Single Image Based Hazard Detection for a Planetary Lander

Max Bajracharya
Jet Propulsion Laboratory, California Institute of Technology

Abstract

Autonomously landing a spacecraft safely in a hazardous area on a planet is a critical mission goal. To help achieve this goal, we have developed a fast algorithm to detect hazards on a planetary surface using a single image from a descent spacecraft camera. This paper proposes a general approach of segmentation and classification for hazard detection and describes an implementation of this approach. The implemented algorithm uses local intensity clustering to segment the image into regions and then classifies these regions as hazards or part of the surface. We have tested our methodology using synthetic terrain images as well as real camera images.

1 Introduction

The ability to land a spacecraft autonomously and safely in a moderately hazardous area on a planet is becoming extremely important to the success of future planetary exploration missions. Scientifically interesting areas on the planet are generally areas that are hazardous to land in; therefore, the ability to land in moderately hazardous areas is a necessity. Selecting a safe landing site during the descent phase requires autonomy largely because the communications delay between the spacecraft and a potential pilot on Earth is too long to allow real-time control. A more complete motivation is provided by [7].

Autonomously landing a spacecraft safely on a planetary surface requires the ability to detect and avoid hazards on the surface. An optical camera is one of several different types of sensors which can be used to achieve this capability. While it does not explicitly provide range data like RADAR or LIDAR sensors, it has several distinct advantages over these other sensors. A camera can have a larger field of view, consumes less power, is lighter, has better resolution, and has fewer mechanical components subject to failure. However, it also suffers from some of the same problems as LIDAR and RADAR: dust in the atmosphere can occlude the surface, the data can be blurred by spacecraft motion, including high frequency vibration of the spacecraft, and the data can be distorted by optical and receiving characteristics. In addition, using a camera constrains the time of entry and descent to the "daytime," when the planet's surface is illuminated by sunlight. Furthermore, most autonomous navigation from image algorithms are, like most vision algorithms, computationally expensive and too slow for real-time spacecraft descent.

1.1 Goal

The goal of the proposed algorithm is to detect safe landing sites on the surface in near real-time (less than one second of processing time on flight hardware) using a single spacecraft mounted camera. The criteria for a safe site is a set of requirements such as an area with rocks no larger than a certain size and with a surface less than a certain slope. However, the requirements may also include constraints such as not landing inside a crater with no feasible path for a planetary rover traversal of a certain distance. A safe landing site is also characterized by the reachable area, which is an elliptical area defined by the spacecrafts descent profile, its amount of fuel, the atmospheric

conditions, etc. The output of the proposed algorithm is a safety map and a certainty map, which will then be integrated with the safety/certainty map derived from other sensors and the reachable area [8].

Rather than trying to duplicate the information which can more easily be obtained by active range sensors (RADAR/LIDAR), such as slope and roughness [9], we have attempted to extract hazardous features from the image, such as rocks and craters. We are also only processing a single camera image at a time to relax the reliance on more computationally expensive algorithms as well as possibly erroneous spacecraft motion knowledge.

1.2 Possible Approaches

Image-based safe landing on Mars was considered by the Mars Program as summarized in [7]. The approach proposed is to simply tile the image and calculate variance in each tile, assuming a low variance corresponds to a safe area. While this method is not very quantitative, it is extremely fast and simple. Another approach, proposed in [13] is to calculate slope by intelligently combining slope from IP tracking and gray level shape from shading information and to calculate roughness using an auto-correlation measure in a local window. Another multi-image approach is also proposed in [4] using the homography transform on certain areas in an image to calculate slope and, as a by-product, roughness.

Explicitly detecting features on a planetary surface has also been explored with several approaches, however most of these methods are designed to run offline, and can afford to be computationally expensive. Several different approaches are proposed in [2], [10], and [12]. Segmentation and texture methods for rover navigation are also useful approaches. Using shadows to detect rocks has been implemented in [6] and texture methods are proposed in [3].

2 Algorithm Description

Our approach to hazard detection is to quantitatively determine the size and abundance of hazards in an area on the surface. To accomplish this, our general approach is to segment the image and then classify the segmented regions as hazards or safe areas. While we have implemented a fast and therefore relatively simple version of this algorithm, the concept is not limited to this specific implementation. There are many alternate ways to segment the image and many ways to classify the regions.

