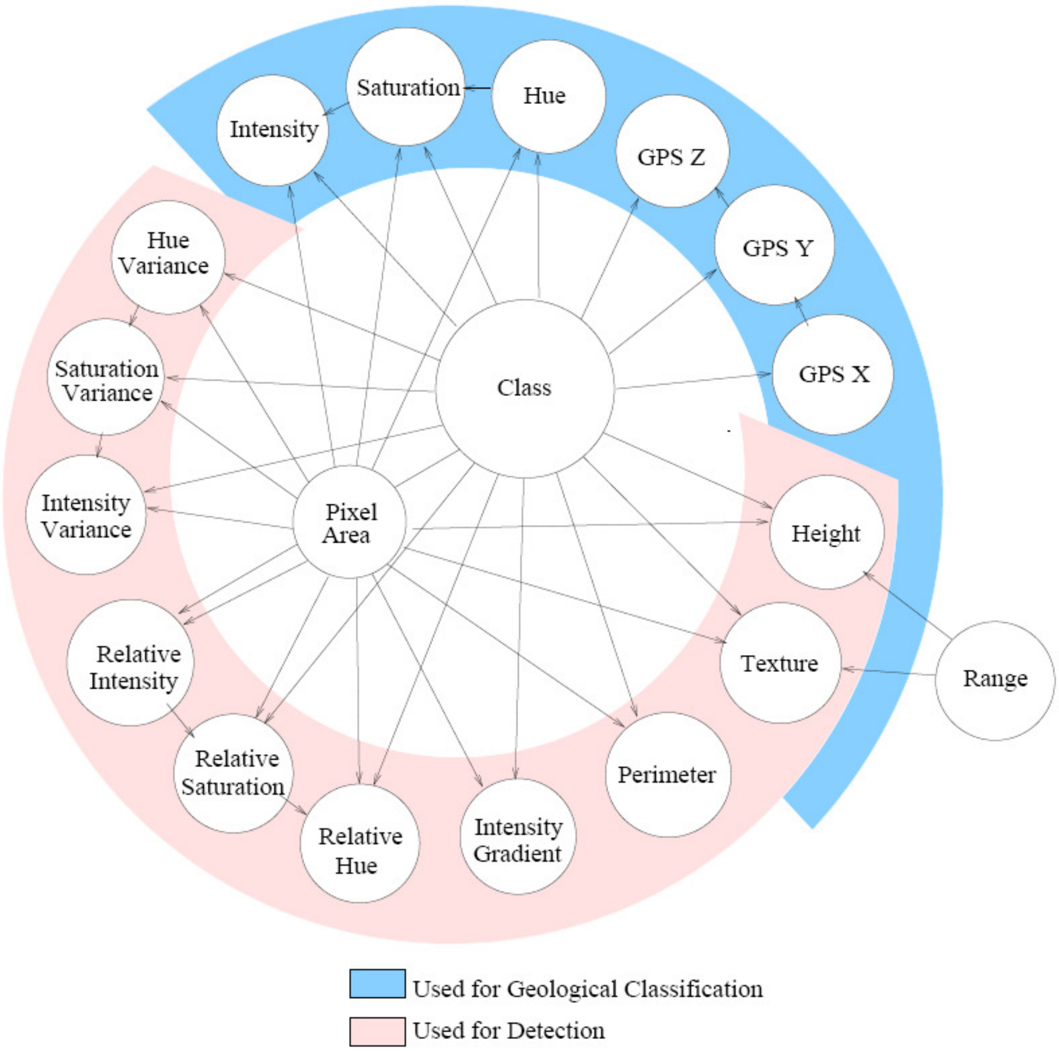

Figure 4.6 – Network topology showing all feature dependencies

After the detector outputs all the class probabilities, it creates a list of all the regions that it classified as "rock" and gives it to the geologic classifier for further processing. Figure 4.7 shows the end product of detection with outlines of detected rocks. Figure 4.8 also shows a cell shading system that was developed to deliver rock information to scientists in a low memory, low bandwidth representation.

