i-SAIRAS-01, Montreal, Canada, June 2001

Automated Detection of Craters and Other Geological Features

M.C. Burl[1], T. Stough[1], W. Colwell[2], E.B. Bierhaus[2], W.J. Merline[2], C. Chapman [2]

[1]Jet Propulsion Laboratory,
California Institute of Technology
MS 126-347, 4800 Oak Grove Drive,
Pasadena, CA 91109
{burl, stough}@aig.jpl.nasa.gov

[2]Southwest Research Institute
1050 Walnut Ave, Suite #426,
Boulder, CO 80302-5143
{wcolwell, bierhaus, merline, cchapman}@boulder.swri.edu

Keywords: object recognition, data understanding, craters, continuously scalable, matched filter, multi-scale, SVD, interpolating functions, basis functions.

Abstract

We report on our effort to develop algorithms to detect, size, and classify geological features in planetary and asteroidal data sets. Our current work has focused primarily on the detection of craters, which are of scientific interest because they are one of the most ubiquitous landforms in the solar system and they enable the relative ages of surfaces or surface units to be determined. The algorithm uses examples provided by a scientist to generate an efficient model for detecting the target object across a continuous range of scales. Tests on regions of the Lunar Maria (provided by Clementine) show that the algorithm achieves an 80% probability of detection for craters larger than four pixels in diameter with a 12% false alarm rate. The algorithm has also shown promise for detecting blocks and boulders in the NEAR imagery of asteroid 433 Eros and lava cones in X-SAR imagery of Earth.

1 Introduction

We report on our effort to develop algorithms to detect, size, and classify geological features in planetary and asteroidal data sets. The ever-growing volume of data being returned by NASA spacecraft has made the development of techniques for automatically interpreting the content of image data essential. Furthermore, it has been realized that deploying data understanding algorithms onboard a spacecraft will create new scientific opportunities that could not be achieved with existing technology.

Our current efforts have focused primarily on the detection of craters. This application is particularly worthy because craters are one of the most ubiquitous features in the solar system and are found on essentially all solid-surface solar system bodies. In the absence of direct sample returns from the surface of a body, craters are the primary means of establishing the age of the surface or surface units. The density of craters can yield the relative ages of these surface units and potentially can yield the relative ages between the surfaces of different bodies. Craters are therefore a fundamental tool for the determination of the geological age and history of a surface. An automated algorithm will help the scientist by eliminating the laborious task of manually labeling craters, and it will also provide for an objective, examinable, and repeatable process for performing the analysis. In contrast, most previous studies of planetary features, and in particular studies of crater statistics and morphology, have been done either manually (visual inspection and marking, sometimes simply with a pencil), or with semi-automated techniques that relieve the user from making measurements with a ruler or tabulating statistics, but still require manual selection and marking of the features on a display.

There are many data sets now sitting in archives for which the cratering record has never been fully examined. Reanalysis of more than a small fraction of these archival data would be difficult; however, as one exception, there is an attempt underway to analyze the Mars Viking dataset using a large, distributed set of human volunteers, known as Clickworkers [7]. Although interesting, such manual approaches do not provide a long-term, reusable solution as each new dataset or feature type requires a reinvestment of the same human effort. Also, improved acquisition, storage, and downlink capabilities will continue to generate ever larger data sets. A further limitation of the manual approach is that it will not enable onboard opportunistic science based decisions.

In contrast, we are using ideas from machine learning and computer vision to develop algorithms that can be trained or customized by scientists to look for particular types of features. This concept traces its roots back to the JARtool project [1], which was an attempt to develop a trainable tool for automatically cataloging small volcanoes in the Magellan radar imagery of Venus. However, unlike the earlier JARtool algorithm, the current algorithm can smoothly detect features across several orders of magnitude in scale. Rather than using domain-specific knowledge (e.g., craters have circular or elliptical boundaries [8, 3]), our algorithm has no specific built-in knowledge about craters. Instead, the algorithm trains itself for the crater-detection problem from a set of examples that are provided by the scientist. The algorithm uses these training examples to derive a crater model that can be used to detect and size craters in other images. To handle scale in an efficient way without requiring the user to supply an excessive number of training examples, the algorithm synthesizes virtual examples by resampling the user-provided example(s) at different pixel spacings.