In our algorithm, local areas of the image are used to segment the image, and the segmented regions are then classified as shadows, hazards, or surface areas. Hazards in the implemented case are limited to rocks; however, this representation can be extended to other surface features such as craters and cliffs. We also make use of approximate knowledge of the sun relative to the surface to detect and classify hazards.

2.1 Segmentation

Rather than using a more computationally expensive texture segmentation method such as edge flow [11] or Gabor filter based techniques [5], our algorithm implements a fast and simple intensity based segmentation method. This segmentation method can also be extended to include texture measures as well as more regions. Currently, to maintain a run-time of less than one second per image (400x400 pixels), only shadows and a single class of "hazards" are segmented from a surface which is assumed to be slowly changing in intensity. Also, only the pixel intensity is considered in the segmentation; however, the algorithm could include variance, entropy, or features extracted from other local texture measures which may come from frequency analysis (Gabor filters, etc.).

The segmentation algorithm is a variant of the k-means clustering algorithm, where pixels are clustered together based on their texture features. But rather than analyzing the entire image, the

clustering is performed on small, local areas in the image. This allows the background surface to change slowly, while still extracting outliers from the surface.

The k-means algorithm is an iterative technique (which is guaranteed to converge [1] but in an undetermined number of iterations) for unsupervised classification of data points into k classes. The algorithm selects the locally optimal k mean values in the space such that the distance of each data points to its nearest mean value is minimized using the following equation:

$$E = \sum_i \min_j \|p_i - \mu_j\|$$

where p_i are the image data points, μ_j is the j^{th} mean for class $j = 1..k$, and E is the error value to minimize with clusters.

Briefly, the general k-means procedure consists of the following steps:

- initialize k means (μ_j for $j = 1..k$).
- loop until ΔE is small:
  - assign class labels:

    $$l_{p_i} = \arg\min_j \|p_i - \mu_j\|$$

    where l_{p_i} is the class label for point p_i.
  - calculate k means:

    $$\mu_l = \frac{\sum_i^{N_l} p_i^l}{N_l}$$

    where μ_l is the new mean value for class l and N_l is the number of samples in class l ($l = 1..k$).

The k-means algorithm can be made extremely efficient in the simple, two-means case. Each classification iteration in the k-means algorithm requires $O(knm)$ or $O(nm \log(nm))$; but since $k = 2$ for all cases, the algorithm runs in $O(nm)$, with a small constant factor. Additionally, for small areas and small k, the k-means algorithm converges in very few iterations.

Segmenting the image using local areas rather than globally, while adding a small overhead to algorithm complexity, provides several advantages. It allows the assumption that only a few regions exist in a small area to hold, so that the k-means algorithm can be made substantially faster. It also allows the segmentation to be robust to a gradually changing surface: the difference in mean values of outliers and the surface will tend to be higher if a local area is used. The overhead incurred by using local areas is the overlap necessary to maintain consistency across the areas. For example, we use an overlap (Δx, Δy) of half of the area size (nxm) (which is based on the size of the largest hazard expected in the image, but not particularly sensitive to this decision) so that a hazard which may fall right on the border of one local area will fall in the center of the next area considered. This can obviously be made smaller and still allow for sufficient overlap. Using the overlapping areas, the run-time for a single iteration of the k-means algorithm is (amortized) $O(knm\frac{w}{\Delta x}\frac{h}{\Delta y})$, where the variables are given in Figure 1 (a).

This method is effective and can be made efficient because of the prior knowledge of the surface. But it relies on certain assumptions to work correctly. The main assumption of the segmentation method is that hazards in a local area have a similar intensity (or all have visible shadows) and the surface they are on is relatively uniform in intensity. While noise is eliminated by pre-filtering the image with a smoothing function, a highly textured surface will still cause a false segmentation.

2.2 Classification

After the image is segmented, separated regions are classified as either part of the surface, a shadow, or a hazard. Only the mean of the region is currently used to classify a region, however, other texture measures such as variance could also be used.