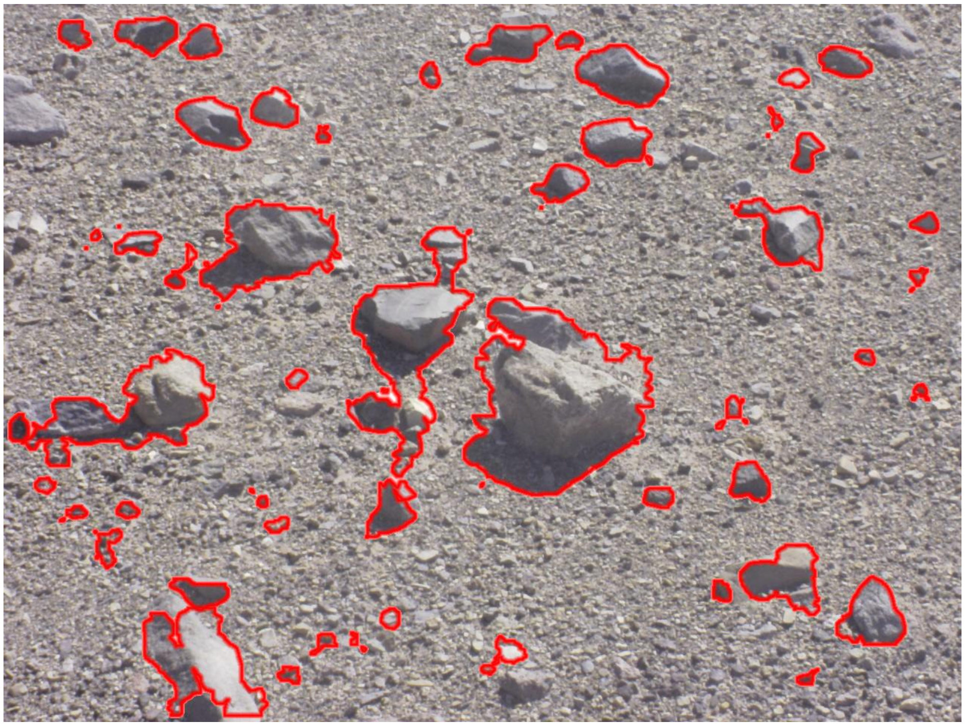

Figure 4.7 – Detection results

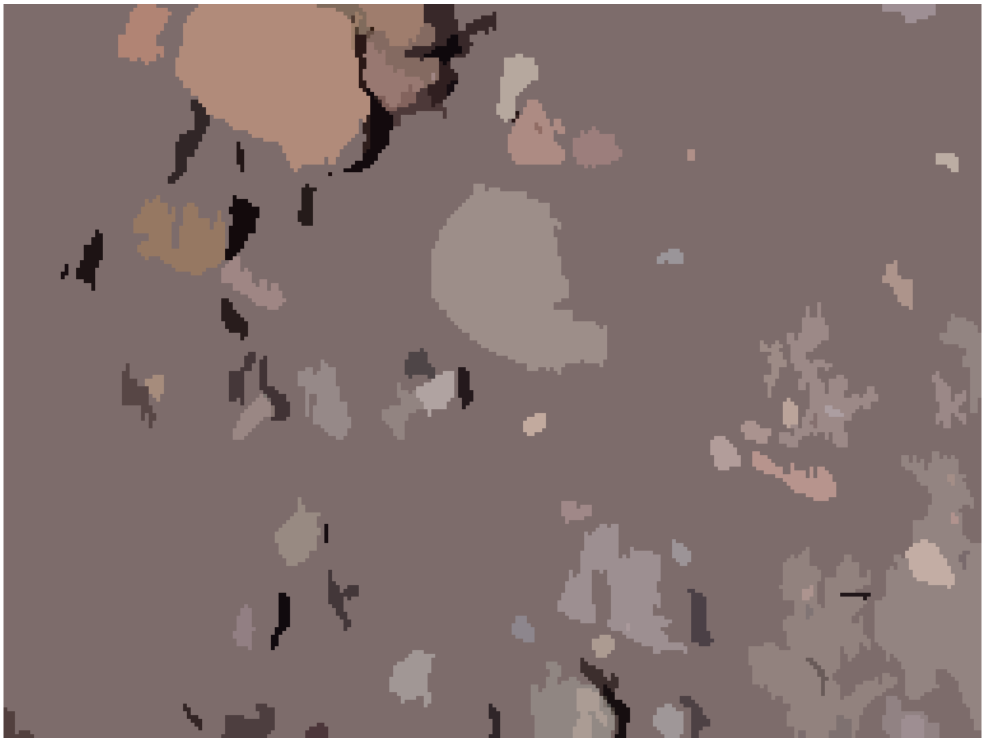

Figure 4.8 – Detection with cell-shading

Classification

The classification stage is designed to create organized feedback about geologic types of rocks (and eventually soils and other features) in the scene. To achieve this, we use Expectation-Maximization, an algorithm that utilizes movable Gaussian clusters to cluster a set of data [5]. The feature set used is noted above in the detection section. The algorithm is run on this feature set and autonomous classes are generated using cross-validation to select an appropriate number of classes. Another desired feature, however, is to be able to have user specified classes. A classifier tool was developed to allow the manipulation of detected rocks into an arbitrary number of classes. Unfortunately, due to Gaussian constraints in feature space, not all the rocks can always be put in the specified classes, so the user specifies a class breakdown, the class centers are updated, and then adjustments can be made. The feature set is then also used to classify all the rocks into these pre-generated classes as well as autonomous classes. This allows a wide range of behaviors from anomaly detection using the autonomous classes, to representative sampling and specific geologic type detection using the hand-made classes. Figure 4.9 displays geologic classes from images in a list format.

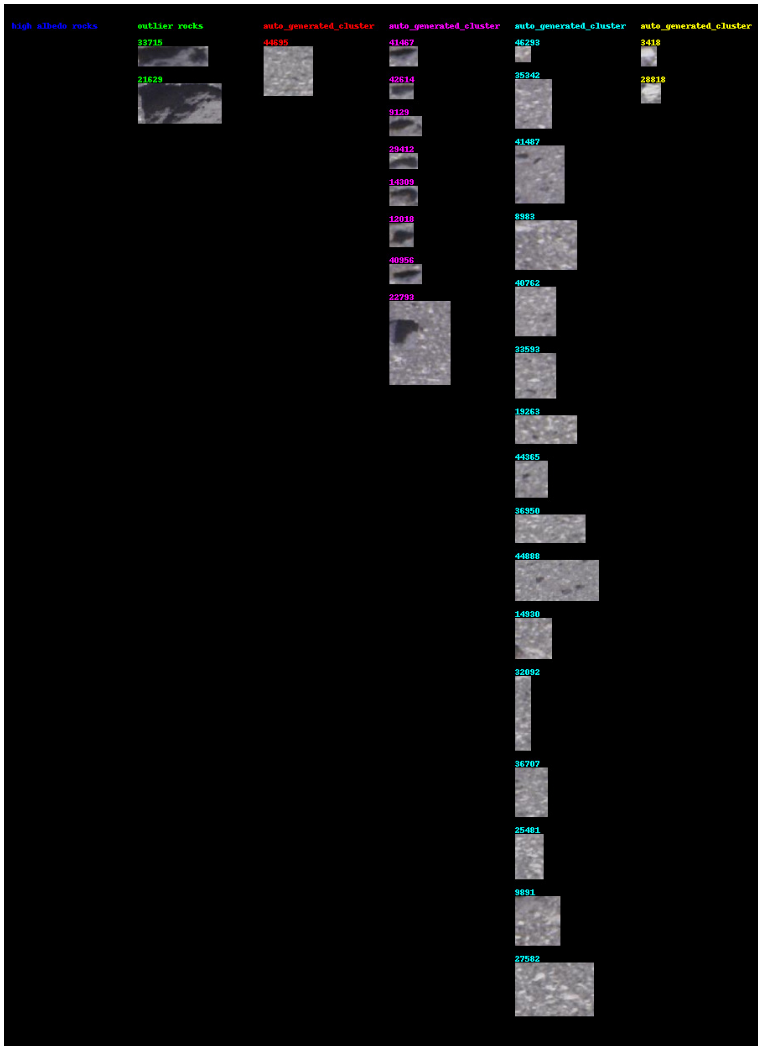

Figure 4.9 – Geological Classifications

A three-stage hybrid system has been presented that meets our criteria for functionality. The image is preprocessed for optimal rock detection, then segmented into homogenous regions, which are then passed through the detection and classification algorithms. Figure 4.10 shows an overview of the system. This process yields both an accurate and flexible system that can handle many sensor channels, deal with missing information, and can produce both high precision and recall behaviors without retraining.

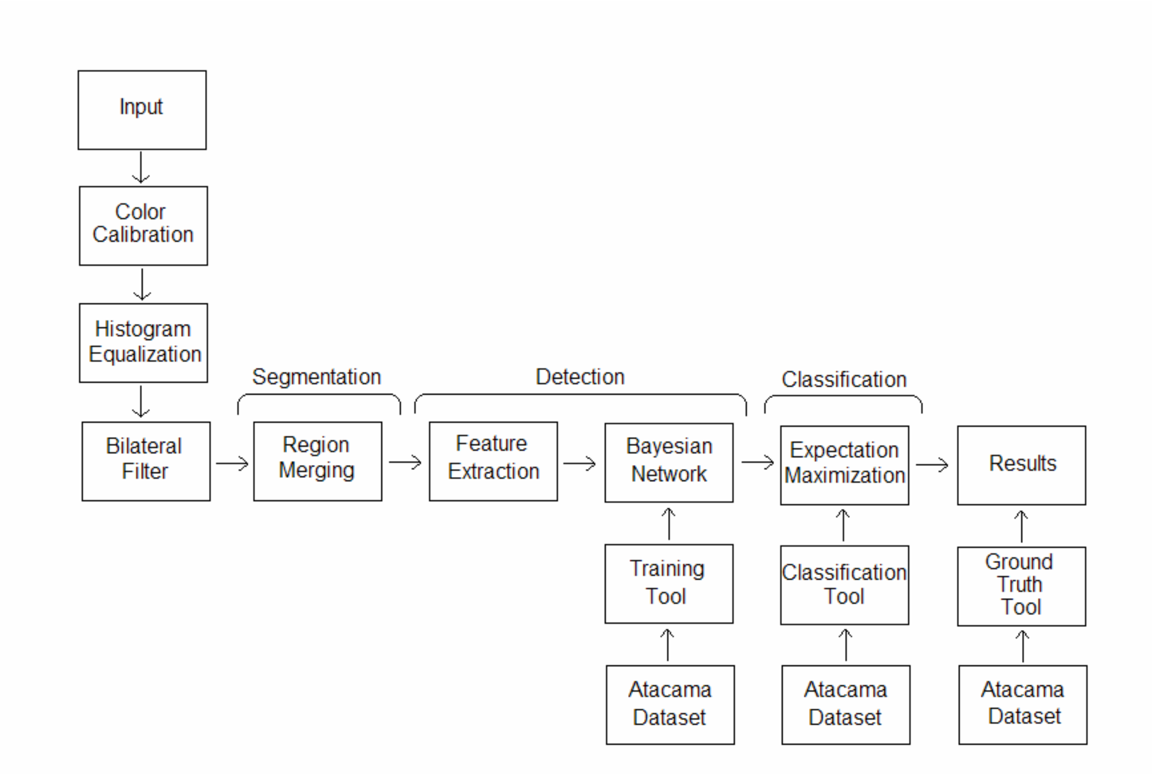

Figure 4.10 – 3-Stage System Architecture

5. Results

Evaluation of the hybrid system and tuning of system parameters was a high priority, yet a difficult procedure. Evaluation metrics, a realistic dataset, ground-truthing methods, and relevant statistics were all necessary. Using a dataset collected in the Atacama Desert of Chile, we evaluated the system's performance on periodic traverse images and panoramas. It was found that the system has both high precision and recall on ground-truthed data and is suitable for use in a science autonomy system.