The resulting continuously scalable detectors (CSDs) provide a principled extension of the matched filtering paradigm, which was developed in the early 1940's for radar and communications problems [14]. This new algorithm can be viewed as *matched filtering over a continuous scale space.* The advantages of matched filtering (or template matching as it is more commonly known in the computer vision and pattern recognition literature) are well known: the procedure is optimal for detecting a known signal in an additive white noise background and often works well even when these conditions do not completely hold. CSDs are closely related to the steerable-scalable-deformable filtering ideas developed by Freeman and Adelson [5], Perona [15], and others, as well as to the parametric feature detection work of Nayar, Baker, and Murase [12]. An important difference, however, is that CSD's are typically constructed from a real example of the object of interest rather than from a closed-form, analytical expression of the object's brightness profile. Hence, a user can interactively select an example of the target object from an image, or even draw a picture of the object using a palette of gray values, and then construct a CSD that can be used to find novel instances of the object over a wide range of scales. For object classes that are not well-represented by a single scalable template, unions of CSD's may be used.

2 CSD Construction

2.1 Prototemplate

Construction of a CSD begins with a prototemplate. Unlike the work of Nayar et al [12, 13] and Perona [15], we do not require an analytical expression for the pixel intensities of the object (or filter) as a function of the spatial coordinates (x and y) and scale parameter (σ). Instead we use a "real example" of the target object as the prototemplate. Of course, if a sufficiently accurate analytical expression were available, it would provide a viable means for generating the prototemplate. However, this is not the usual case except when dealing with very simple *parametric features* such as the step edges, lines, and corners treated in [12]. Hence, we typically generate the prototemplate by interactively selecting and extracting an image region containing an instance of the target object. A somewhat more refined version of this approach is to extract multiple instances, resize to a canonical size, and average to produce a smoothed prototemplate with higher signal-to-noise ratio. It is also possible to generate a prototemplate by hand-painting on a canvas with a computer mouse and a palette of gray values. This technique is necessary whenever there is not an instance of the target object available (for example, when formulating a query to an unfamiliar image database).

2.2 Scaled family

In the next step, a template *family* is generated by resampling the prototemplate at a number of different scales. For each scale, the number of pixels in x and y will be the same as in the prototemplate, however, the physical distance between the pixel centers will be different according to the scale. To avoid confusion in terminology, we will define the *scale* of an example as the ratio of pixel spacing in the prototemplate to pixel spacing in the example. Hence, if the pixel spacing in the prototemplate is 10 m between pixel centers and the object spans 7 pixels, but in a resampled version the pixel spacing is 5 m and the object spans 14 pixels, we will say the resampled version is at scale=2. (In other words, if the object looks twice as big in pixel units, it is at scale 2.)

In our experiments to date, we have generated the scaled family using one hundred linearly spaced scales in the range from 1 to 2, i.e., a family member is synthesized at scale 1.01, 1.02, 1.03, $\ldots$, 2. We have used the general image warping procedure described in [6] to perform the resampling, but alternative implementations should work fine. Figure 1 shows a prototemplate for a bowl-floored crater extracted from an image of Mars and the corresponding scaled family. (The prototemplate is the example in the upper left corner.)

2.3 Compression of the Family

Ideally we would like to use each template in the scaled family as a matched filter to locate objects of a

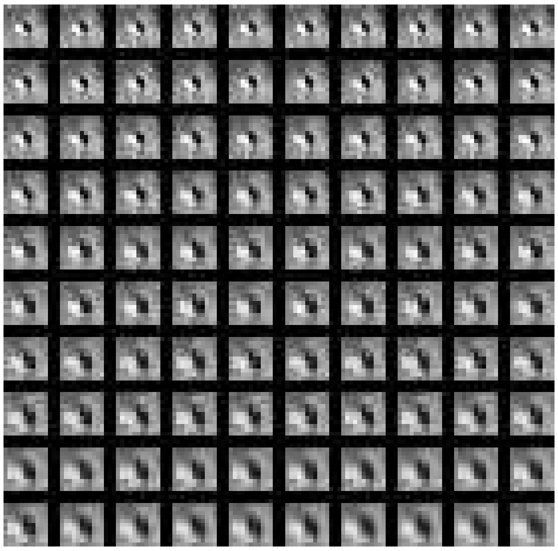

Figure 1: The upper left example shows a crater prototemplate selected from an image of Mars. The remaining examples show the family of scaled templates constructed by resampling the prototemplate at finer pixel spacings.