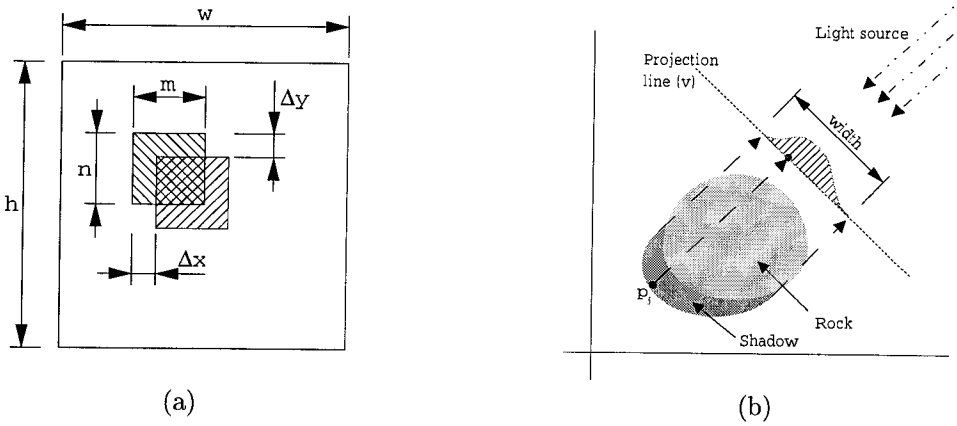


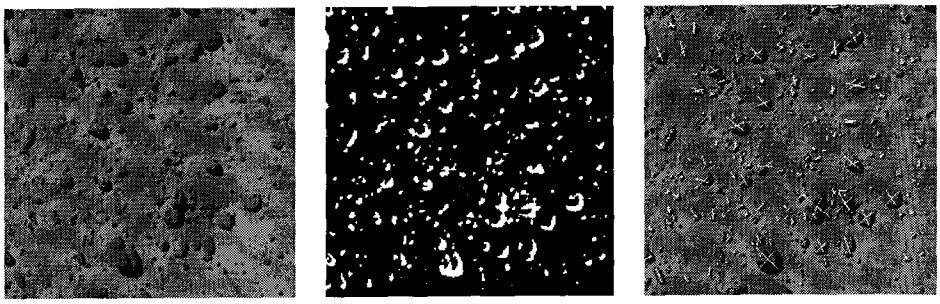
Figure 1: (a) Overlapping local regions for k-means segmentation, (b) Shadow projection to determine hazard width.

Regions classified as shadows are then used to determine the hazards that have caused them. The method assumes an approximate knowledge of the sun angle, which could be inferred from the spacecraft position and time of day or explicitly detected with a sun sensor. A detected shadow is projected onto a line perpendicular to the azimuth angle of the sun (measured from the surface) to estimate hazard size (Figure 1 (b)). The minimum and maximum points on the projection line give an estimate of one dimension of size, while the maximum or minimum distance to the projection line give an estimate to the other. While this is an overestimate, using the "thickness" of the shadow at the maximum/minimum point can give a better estimate, as well as providing the third dimension of size. The curvature of the shadow can also be used to differentiate concave and convex objects (rocks versus craters), although in our case, size is generally sufficient to disambiguate the expected hazards. The size and outline of the hazard is then approximated by a bounding box, defined by the minima and maxima of the dimensions. This conservative estimate can be improved by attempting to fit an ellipse to the detected dimensions, or by growing a region around an approximate center of the hazard (inferred from the detected shadow). However, these methods tend to increase the computational complexity of the algorithm, and, in this case, a conservative example was well suited to the application.