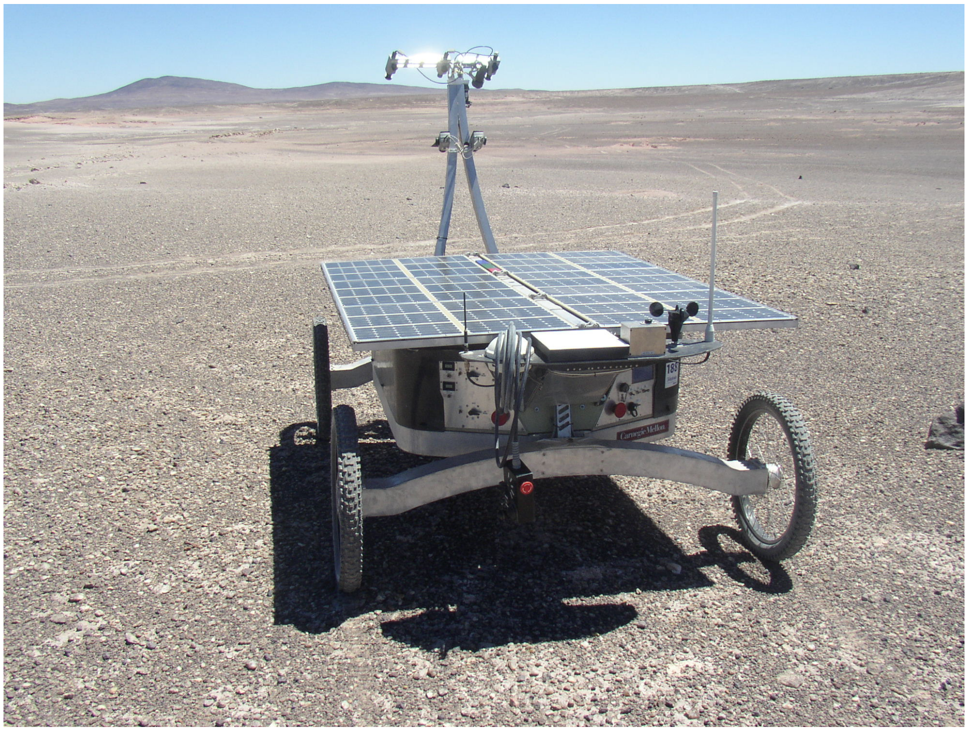

Figure 5.1 – Zoë in the Atacama Desert

Since the project is focusing on finding life in the Atacama Desert, we decided to go there to collect a realistic, traverse-based dataset instead of using other available datasets such as MIR and Mars Yard data. The goal of data collection was to assemble a dataset that could be used long-term to evaluate the current and future capabilities of the science autonomy system. This covers everything from rock detection accuracy, to classification and selective data return, to future pursuits such as detection of geological boundaries during traverse. To achieve these objectives, we collected two kinds of datasets. The first type is a series of periodic camera images at a standard angle during a long distance traverse. The second type is a series of panoramic images (spaced much further apart than the periodic images) where the rover stops during traverse to capture them. This is the kind of data that would be expected to be available during a mission, so that we can "replay" the traverse as a simulation later on, even as the software changes, allowing us to test as though we were in the desert.

Once a large dataset had been obtained, ground-truthing methods and evaluation metrics were developed. To ground-truth the data, we found it would be too difficult and subjective for a human to go through and outline each and every "rock" (or what they thought was a rock) in every image that we had. Instead, a tool was devised to assist in this task and remove human bias. Images were presented to a human coder via this tool and 50 rocks were labeled in each image as ground-truthed rocks. To fairly choose a population of 50 rocks in each image, the tool highlights a random point in the image during each iteration. The human coder is then supposed to draw a bounding box around the nearest rock to that point. We only considered areas larger than 20x20 pixels as a rock, and anything above 75x75 pixels as a "large" rock. Once the bounding box is drawn, the tool draws a circle from the random point to the near corner of the bounding box. This circle signifies that there are no rocks in the circular region, since the coder bounded the closest rock. Therefore, we can say that if the system thinks there is a rock in this region, it is a false positive. Conversely, if the system things there is a rock where we ground-truthed a rock, then we know that a rock was detected successfully. Detection in non-coded regions can be ignored. A diagram of the workings of this coding tool is shown in figure 5.2.

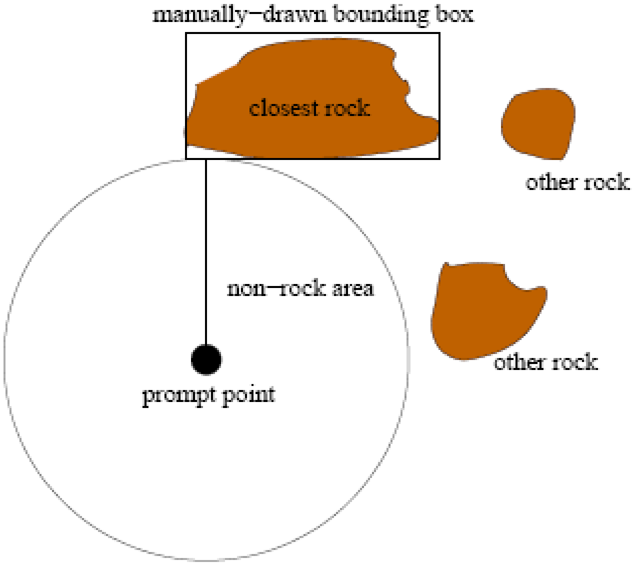

Figure 5.2 – Coding tool

The question remains how to decide if a detected rock is "within" a ground-truthed rock or non-rock area. Two different metrics are used to more fully describe system performance. If a detected rock's center lies in either of the ground-truthed areas, we say it is "within" it. If the center falls with a ground-truthed rock area, this is called a true positive, and if it is within a non-rock region it is a false positive. The second metric checks to see if the bounding box of the detected rock and the bounding box of the ground-truthed rock overlap by at least 50%. If they do, this is called a "well localized rock". Figure 5.3 demonstrates these 2 metrics.

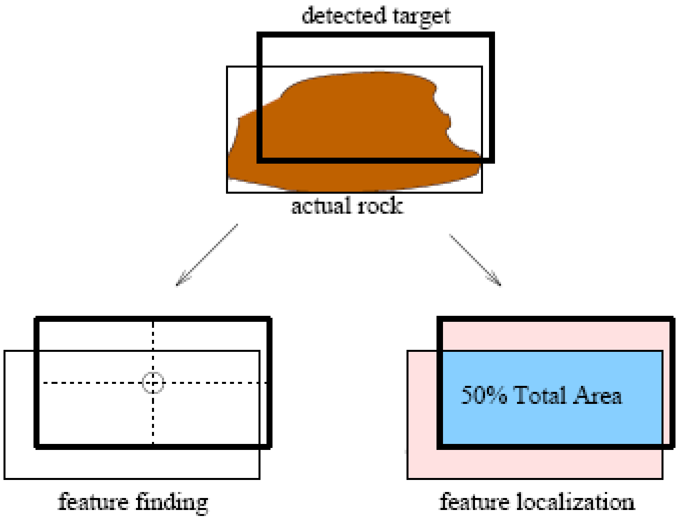

Figure 5.3 – Detection metrics

A test set of 30 images from a traverse was coded using the ground-truth tool and used to evaluate the performance of the detection system. Accuracy of the system was measured using precision and recall. Precision is the fraction of regions the detector flags as rocks that are actually rocks. Recall is the fraction of the ground-truthed rocks that the detectors flags as rocks, using the detection metrics discussed earlier. Generally, these quantities are inversely proportional. The test set was run through the detection system multiple times, each time with a different prior probability given to the Bayesian network of finding a rock. As expected, the higher the prior probability, the higher the recall and the lower the precision, since more regions are accepted as rocks. Although it should be noted that as prior probability goes up, recall must monotonically increase, but precision can increase or decrease. Figures 5.4 - 5.6 show median system performance across images (with error bars) against prior probability of a rock.