particular size; however, this is computationally infeasible since the number of convolutions would be too large. Hence, we use singular value decomposition (SVD) to compress the scaled family. This technique has become commonplace in vision applications [18, 15, 1, 11, 10] and is known under various other names, such as eigenfaces, principal components analysis (PCA), Karhunen-Loeve. Each template in the scaled family is normalized for DC value and contrast (more on this in Section 3.1), reshaped as a column vector, and stored as a column in a matrix $\mathbf{X}$. The columns are organized so that the first one corresponds to the smallest scale and the last one corresponds to the largest scale. The number of rows in $\mathbf{X}$ is just the number of pixels per template and the number of columns is the number of scale samples included in the family. The SVD decomposes $\mathbf{X}$ as the product of three matrices:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{1}$$

where the columns of $\mathbf{U}$ are orthonormal, $\mathbf{S}$ is diagonal, and the columns of $\mathbf{V}$ are also orthonormal. Equation 1 says that any column of $\mathbf{X}$ can be exactly expressed as a linear combination of the columns of $\mathbf{U}$. The j-th column of $\mathbf{S}\mathbf{V}^T$ provides the set of coefficients with which to weight the columns of $\mathbf{U}$ to generate the j-th column of $\mathbf{X}$.

In many vision problems it turns out that the columns of $\mathbf{X}$ can be well-approximated as a linear

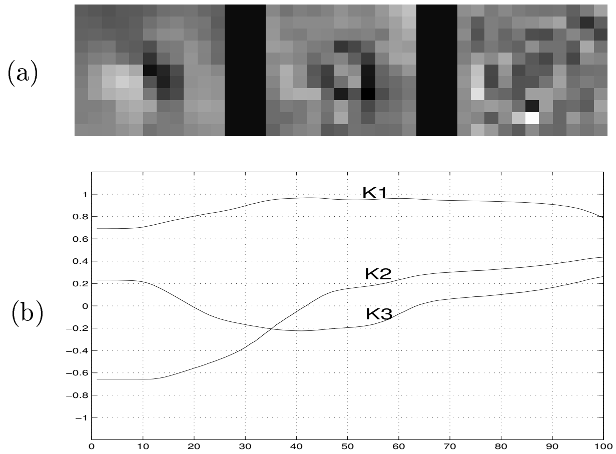

Figure 2: (a) First three basis functions obtained through SVD of the template family shown in Figure 1. (b) Corresponding interpolating functions as a function of a continuous scale parameter. On the x-axis, 0 corresponds to scale $= 1$ and 100 coresponds to scale $= 2$. The y-axis is unitless.

combination of the first k columns of $\mathbf{U}$, where k is much smaller than the total number of columns. The accuracy of this approximation is determined by the decay in the singular values (the elements on the diagonal of $\mathbf{S}$). For the template family shown earlier, we have found that three basis functions provide a sufficiently accurate approximation. These basis functions (first three columns of $\mathbf{U}$ reshaped as chips) are shown in Figure 2a.

In addition to the basis functions, the SVD produces a set of *interpolating functions*. If we define $\mathbf{K}_i(\cdot)$ as the i-th row of $\mathbf{S}\mathbf{V}^T$, then we can write:

$$\hat{\mathbf{X}}_j = \sum_{i=1}^{k} \mathbf{K}_i(j)\mathbf{U}_i \tag{2}$$

This expression gives the best approximation to template family member j in the subspace spanned by the $\mathbf{U}_i$'s ($i = 1, \ldots, k$). By passing from the discretely sampled $K_i(\cdot)$ functions to a continuous version $K_i(\sigma)$, we can generate members of the template family at an arbitrary scale. (Note: the accuracy will be best if σ is such that we are interpolating between template family members rather than extrapolating outside the range of known family members.) Figure 2b shows the first three interpolating functions.

3 Application of the Model

3.1 Correlation vs Scale

Suppose that we have an incoming image $\mathbf{I}$ and want to compute the normalized correlation between each patch of the image and the j-th member of the template family, which corresponds to some scale σ_j. Mathematically, we can express the correlation image $\mathbf{R}$ as follows:

$$\mathbf{R}_{\sigma_j} = \sum_{i=1}^{k} \mathbf{K}_i(\sigma_j) \cdot \mathbf{Q}_i \tag{3}$$

where $\mathbf{Q}_i$ is defined as the correlation between the i-th basis function and the image. In words, Equation 3 says that the correlation image generated by a specific template family member is a linear combination of the correlation images produced by the basis functions. More broadly, the correlation with *any* template family member is a linear combination of the correlation images produced by the basis functions; the interpolating functions $\mathbf{K}_i(\sigma)$ provide the proper weightings. This result is not new (see for example [15]), however, we have included it for completeness.