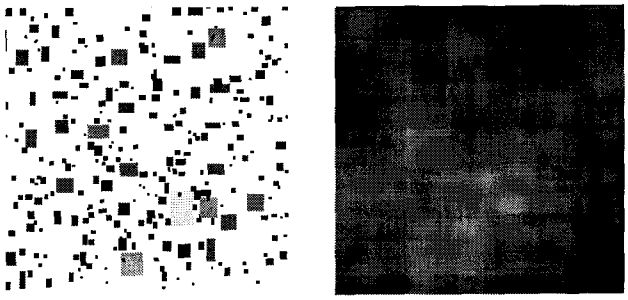
Hazards, like shadows, are classified based on a segmented region's mean value, which is significantly different from the surface mean intensity. For consistency in the size metric, the first size dimension of detected hazards are labeled along an axis perpendicular to the azimuth angle of the sun. Then the maximum or minimum distance from this dimension is used as the second dimension. Again for consistency, a bounding box on the minimum and maximum of these two lines then defines the rock size and outline. Determining a more precise size in this situation is easier because the region is already defined, and a simple pixel count could be used.

2.3 Hazard Map Generation

Once hazards are detected, a size and abundance map are generated for input into a rule-based system which combines the hazard maps of multiple sensors [8]. The size map is created from the bounding boxes of the hazards, where each value in the bounding box is the area of the bounding box. The abundance at any point is generated by calculating the number of unique hazards (defined by their bounding box) which fall within a window around that point. The abundance map is then created by finding the abundance at every point in the image (equivalent to a sliding window operation). This can be done efficiently by using the abundance at one point to calculate a neighboring point – for a window sliding along a row, only the next and last column need to be checked for



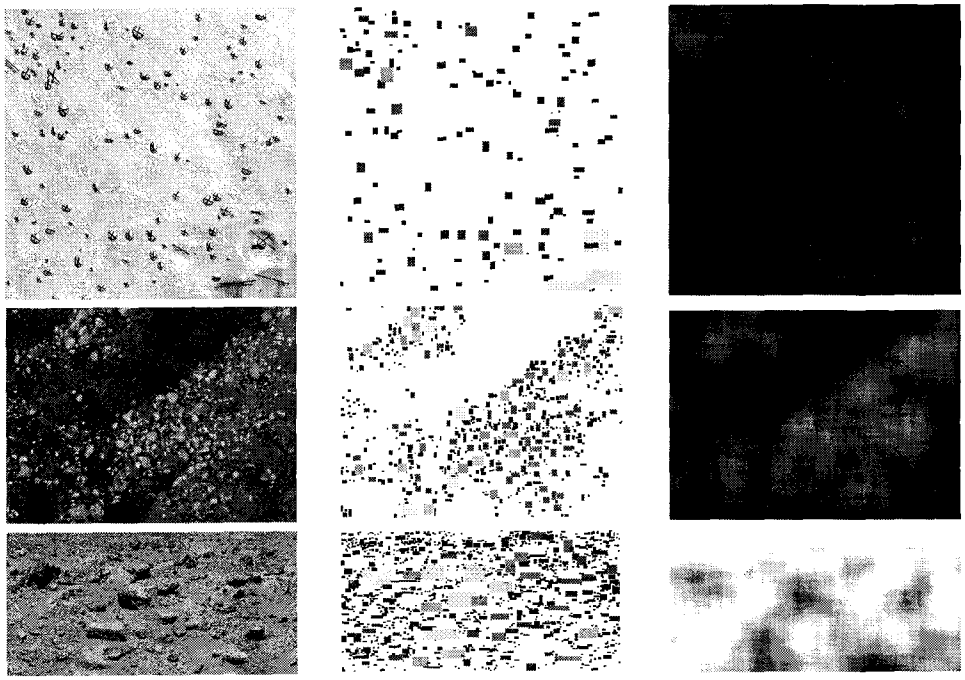
(a) original image (b) segmentation (c) rock detection



(d) size map
dark (small) - light (large)

(e) abundance map
dark (low) - light (high)

Figure 2: Example of (a) original image, (b) segmentation with local k-means, (c) rock detection from segmentation and shadow projection, and (d) the generated size map, and (e) abundance map



Synthetic:

Helicopter:

Pathfinder:

(a) images with detection

(b) size maps
dark (small) - light (large)

(c) abundance maps
dark (low) - light (high)

Figure 3: Hazard detection on a synthetic, helicopter, and Mars Pathfinder image

hazards entering or leaving the window.