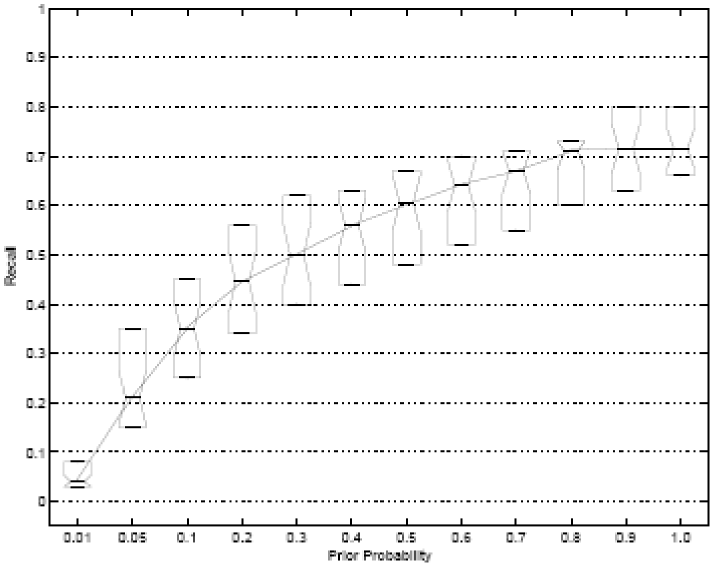

Figure 5.4 – Recall vs. Prior probability

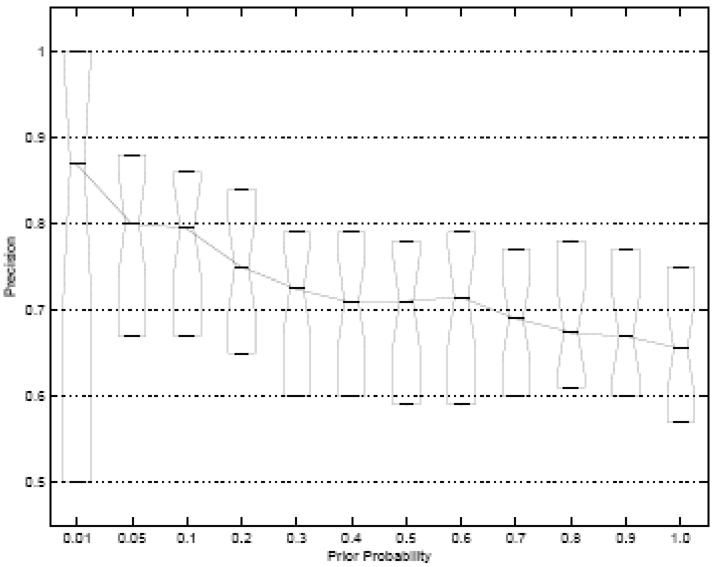

Figure 5.5 – Precision vs. Prior probability

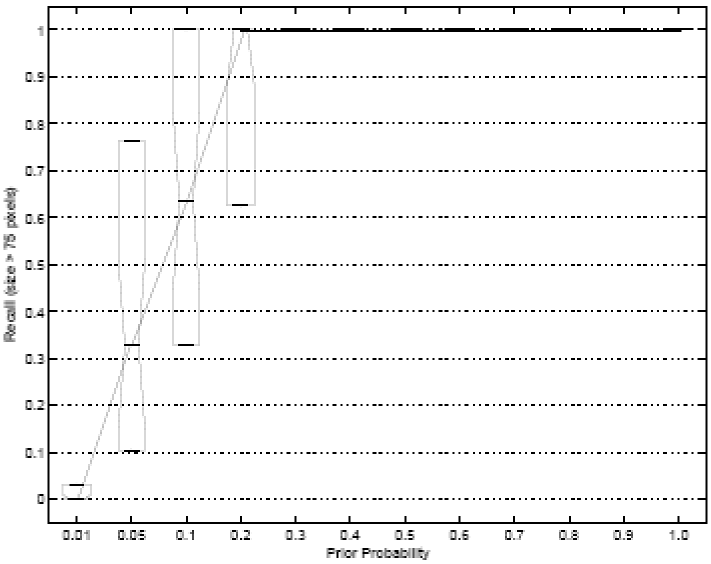

Figure 5.6 – Recall vs. Prior probability on large rocks

It can be seen that even when the prior probability is 1, there is not always 100% recall. This is because rocks can only be detected if the segmentation algorithm first separates them as candidate regions. This data indicates that our segmentation system is slightly more than 70% effective. Looking at both the precision and recall graphs, a prior probability of approximately 0.6 will yield a recall of around 65% and a precision greater than 75%. Though maximally nearly 90% precision can be achieved, this setting is a good compromise of the two attributes. Figure 5.6 shows the recall on large rocks to be much better than the general case, at a perfect 100% any time the prior probability is greater than 0.2. All the above statistics are with respect to 'detected' rocks. Figure 5.7 shows 'well-localized' rocks with respect to prior probability which, as expected, is significantly lower.

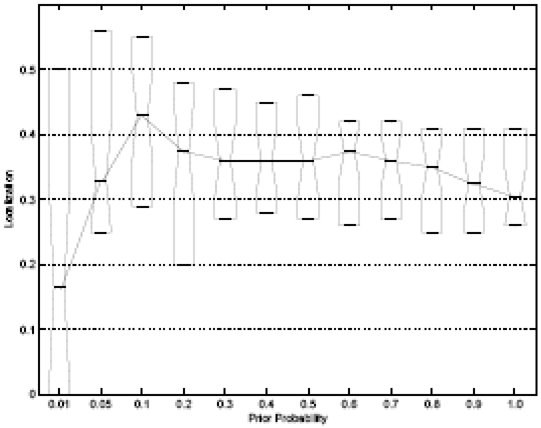

Figure 5.7 – Well-localized recall vs. prior probability

Though no quantitative analysis has been done to evaluate classification results, qualitative evidence from looking at classified images from the test set and input from geologists seems to suggest that it is doing something reasonable, at least in regard to albedo and size. Figure 5.8 shows an annotated image where outline color corresponds to class.