To compute the normalized correlation between two patches $\mathbf{Y}$ and $\mathbf{Z}$, the mean of the patches must be removed and the energy (or contrast or variance) of each patch must be normalized. For convenience of notation we can think of reshaping the patches as vectors $\mathbf{y}$ and $\mathbf{z}$. The normalized correlation ρ is then given by:

$$\begin{aligned} \rho &= \left(\frac{\mathbf{y}-\mu_{\mathbf{y}}}{\sigma_{\mathbf{y}}\sqrt{n}}\right)^T \left(\frac{\mathbf{z}-\mu_{\mathbf{z}}}{\sigma_{\mathbf{z}}\sqrt{n}}\right) \\ &= \tilde{\mathbf{y}}^T\tilde{\mathbf{z}} \end{aligned} \tag{4}$$

where $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{z}}$ are normalized versions such that $\mathbf{1}^T\tilde{\mathbf{y}} = 0$, $\tilde{\mathbf{y}}^T\tilde{\mathbf{y}} = 1$, and similarly for $\tilde{\mathbf{z}}$.

The members of the template family undergo this normalization procedure before being placed in the columns of $\mathbf{X}$. The basis functions $\mathbf{U}_j$ that result from the SVD of $\mathbf{X}$ necessarily satisfy the zero-DC property (i.e., $\mathbf{1}^T\mathbf{U}_j = 0$), but to restore the unit energy condition, we must define a new set of interpolating functions as follows:

$$\tilde{\mathbf{K}}_i(\sigma) = \frac{\mathbf{K}_i(\sigma)}{\sqrt{\sum_{i=1}^{k}\mathbf{K}_i^2(\sigma)}} \tag{5}$$

The $\tilde{\mathbf{K}}$ interpolating functions are used in place of the original $\mathbf{K}$'s appearing in Equation 2.

From a computational perspective normalizing every patch of the image before computing the inner product with the basis functions (as suggested by Equation 4) is a bad idea. Instead, we apply the basis functions to the *un-normalized* image and later apply the appropriate correction. The "fix" is to divide the un-normalized correlations by the standard deviation of pixel values in each image patch. The standard deviation can be conveniently computed with separable kernels [17]. The separable kernel technique simply decomposes a 2-dimensional rectangular box filter as the outer product of two 1-dimensional filters (a row filter of all ones and a column filter of all ones). By convolving first with the row filter and then with the column filter, the necessary statistics for each image patch can be obtained with $O(4\sqrt{n})$ operations per pixel rather than the $O(2n)$ required with direct 2-d convolutions, where n is the number of pixels in the patch.

3.2 Maximum response versus scale

In the previous subsection we saw that the normalized correlation of an image and a template family member at *any* scale could be obtained (approximately) from the correlations of the basis functions with the image. At a particular spatial location (x, y), we are now interested to find the template family member that produces the largest response value (correlation) and the value of the maximum response.

$$\begin{aligned} R_{\max}(x,y) &\triangleq \max_{\sigma}\ R_\sigma(x,y) \\ &= \max_{\sigma}\ \sum_{i=1}^{k}\tilde{\mathbf{K}}_i(\sigma)\cdot\mathbf{Q}_i(x,y) \end{aligned} \tag{6}$$

At a particular spatial location the response profile versus scale is just a *linear combination of the interpolating functions.* It is possible to explicitly form the response profile versus scale and search for the maximum value (similar to the approach used by Nayar et al [12]). However, a slightly more efficient alternative is to approximate the interpolating functions as closed-form expressions of σ. Cubic polynomial approximations (or piecewise combinations of cubic polynomials) are especially useful since the zeros of the derivative of response with respect to scale can be obtained in closed form. In particular, the derivative is just a quadratic polynomial (or piecewise quadratic) in σ, so the roots, which correspond to extrema in the response, fall out as solutions of a quadratic equation. Thus, we simultaneously obtain the maximum response versus scale and the scale that yields the maximum.