3 Algorithm Testing

The implemented algorithm has been tested most thoroughly on synthetic terrain images from simulated descent, but has also been applied to helicopter descent images and Mars Pathfinder images. The algorithm works well on overhead synthetic terrain images (Figures 2 and 3) where rocks with and without shadows are present. Initial testing with helicopter images (Figure 3) on natural terrain with very few shadows is also very promising. Additionally, the algorithm is moderately effective on the few horizontal Mars Pathfinder images (Figure 3) tested. While the detection of rocks from shadows on horizontal images is less accurate, these images show that the contrast of rock and surface intensity can be used as a distinguishing factor for parts of the Martian terrain.

Eventually we would like to integrate the algorithm into a gantry/sandbox setup to simulate descent conditions and test the dependence on sun angle and position. We then intend to actually run the algorithm on a helicopter descent.

4 Future Work

The algorithm, while generally robust, can be extended in several ways, although subject to increased computational complexity. Segmentation and classification can both be improved by using additional texture measures (such as variance or frequency responses) or more clusters. Hazards can also be differentiated by considering the shape of the shadow or the projected shadow. Additionally, rather than relying on a known sun position, the angle could be estimated from shadow shapes.

5 Conclusion

In this paper, we describe a general approach for detecting hazards from a single planetary lander camera image as well as a specific implementation of this approach. We use local intensity clustering to segment the image and then classify segmented regions as shadows and hazards or part of the surface. We then detect hazard sizes based on shadow size and sun angle or region size, and use this information to generate maps corresponding to the size and abundance of hazards on the surface. The experimental tests show that our algorithm is sufficiently fast and robust to enable an effective, near real-time implementation on board a spacecraft.

6 Acknowledgments

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1] Leon Bottou and Yoshua Bengio. Convergence properties of the K-means algorithms. *Advances in Neural Information Processing Systems*, 7, 1995.

[2] M.C. Burl, T. Stough, W. Colwell, E.B. Bierhaus, W.J. Merline, and C. Chapman. Automated detection of craters and other geological features. *Int. Symposium on Artificial Intelligence, Robotics and Automation for Space*, June 2001.

[3] Rebecca Castano, Tobias Mann, and Eric Mjolsness. Texture analysis for mars rover images. *Application of Digital Image Processing, SPIE*, 3808, July 1999.

[4] Yang Cheng, Andrew E. Johnson, Larry H. Mattheis, and Aron A. Wolf. Passive imaging based hazard avoidance for spacecraft landing. *I-SAIRAS*, 2000.

[5] V. Gor, R. Castano, R.Manduchi, R.C. Anderson, and E. Mjolsness. Autonomous rock detection for mars terrain. *AIAA*, 2001.

[6] V.C. Gulick, R.L. Morris, M. Ruzon, and T.L. Roush. Autonomous science analyses of digital images for mars sample return and beyond. *30th Lunar and Planetary Science Conference*, March 1999.

[7] Timothy D. Halbrook, Jim D. Chapel, and Joseph J. Witte. Derivation of hazard sensing and avoidance maneuver requirements for planetary landers. *Advances in the Astronautical Sciences, Guidance and Control*, 107, 2001.

[8] Ayanna Howard and Homayoun Seraji. Intelligent terrain analysis and information fusion for safe spacecraft landing. *JPL Technical Report*, 2001.

[9] Andrew Johnson, Allan Klumpp, James Collier, and Aron Wolf. Lidar-based hazard avoidance for safe landing on mars. *AAS/AIAA Space Flight Mechanics Meeting*, February 2001.

[10] Betrand Leroy, Gerard Medioni, Andrew Edie Johnson, and Larry Matthies. Crater detection for autonomous landing on asteroids. *Workshop on Perception for Mobile Agents (CVPR)*, June 1999.

[11] W.Y. Ma and B.S. Manjunath. Edge flow: A framework of boundary detection and image segmentation. *IEEE Conf. on Computer Vision and Pattern Recognition*, 1997.

[12] Clark F. Olson. A general method for feature matching and model extraction. *Vision Algorithms: Theory and Practice*, 2000.

[13] Stein E. Strandmoe, Thierry Jean-Marius, and Stephane Trinh. Toward a vision based autonomous planetary lander. *AIAA GNC Conference*, August 1999.