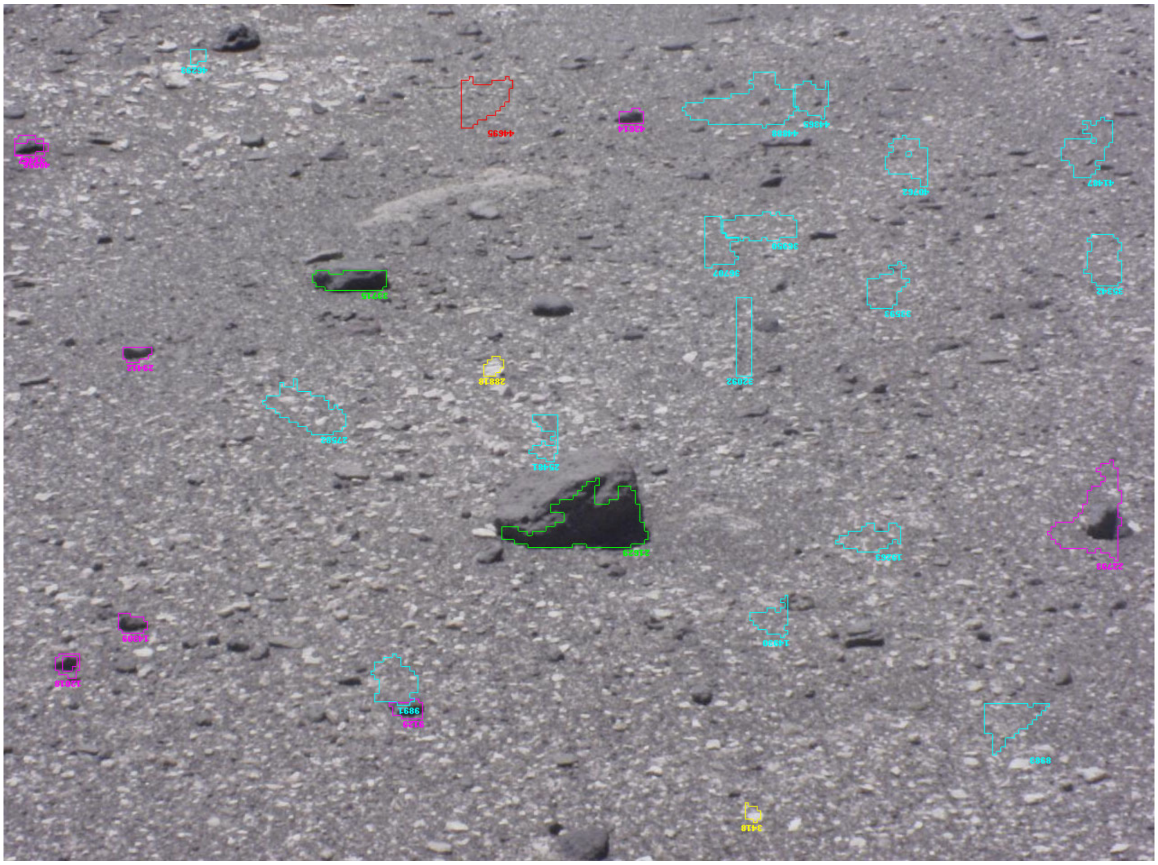

Figure 5.8 – Annotated Classifications

6. Conclusions

The experiments outlined above demonstrated that the rock detection system reliably localizes the majority and most significant rocks in natural scenes. Visualizations of the detection and classification stages corroborate that the system is performing in a reasonable manner. The three-stage detection architecture meets the needs of the problem well – namely, it is flexible with respect to sensor type and missing data, it can elicit both high precision and recall behavior, and it generalizes well to a variety of rock types, terrain, and conditions. We believe this is because the system uses a combination of computer vision and machine learning techniques which capitalize on the idea that there is no constant set of characteristic that can separate rock from background.

There are still many issues to be addressed, as this is a first implementation of the approach. It has been seen that many rocks can be detected easily, but it is much more difficult to get them "well-localized". Also, performance quickly degrades with the size and distance of the rocks in question. To improve the performance of the detector, a more accurate segmentation algorithm would be needed boost the recall closer to 100% when the rock prior is 1. The use of the Watershed algorithm to create the initial regions for region merging (as opposed to shattering into 5x5 blocks) has been suggested to preserve strong edges. Next, improved stereo segmentation would help localize larger rocks. Some early experimentation has been done using the fuzzy c-means algorithm for this. Finally, more features will improve the detection and classification stages. One area for extension is a more descriptive measure of texture. Improved shadow removal using sun direction has also been discussed.

Currently, the 3-stage detection architecture is being expanded and fit into a larger science autonomy system. A scientist preference interface for selective data return is being developed along with graphical displays to visualize data in an intuitive way. The detection system is also being expanded to deal with other science targets like soil and salt patches, gravel beds, and rock layers. A science planner will plan and re-plan science goals on the fly. When this is integrated with the rock detection system, we will have a complete science autonomy capability that can push the limits of traverse distance, quality of science data return, and the ability to find life with robotic explorers.

References

[1] R. Castaño, M. Judd, R. C. Anderson, T. Estlin, "Machine Learning Challenges in Mars Rover Traverse Science". Workshop on Machine Learning Technologies for Autonomous Space Applications, *International Conference on Machine Learning*, Washington, D.C., August 2003.

[2] R. Castaño, R. C. Anderson, T. Estlin, D. DeCoste, F. Fisher, D. Gaines, D. Mazzoni, and M. Judd, "Rover Traverse Science for Increased Mission Science Return," *Proceedings of the IEEE Aerospace Conf.*, Big Sky, Montana, 2003.

[3] A. Castaño, R. C. Anderson, R. Castaño, T. Estlin, and M. Judd, “Intensity-based rock detection for acquiring onboard rover science,” *Lunar and Planetary Science*, 35, 2004.

[4] B. B. Chaudhuri, N. Sarkar, “Texture segmentation using fractal dimension,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:1, 72—77, January 1995.

[5] A. P. Dempster, N. M. Laird, D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society*, B, 39, 1-38.

[6] T. Estlin, T. Mann, A. Gray, G. Rabideau, R. Castaño, S. Chien and E. Mjolsness, "An Integrated System for Multi-Rover Scientific Exploration," *Sixteenth National Conference of Artificial Intelligence (AAAI-99)*, Orlando, FL, July 1999

[7] C. Farfán, R. A. Salinas, G. Cifuentes, “Rock segmentation and measures on gray level images using watershed for sizing distribution in particle systems.”

[8] J. Fox, R. Castaño, R. C. Anderson, “Onboard autonomous rock shape analysis for Mars rovers,” *Proceedings of the IEEE Aerospace Conf.*, Big Sky, Montana, 2002.

[9] V. Gor, R. Castaño, R. Manduchi, R. C. Anderson, and E. Mjolsness, “Autonomous rock detection for Mars terrain,” *Proceedings of AIAA Space 2001*, Albuquerque, August 2000.

[10] S. Mkwelo, G. De Jager and F. Nicolls, “Watershed-based segmentation of rock scenes and Proximity-based classfication of watershed regions under uncontrolled lighting conditions,” 2003.

[11] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference* (San Francisco: Morgan Kaufmann), 1988.

[12] L. Pedersen, *Robotic Rock Classification and Autonomous Exploration*, PhD thesis, Robotics Institute, Carnegie Mellon University, CMU-RI-TR-01-14.

[13] Stephen F. Peters, "A Science-Based Executive for Autonomous Planetary Vehicles," 6th i-SAIRAS Conference Proceedings, Quebec, Canada, June 18-22, 2001.

[14] Trey Smith, Scott Niekum, David Thompson, and David Wettergreen, “Concepts for Science Autonomy during Robotic Traverse and Survey,” *Proceedings of the IEEE Aerospace Conf.*, Big Sky, Montana, 2005.

[15] David Thompson, Scott Niekum, Trey Smith, and David Wettergreen, "Automatic Detection and Classification of Features of Geologic Interest," *Proceedings of the IEEE Aerospace Conf.*, Big Sky, Montana, 2005.

[16] C. Tomasi, R. Manduchi, "Bilateral Filtering for Gray and Color Images," *Proceedings of the Sixth International Conference on Computer Vision*, 1998.

[17] M. D. Wagner, D. Apostolopoulos, K. Shillcutt, B. Shamah, R. G. Simmons, W. Whittaker, "The Science Autonomy System of the Nomad Robot," *ICRA 2001*, 2, 1742—1749.

[18] K. L. Wagstaff, R. Castaño, S. Dolinar, M. Klimesh, R. Mukai, "Science-based region-of-interest image compression," *Lunar and Planetary Science*, 35, 2004.