3.3 Full Scalability

The SVD-based approach we have described so far is useful for representing a template over a continuous set of scales; however, we do not want to make the range of scales to be covered by this technique to be too large for two reasons: (1) the support of the basis functions is nominally as large as the largest

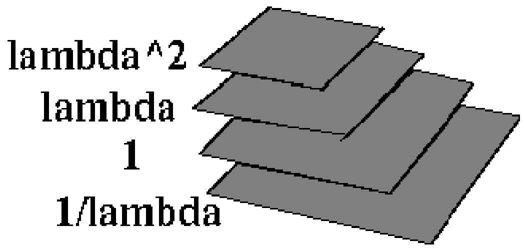

Figure 3: The CSD model is applied to a pyramid representation of an image to cover orders of magnitude in scale variation.

family member and (2) we do not expect that a low-dimensional linear basis will be adequate for spanning a large scale range (too many basis functions would be needed). Manduchi and Perona [9] considered this problem in the context of multi-orientation, multi-scale filtering. Their solution was to apply basis filters to different levels in a pyramidal decomposition. We use essentially the same technique. The SVD is used to span a smaller range of scales, typically a factor of two in our experiments; then we apply the resulting basis functions to a pyramid representation of the image. If the spacing between pyramid levels is properly matched to the scale-range spanned by the template family, then the end-product is a continuously scalable detector that smoothly covers orders of magnitude in scale variation.

Figure 3 illustrates an image pyramid. The basic image is at scale 1. Using the resampling algorithm of [6] the "pyramid" can be constructed with both reduced size versions of the image (as in standard pyramids [2]) and interpolated versions that are larger than the original image. The interpolated pyramid levels are useful for detecting objects that are smaller than the prototemplate. For each spatial location where the maximum correlation value exceeds a threshold, a marker of the appropriate size is recorded. (The position and size are remapped to the scale 1 image coordinates).

3.4 Illumination Invariance

Variations in the appearance of an object due to illumination direction must be handled in addition to the variations due to scale. In the case of optical imaging of craters, we make the assumption that the illumination direction is recorded when the image is captured. The azimuth component of the sun angle can then be removed by rotating the image (or the model). Variations in the elevation component of the sun angle are handled through elevation angle specific models. For active imaging systems, such as SAR, the illumination and receiver are co-located so the problem of illumination invariance does not occur for rotationally symmetric objects.

3.5 Arbitration

Two factors can cause multiple hits (markers) to be generated for a single object. Objects whose scale is near the "seam" between two different pyramid levels may be marked twice (once at each level). Also, multiple markers may be generated at slightly shifted spatial positions. With classical matched fitlering, this problem is usually resolved by taking a local max over a small spatial window to find the peak of the correlation surface. For the continuously scalable case, more sophisticated processing is required. One view is as a competition over scale and spatial neighborhood to determine the highest correlation value and the (x, y, σ) coordinates at which the best correlation occurs. We implement this competition by grouping markers into clusters using a single linkage algorithm [4]. Once the markers are assigned to clusters, we simply choose the marker with the highest correlation value as the representative of the cluster. The metric between the markers can be a function of both the marker centers and their scales. In the single linkage algorithm, two markers are assigned to the same cluster if their distance is less than a threshold. This rule is applied transtitively so that if A is close to B and B is close to C, then A, B, and C are all assigned to the same cluster. In studies of cratering it is quite common for small craters to appear inside larger craters. Hence, the grouping metric must be defined so that a large crater and any nested smaller craters are not all grouped into the same cluster. For this case, we used the following "distance" measure:

$$d = 1 - \sqrt{\frac{\text{area}(A \cap B)}{\max(\text{area}(A), \text{area}(B))}} \tag{7}$$

which is based on the area of overlap between the cricles.

4 Experimental Results

In this section, we present experimental results on an application of CSD's to the detection of craters, as well as some initial experiments on the detection of other geological features.

4.1 Detection of Craters

For the crater application, a CSD was constucted from a crater example selected from an image of Mars. The CSD consists of three basis filters and the corresponding interpolating functions, which are approximated with cubic polynomials. (The basis functions and raw interpolating functions are shown in Figure 2.)