Appendix – Algorithm Details

Class: Body algorithms
Type: Edge Detection
Goals: With appropriate preprocessing, decide on appropriate edges and object boundaries. Possibly determine what edges are "strong" enough and what objects are large enough to keep.

Roberts Operator
Desc: Provides an approximation of the gradient magnitude with a cross operator. This is the addition of convolution with the matrices $Gx = [1\ 0\ ;\ 0\ -1]$ and $Gy = [0\ -1\ ;\ 1\ 0]$.
Pros: Simple approximation of the continuous gradient, computationally cheap.
Cons: Since it uses 2x2 matrices the gradient is approximated at an interpolated point.

Prewitt Operator
Desc: Provides an approximation of the gradient magnitude with a 3x3 neighborhood. 2 matrices are computed based on partial derivatives. The equations are:
$Sx = (a2 + ca3 + a4) - (a0 + ca7 + a6)$
$Sy = (a0 + ca1 + a2) - (a6 + ca5 + a4)$
$M = sqrt(Sx^2 + Sy^2)$
Where $c = 1$;
Pros: Finds gradient at a real point.
Cons: Weights points equally, no matter how far from the center of the mask they are.

Sobel Operator
Desc: Same as prewitt, but $c=2$. Two matrices are computed based on partial derivatives.
Pros: Weights closer points to center of mask more strongly. Finds gradient at a real point.
Cons: Same as most edge detectors – simple and sensitive to noise.

Second Derivative (Laplacian)
Desc: Assume that edges occur at zero crossing of the second derivative.
Pros: Equivalent finding peaks in first derivative.
Cons: Extremely sensitive to noise, even small local peaks.

Laplacian of Gaussian
Desc: To get rid of noise, first smooth with a Gaussian. Then use a laplacian operator as above, but only make edges where there is both a zero crossing, and a first derivative above a certain threshold.
Pros: More robust than normal Laplacian in terms of noise and edge strength acceptance. Can be calculated as one filter (Mexican hat) instead of 2 convolutions, making it cheap to implement.
Cons: Gaussian can also blur some edges, causing them to disappear. Also, only finds edges at a certain resolution based on size of operator – may need to use multiple resolutions.

Canny Edge Detector

Desc: First convolves image with the first derivative of a Gaussian to smooth and calculate the gradient. Then uses heuristics like nonmaxima suppression, hysteresis, and double thresholding to get rid of noise, detect, and connect the edges.

Pros: Complex detector finds edges more reliably and intelligently and is tunable based on parameters of the Gaussian, thresholding, etc.

Cons: Mathematically expensive and slow.

Class: Body algorithms
Type: Property detection
Goals: Detect rocks in the image using known properties about rocks in general, such as color, texture, shape (geometry), and size.

Fourier Transform
Desc: Transform that goes from image space into frequency space.
Pros: Fast, and good for texture and size analysis, and histogram characterization of a scene. Information preserving - can recover original image with an inverse transform. Also useful for estimation of the number of small uncountable rocks based on area and frequency information.
Cons: No directional component. Not useful in all scenes.

Wavelet Analysis
Desc: Similar to an FFT in some ways, divides up the image into frequency components, but then analyzes each component based on its scale. Also localized in space.
Pros: Better than FFT for images that contain objects of many sizes and sharp discontinuities and peaks. Multi-resolution and localized.
Cons: Generally slower than FFT analysis.

Gabor Wavelets
Desc: Wavelets with a directional component.
Pros: Excellent for texture detection when multiple directions and resolutions are histogrammed, due to the directional properties of most textures.
Cons: Some textures have no directional component and cannot be characterized well by Gabor wavelets.

Hough Transform
Desc: This is used when a certain structure, like a circle is believed to be contained in a image. A parameterization of the object that is being searched for is provided. Each pixel basically 'votes' for a place in parameter space that may represent it. A highly voted area is parameter space is hopefully a fairly accurate model of the structure being search for. This is accomplished by looking at areas of high intersection in 'hough space' where points are represented as parametric curves.
Pros: If the object being searched for can be reasonably described, this is an effective geometric method. Excellent at dealing with noisy images and multiple objects.
Cons: Can only deal with easily parameterized shapes like lines, circles, and ellipses, not irregular geometric shapes.

Generalized Hough Transform
Desc: High transform that uses a lookup table instead of a parametric equation to describe the boundary of the target object type.
Pros: The algorithm can now deal with irregular object geometries.
Cons: Still only works for one particular oddly shaped object, not a general class of them.

Color thresholding

Desc: Basically the same as normal thresholding, but in hue and saturation space instead of intensity space.

Pros: Good, if rocks are significantly different from the background in hue, but intensity does not vary much, or lighting conditions are poor.

Cons: Assumes that all rocks fall in a certain area of color space and background does not. Carries all the same problems of choosing cutoffs and dealing with noise as in regular thresholding.

Color reduction

Desc: Region based segmentation where like, nearby colors are merged into one homogenous region of an average color.

Pros: See region segmentation document – this is just a special case.

Cons: See region segmentation document – this is just a special case.

Class: Body algorithms
Type: Region-based
Goals: Using local and global constraints of homogeneity, these algorithms should grow or divide some starting points/regions in an optimal way until every pixel in the image belongs to a stable region, achieving segmentation of the image.

<u>Watershed Transform</u>
Desc: 'Floods' image from minima. Creates 'watershed lines' to keep different waters from merging. Segments entire image into regions and boundary pixels.
Pros: Industry standard for rock detection and similar segmentation problems. Very effective in finding rocks and their size and boundaries, as well as boundaries of the nonrock
regions.
Cons: Requires a huge amount of preprocessing that must be very accurate. First a good set of markers must be obtained, which is a task in itself. Then, the markers must be imposed, while downplaying other shallow minima. Then an operation has to be performed to get rid of plateaus and make values strictly increase as they radiate from each minima to a boundary. A distance transform works well here. Finally, if all the above goes well and good values are picked for each transform, then the watershed transform should be effective.

<u>Recursive Region Growth</u>
Desc: Begin with a seed pixel chosen randomly or by a metric. Recursively add pixels surrounding this pixel to this 'region' based on a similarity metric like standard deviation. Continue until no more pixels are added to the region. Pick a new seed pixel and do this for another region. Repeat until all pixels belong to a region.
Pros: Segments based on local characteristics, and can segment well with a good starting point and similarity metric. Complex similarity metrics can be used, such as combinations of intensity, texture, color, etc.
Cons: Only one region is grown at a time, so ambiguities at edges often are resolved incorrectly, or a region expands further than it should. Seeds and similarity metrics can be hard to choose, and problems can arise if the seed is chosen on an edge.

<u>Simultaneous Region Growth</u>
Desc: Same as recursive region growth, but multiple seeds are chosen at the beginning, and the regions are grown in parallel. Also takes adjacent regions, not just pixels into consideration, so that similar regions can also merge into one.
Pros: All the benefits of recursive region growth, but helps prevent over-growth by growing regions in parallel. Also forces the algorithm to pay attention to more global, not just local characteristics of the image.
Cons: The number of seeds you choose at the beginning is an upper bound on the number of regions you can have at the end, unlike recursive growth that can have any number of regions, so it must be chosen carefully. Also, the algorithm is more complicated and more difficult to tune that recursive region growth.

<u>Region merging</u>

Desc: Begin with every pixel (or square groups of 2x2 or 4x4 pixels) as its own region. Then merge based on homogeneity until no more merges can happen.
Pros: Every spot in the image equally gets a chance to be the center of a larger region, unlike in seeding methods. Information between the regions like edges can be used as the 'melting' criterion.
Cons: Again, homogeneity is difficult to define. Also, by not seeding, the assumption is made that we know nothing about the image or probable locations of objects within it, which is often not true, and a loss of information.