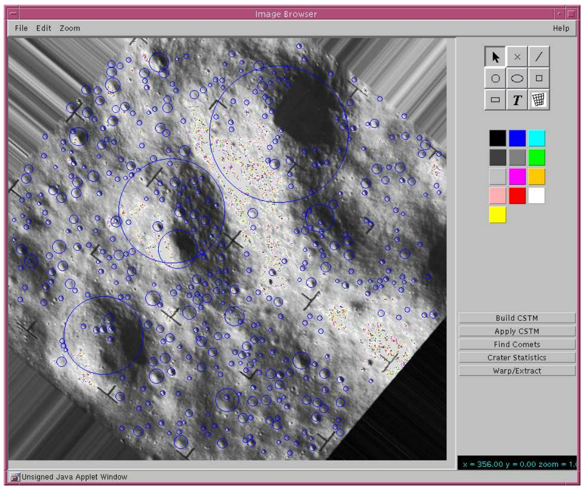

Figure 4: Detection of lunar craters in a Ranger 9 image. The diameter range spans more than an order of magnitude.

The model was then applied to a novel image of the Moon. "Raw" detections were clustered using the area of overlap relative to the area of the larger marker as the distance metric. The final result is shown in Figure 4 as a set of circles overlaid on the image. (The black +'s, T's, and L's are fiducial marks added by the camera to aid in calibration and can be ignored for our purposes.) Note that the craters in this image span more than an order of magnitude in scale and that the algorithm does indeed detect large and small craters as well as nested craters. Overall, the detection and sizing performance is quite good, however, it is probably below that of human experts.

Figure 5 shows the result of a more recent CSD model, which incorporates masking to focus solely on the area hypothesized to be inside the crater. The red circles are correct identifications made by the algorithm. The black circles show the corresponding human expert labels for the craters that were detected by the algorithm. The blue circles show true craters (identified by the human expert) that were missed by the algorithm. The orange circles are false alarms (regions identified by the algorithm that do not correspond with the human expert's labeling).

The linear grooved features that run across the image are responsbile for generating a large percentage of the false alarms. This result is not suprising and can be traced to the fact that the algorithm is looking within a small circular window and the shading of the groove within the mask looks very crater-like. Of course, by looking at the surrounding context, it is clear that these places are not true craters so this is one area for improvement. Specifically, we will feedback the results of the CSD to a supervised classification algorithm that can also look at the supporting context to reject false alarms such as these.

A second point to note is that many of the missed craters are at sizes less than four pixels in diameter. The crater detection model was generated with a minimum size of four pixels so it is not suprising that these smaller craters were missed. We also believe that subpixel alignment problems may be producing degraded results at the smaller sizes. (A half pixel shift of on object that is only four pixels across may significantly degrade the correlation.)

Figure 6 quantifies the performance of the CSD on the Clementine image of Figure 5. The curve, which is known as a receiver operating characteristic (ROC) curve, shows the trade-off between detection rate and false alarm rate as a function of an implicit threshold. At high threshold values, the false alarm rate is lower, but the detection rate is also lower. At lower threshold values, both the detection and false alarm rates increase. The ROC curve is computed by automatically matching the algorithm output against human ground truth. If an algorithm-generated circle and a human-generated circle overlap sufficiently (using the dissimilarity measure of Equation 7, a correct decision is recorded. If there is no ground truth circle that is sufficiently overlapped with a given algorithm-generated circle, a false alarm is recorded. The probability of detection is determined by dividing the number of correct decisions by the number of craters in the human ground truth. The false alarm rate is expressed relative to the total number of hits produced by the algorithm ($1 - \mathit{precision}$). At a threshold of 0.7, the algorithm successfully detects 80% of the craters, with a 12% false alarm rate. (Of all the things the algorithm claims to be craters, 12% are not.)

4.2 Detection of Other Features

An advantage of the learning-based approach that we have used to derive a detector for craters is that detectors for other types of geological features can be developed quickly by supplying a different set of training examples. For example, we have adapted (re-trained) the system to detect blocks and boulders on the surface of Eros, as well as to detect lava cones near the Lava Bed National Monument using imagery obtained with the X-SAR instrument. The latter experiment is in support of a New Millennium Program study phase activity known as the Autonomous Sciencecraft Constellation (ASC) [16]. ASC will use onboard planning and science processing capabilities to reduce downlink bandwidth and provide autonomous retargeting of features of interest.

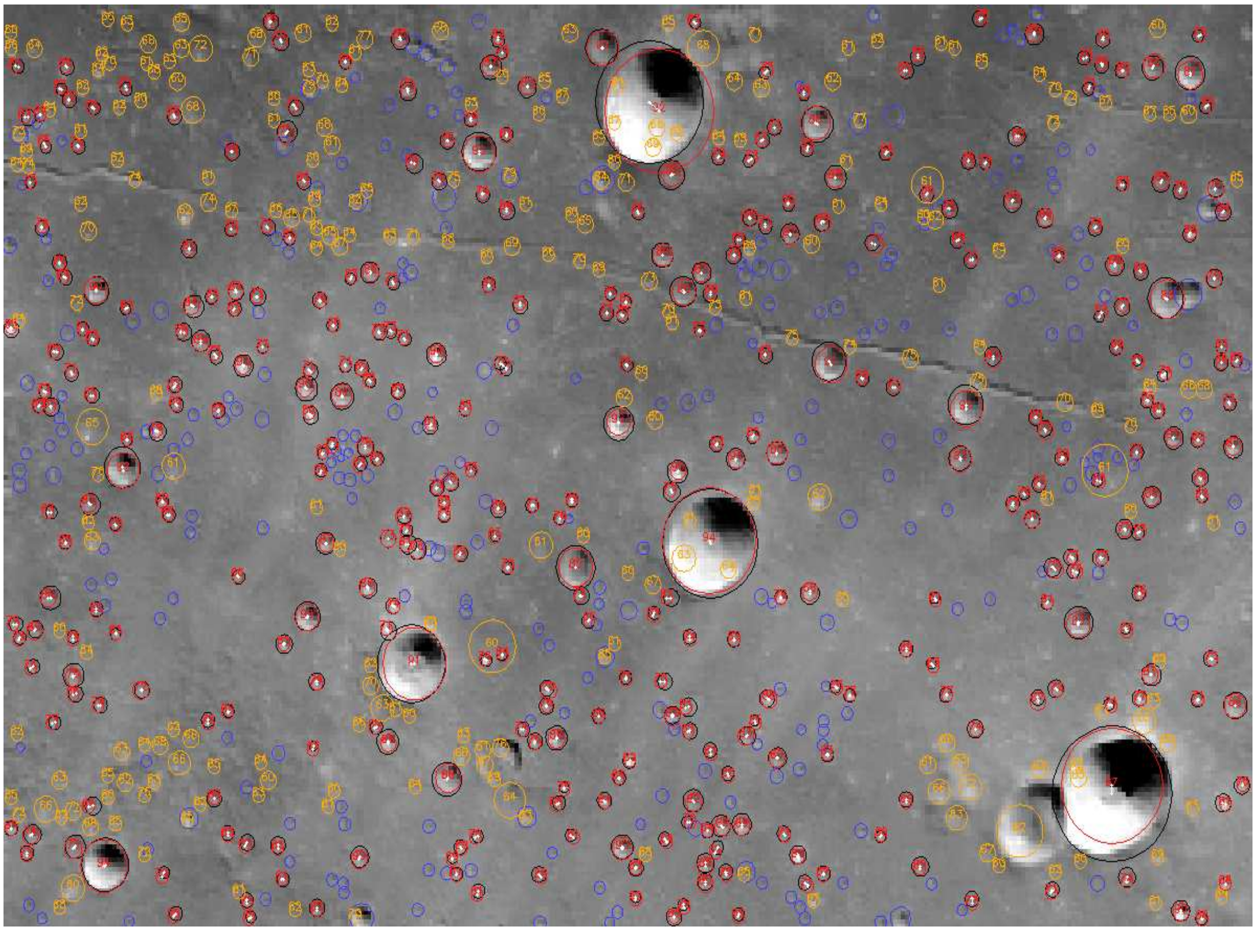

Figure 5: Detection of lunar craters (threshold = 0.6) on a Clementine image taken in the Lunar Maria. Red circles are correct identifications made by the algorithm. The black circles show the corresponding human expert labels for craters that were detected by the algorithm. Blue circles represent true craters that were missed by the algorithm and orange circles show false alarms.

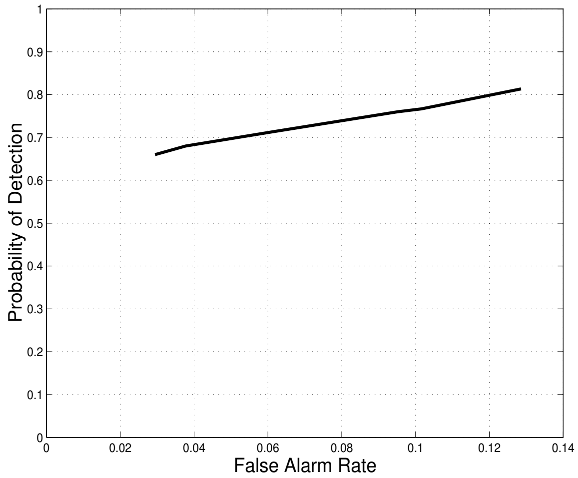

Figure 6: Probability of detection vs false alarm rate (craters $\geq$ 4 pixels) for Clementine image shown in Figure 5.