Region splitting
Desc: Begin with the entire image as one region that does not meet the homogeneity criteria. Recursively split it into smaller regions until every region meets the homogeneity criteria.
Pros: Region splitting is the exact opposite of Region merging, but does not give the same answer even if the same homogeneity criteria are used. So it can solve some problem better than merging.
Cons: Merging can also solve some problems better than splitting, like very evenly distributed, but clearly segmentable images, like a chessboard. Even though each square is very different from those around it, the image as a whole fits homogeneity criteria quite well, causing splitting to fail, but merging to work.

Split/Merge
Desc: Uses a combination of splitting and merging to gain the benefits of both. Utilizes a pyramid representation of the image where the regions are square, and each level of the pyramid having different size square divisions of the image (usually each region is further split into fours). Based on homogeneity regions are either split into more regions at a lower level of the pyramid, or merged in the next higher level of the pyramid. Eventually a stable solution will be found. There are many variations on this method. For examples go to:
http://www.icaen.uiowa.edu/~dip/LECTURE/Segmentation3.html
Pros: Gains the benefits of splitting and merging.
Cons: Makes the assumption of square (sub)regions.

Class: Preprocessing
Type: Binary algorithms
Goals: These binary operations should help to merge separated pieces of solid objects, separate merged pieces of separate objects, fill in holes in solid objects, and get rid of noise in empty space. To summarize, these should help to begin partially identifying objects in the image and separating them from background.

Dilation
Desc: Center a structuring element on each pixel. If any of the pixels contained within the structuring element are on, then turn them all on.
Pros: Grows regions, merges regions, and fills in holes.
Cons: Can incorrectly merge 2 unconnected regions.

Erosion
Desc: Center a structuring element on each pixel. If all the pixels in the structuring element are on, then leave the center pixel on. Otherwise, turn it off.
Pros: Shrinks regions and separates wrongly connected regions.
Cons: Can enlarge holes and incorrectly separate regions.

Opening
Desc: Erosion followed by dilation with the same structuring element.
Pros: Removes filled in regions too small to contain structuring element.
Cons: Very sensitive to size and shape of structuring element.

Closing
Desc: Dilation followed by erosion with the same structuring operator.
Pros: Fills in holes smaller than the structuring operator.
Cons: Very sensitive to size and shape of structuring element.

Distance transform
Desc: Make each pixel value this distance (defined by some distance metric) to the nearest non-zero pixel.
Pros: Helps separate overlapping rocks and make object centers more clear.
Cons: While centers become more apparent, edges become much less pronounced.

Thresholding
Desc: By using a function of the median value, dividing the modes, or adaptive tiling methods (or a number of other ways), choose a threshold to make everything above it a ‘1’ and everything below it a ’0’.
Pros: Can get rid of background noise and isolate objects.
Cons: Very sensitive to method of choosing the threshold if you want it to generalize well. Often create artificial holes in objects, or makes objects too large. Often needs to be followed by opening, closing, or blurring operators to fix this.

Size Filter
Desc: Get rid of all connected blobs below a certain number of pixels.

Pros: Destroys mid size noise that erosion or opening can have difficulty getting rid of.
Cons: Sometimes throws out small rocks that we may wish to find. Very sensitive to pixel number threshold per image – needs a good adaptive way to choose it.

Class: Preprocessing
Type: Contrast enhancement, lighting, and intensity correction
Goals: Enhance existing features like edges while bringing out previously hidden features in the image. Suppress noise if possible. Correct for uneven lighting and poorly distributed intensity values.

Stretching
Desc: Map original intensity values onto new values so that some percentage (1% is the default in Matlab) of the data is saturated at the high and low ends of the intensity spectrum. Basically 'stretches' intensity values from 0 to 255.
Pros: Good with images that have a small range of intensity values.
Cons: Effect almost unnoticeable when used on an image that spans most of the 0 to 255 intensity space already, no matter how unevenly.

Histogram Equalization
Desc: Transform the original intensity values so they match some predetermined histogram like a uniform or normal distribution.
Pros: Has a large enhancement effect on most images, since images are not often evenly distributed according to some rule. Can effect images that stretching cannot. Also causes many previously hidden features to come out if the right distribution is chosen.
Cons: Tends to over-saturate many areas of the image, which could cause information and detail loss, and poor edge detection.

Adaptive Histogram Equalization
Desc: Histogram equalization on sub-sections of the image. Can also include more complex features like limits on the amount of contrast enhancement.
Pros: Prevents over-saturation and limits prevent amplification of noise. Optimal per 'object' in the image instead of over the entire image.
Cons: Tile size and enhancement parameters must be chosen carefully or the algorithm will perform poorly. This varies by image, making generalization difficult.

Plane Fitting
Desc: Fit a plane of best fit (or a low-order polynomial) to the image, and then subtract its value at each point from the intensity at that point.
Pros: Corrects for an artificial gradient in the lighting. Brings out edges that may have been undetectable otherwise.
Cons: A plane is not always the best approximation, or the lighting may already be even, causing loss of information instead of enhancement. Can lead to low image contrast, so may need to be followed by contrast enhancement.

Top Hat / Bottom Hat Transform
Desc: First take the erosion of an image using a structuring element to flatten the peaks. Then dilate to reconstruct the scale of the image, but with the softer peaks. Subtract this from the original image to get peaks only where local maxima (or minima for bottom hat) existed.
Pros: Enhances separability of nearby, disjoint maxima (or minima for bottom hat)

previously separated by poor contrast. Fixes uneven lighting conditions since it is a local operator.
Cons: Structuring element must be larger than the peaks, but smaller than the mountains, which can be difficult for complex scenes with objects of many sizes. Can also leave artifacts in the shape of the structuring operator, since it is a local operator.

Original + Top Hat – Bottom Hat
Desc: Add the top hat and subtract the bottom hat from the original image to enhance contrast.
Pros: Makes the bright brighter and the dark darker. Good for separating foregrown objects from background, and for finding object of a certain size.
Cons: Can cause over-saturation and is very sensitive, as above to the size of the structuring element.

Max(Top Hat , Bottom Hat)
Desc: Make an image that is the pixel-wise max of the top hat and bottom hat images.
Pros: Can be good for the specific application of rock detection, because it makes both bright rocks and dark rocks (compared to the background) maxima in the new image so that both can be detected.
Cons: Still very sensitive to structuring element size, should probably be made adaptive in some way.

Class: Preprocessing
Type: Filtering and smoothing methods
Goals: Reduce noise, prevent future over-segmentation, and merge loosely connected, but joined regions.

Mean Filter
Desc: For each pixel, make its new intensity value the mean of the values of all pixels in a local neighborhood.
Pros: Simple. Smooths out noise.
Cons: Very simplistic operator that weights close pixels as strongly as ones further away. Smooths over edges severely.

Median Filter
Desc: For each pixel, make its new intensity value the median of the values in its local neighborhood.
Pros: Gets rid of noise while retaining image edges and details since it does not rely on values that are significantly different from the average neighborhood value. Nonlinear, yet still very simple to implement and compute.
Cons: Only takes intensity as a parameter, not distance. More complex algorithms that take both into consideration may perform better.

Gaussian Filter
Desc: Use a 3d Gaussian to give each pixel a new value by weighting local pixel intensities by distance via the Gaussian function.
Pros: Smooths locally in an adaptive manner based on sigma and size of the Gaussian. Gets rid of noise and unevenness, leaving larger objects intact. Gaussian function fades with distance in a better way than a mean filter.
Cons: Smooths edges and surfaces equally, making later edge detection difficult. Very sensitive to sigma and size values.