5 Status and Future Work

We are now approximately two-thirds through a 3-year development program supported by the NASA Office of Space Science Applied Information Systems Program. The goals by year were roughly: 1) development of the tool, 2) extensive reliability testing and refinement, and 3) applications to real scientific data sets and introduction to the scientific community. We have made significant progress toward these goals, but not without some difficulties. We have a working detection tool and have tested it on a number of sample surfaces and compared the results against human measurers. Trial data sets include Lunar Ranger and Clementine images, Galileo images of Europa and Callisto, and NEAR images of asteroid 433 Eros. Results on the Lunar Maria and Eros are quite good; however, the complicated background of Europa and the variability in appearance of craters on Callisto clearly cause difficulties, which we are working to resolve. We have also shown that the system can be adapted to (re-trained for) other types of features; in particular, we have demonstrated that the system can detect blocks and boulders on the surface of Eros and lava cones on Earth.

Directions for future algorithm development work include handling subpixel shifts, using supervised classification techniques to reject false alarms, and enahncing the CSD detectors to represent inherent variation between different instances of the same class.

Acknowledgements

This research has been carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1] M.C. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, L. Crumpler, and J. Aubele. Learning to recognize volcanoes on Venus. *Machine Learning*, 30:165–194, 1998.

[2] P.J. Burt and E.A. Adelson. "The Laplacian Pyramid as a Compact Image Code". *IEEE Trans. Commun.*, COM-31:532–540, 1983.

[3] A.M. Cross. Detection of circular features using the Hough transform. *Int. J. of Remote Sensing*, 9(9):1519–1528, 1988.

[4] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[5] W. Freeman and E Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[6] Paul S. Heckbert. Fundamentals of texture mapping and image warping. Master's thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, 1989.

[7] B. Kanefsky, N.G. Barlow, and V.C. Gulick. Can distributed volunteers accomplish massive data analysis tasks? In *Lunar and Planetary Science Conference*, 2001.

[8] B. Leroy, G. Medioni, A. Johnson, and L. Matthies. Crater detection for autonomous landing on asteroids. In *Proc. IEEE Workshop on Perception for Mobile Agents*, June 1999.

[9] R. Manduchi and P. Perona. Pyramidal implementation of deformable kernels. In *Proc. of the 2nd Int. Conf. on Image Processing*, 1995.

[10] B. Moghaddam and A. Pentland. *"Probabilistic Visual Learning for Object Representation"*, chapter 5, pages 99–130. Oxford University Press, 1996.

[11] Hiroshi Murase and Shree Nayar. "Visual Learning and Recognition of 3-D Objects from Appearance". *Int J. of Comp. Vis.*, 14:5–24, 1995.

[12] S. Nayar, S. Baker, and H. Murase. Parametric feature detection. In *Proceedings of the IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition*, pages 471–477, 1996.

[13] S. Nayar, H. Murase, and S. Nene. *Parametric Appearance Representations*, chapter 6, pages 131–160. Oxford University Press, 1996.

[14] North. title unavailable. Technical report, RCA, 1943.

[15] P. Perona. Deformable kernels for early vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(5):488–499, 1995. Previously appeared as MIT-LIDS TR 2039, 1991.

[16] R. Sherwood, S. Chien, M.C. Burl, R. Knight, G. Rabideau, B. Engelhardt, A. Davies, P. Zetocha, R. Wainwright, P. Kuplar, P. Cappelaere, D. Surka, B. Williams, R. Greeley, V. Baker, and J. Doan. The Techsat-21 Autonomous Sciencecraft Constellation demonstration. In *Int. Symp. on Artif. Intelligence and Robotics for Space (i-SAIRAS)*, June 2001.

[17] S. Treitel and J. Shanks. The design of multistage separable planar filters. *IEEE Trans. Geosci. Electron.*, GE-9(1):10–27, 1971.

[18] M. Turk and A. Pentland. "Eigenfaces for Recognition". *Journal of Cognitive Neuroscience*, 3(1), 1991.