Anisotropic Filter
Desc: Uses physical heat diffusion equation as additional error constraint along with Gaussian distance to blur noise while keeping edges intact where the gradient is high.
Pros: Keeps edges intact while smoothing over noise.
Cons: Must be implemented either iteratively or recursively, both of which are slow. Also has the problem of getting confused by a crack in the middle of a rock and keeping that internal edge.

Bilateral Filter
Desc: (from Tomasi and Manduchi, 1998) Bilateral filtering smooths images while preserving edges, by means of a nonlinear combination of nearby image values. The method is non-iterative, local, and simple. It combines gray levels or colors based on both their geometric closeness and their photometric similarity, and prefers near values to distant values in both domain and range.
Pros: Can smooth over internal edges and noise in rocks while preserving their strong, outer edges since similarity, as well as distance is used as part of the local weighting

metric. Also, being non-iterative is good. Helps reduce over-segmentation in algorithms like the watershed transform, and helps objects splitting in other algorithms.
Cons: Strong cracks in rocks can separate the rock into 2 incorrectly. This can possibly be overcome with multi scale bilateral filtering and hierarchical analysis of processed images (see Mkwelo, De Jager and Nicolls, Watershed based Segmentation of Rock Scenes…).

Class: Preprocessing
Type: Minima detection, enhancement, and processing
Goals: These algorithms make the assumption that appropriate preprocessing has been done so that all rocks of interest are some sort of minima, not both minima and maxima. These algorithms should detect regional minima, smooth out false minima, enhance minima, and smooth out noise in non-minima regions, so that later algorithms relying on minima can be more effective.

Regional minima detection
Desc: Finds all regional (local) minima.
Pros: Can help identify good starting places to look for rocks.
Cons: Most of these are false local minima caused by noise in the image, or natural features that are not interesting. Some minima may also be very shallow.

Extended minima detection
Desc: Finds all regional (local) minima that are some threshold lower than their neighbors.
Pros: Eliminates shallow minima.
Cons: Hard to find a good constant to use for the threshold.

H-minima transform
Desc: First, it finds all regional minima. Then, it eliminates all minima other than the identified extended minima by making them like their neighbors.
Pros: Unlike the above algorithms, it affects the image instead of just finding minima. Smooths shallow minima.
Cons: Also raises the value of the extended minima, making them stand out less.

Morphological reconstruction
Desc: Requires a marker image created by hand or by feature detection. This should be some representation of interesting minima. Uses repeated dilations until the contour of the marker image fits completely under the original image.
Pros: Flattens out peaks in image (the opposite can be done for minima as well).
Cons: Also dilates the rest of the image.

Imposing minima
Desc: First, makes a marker image using extended minima detection. Then, sets all these minima to 0 in the original image to emphasize them. Finally, eliminates all other regional minima like the h-minima transform.
Pros: Complex algorithm that both emphasizes minima and gets rid of shallow minima.
Cons: Equalizes all extended minima that make the depth threshold, so that they are all ‘0’ and not different degree of depth. This is not always bad, but can result in a loss of information.

Class: Preprocessing
Type: Object information algorithms
Goals: These algorithms should provide information about objects in the image. This includes pixels that are definitely part of objects, part of the background, or even rough region boundaries. This could also include information for a later algorithm to use, such as marker points for a watershed transform.

Foreground pixel detection
Desc: Use various algorithms like thresholding and erosion to find pixels that are definitely contained within foreground objects in the image. It is crucial that none of these pixels are on an edge or part of the background.
Pros: Usually used for 'marking' objects for algorithms like the watershed transform that need starting points, in this case to start 'flooding' from.
Cons: Only useful for a small class of algorithms, otherwise just loses image information or takes up more space.

Background pixel detection
Desc: Use various means like SKIZ, watershed transforms, or thresholding to find pixels that are definitely part of the background and not part of any foreground object.
Pros: Like foreground detection, helps algorithms like the watershed transform.
Cons: Again, only useful for a small class of algorithms.

SKIZ
Desc: Stands for 'Skeleton Influence Zones'. Creates rough region boundaries by placing dividing lines at the boundaries of the 'influence zones' of minima, the locations where a pixel is equidistant from 2 or more minima.
Pros: Can nicely partition an image into regions for texture or object analysis.
Cons: Minima must already be well defined by other preprocessing steps. Can often result in severe over-segmentation or large, meaningless regions.

Shadow removal
Desc: Make intensity space comparisons with hue space to detect and remove shadows in the image.
Pros: Removes shadows that could be mistaken as an object or that would occlude an object that we want to detect. Also can use shadow information for geometric information.
Cons: It is possible that there will not always be enough color information to make this sort of detection.

Class: Preprocessing
Type: Automatic Thresholding Algorithms
Goals: These algorithms should find an optimal threshold at which to make everything below it a '0'. This can be a globally optimal or locally optimal threshold, and should be able to generalize well to many images. It should be assumed that some preprocessing has already been done to make all the rocks in the image either all darker or all lighter than the background so a threshold will treat them all the same as foreground objects.

P-Tile
Desc: If you know the approximate percentage, p, of the area of an image that foreground object(s) will take up, threshold where p% of the histogram of pixel intensity will still be shown.
Pros: Very effective if you have a good area estimation and the objects have a uniformly different intensity from the background.
Cons: It is rare that these assumptions and estimations can be made, and are unrealistic for rock detection where number and size of foreground objects vary greatly from scene to scene.

Median Threshold
Desc: Set the threshold of the image or sub-images as a function of the median value within that image or sub-image.
Pros: Simple heuristic that can work well for simple images.
Cons: Natural scenes are very complicated and it is hard to find a function, linear or non-linear, to model this complexity correctly.

Mode Splitting
Desc: If foreground objects have some intensity value and background objects have another, both permeated with Gaussian noise, then the histogram will look bimodal with two Gaussian distributions. The goal of this algorithm is to set the threshold at the minimal point in the valley between these two distributions. This can be generalized to n modes.
Pros: If the assumptions hold, the threshold will be very accurate.
Cons: A ton of preprocessing would need to be done to get a natural scene to look like a series of Gaussians. It may even be impossible. Many local minimal would likely cause vast over-segmentation. Some polynomial fit for the histogram may be able to fix the problem is the degree of the polynomial could be approximated.

Iterative Selection
Desc: Start with an approximate threshold like the mean of all pixels. Then calculate the means of the two classes the threshold creates and put the new threshold directly in between them. Continue this process until the means stop changing. This is only one example – other criterion can be used to reassign the threshold and stop the iteration.
Pros: Accurate if you can make good assumptions about what characteristic ought to split up the data.
Cons: Very sensitive to the metric that is chosen to separate the data and the quality of the previous preprocessing to make that characteristic effective.

Adaptive Sub-image Thresholding

Desc: Use any of the other listed algorithms, but on sub-images instead of the whole image.
Pros: Gives more locally optimal results.
Cons: Can cause sharp discontinuities between adjacent sub-images if a threshold is picked poorly.

Plane/Polynomial Fit Threshold

Desc: Fit a plane or low order polynomial of best fit to the data. Use it as a threshold – only the things above it get to stay.
Pros: Good for getting rid of artificial intensity gradients in the image and other undesirable features.
Cons: It is hard to decide whether a plane or fixed order polynomial can generalize to many images.

Double Thresholding

Desc: Pick 2 thresholds x and y and use these thresholds to divide all pixels into classes 1, 2, and 3. Everything in class 1 will stay in the image. Everything in class 3 will be thrown out. Use a decision algorithm to decide which pixels of class 2 should stay and which should go. For example: If a pixel in class 2 has any neighbors in class 3, then put it in class 3. Repeat until no pixels change class and put the rest in class 1.
Pros: This can similar to iterative region growing or a number of algorithms, depending on the class 2 decision process. Solves the problem of having some pixels that are definitely foreground pixels, some that are definitely background pixels, and some that are in between, but don't all belong to the same class.
Cons: It can be difficult to create a robust class 2 decision process.