Synthetic Environments for Simulated Missions[1]

Robert Gaskell
M/S 301 - 150
818-354-2116
Robert.Gaskell@jpl.nasa.gov

James B. Collier
M/S 126 - 234
818-354-3159
James.B.Collier@jpl.nasa.gov

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

Laura Ekroot Husman
M/S 126 - 234
818-354-2244
Laura.Husman@jpl.nasa.gov

Richard L.Chen
M/S 300 - 320
818-354-1259
Richard.L.Chen@jpl.nasa.gov

Abstract—Synthetic environments offer Space Flight projects an opportunity to perform rapid, comprehensive, and rigorous modeling of the critical elements of a mission in order to compute quantitative measures of design performance, associated risk, and actual mission utility value. Significantly, these studies can be performed early in the mission cycle. These synthetic terrains are generated on parallel, high-performance computers and served to remote simulations at near-interactive speeds. The terrain creation uses a realistic sequence of physical phenomena such as cratering and dusting and with parametric control of features such as surface roughness and rock density. Terrain resolution may be arbitrary but typically ranges from 0.01 to 10 meters. The terrain server has been used for two rover simulations, one using the actual Sojourner autonomy flight code, and the other a more general algorithm. Monte Carlo studies of rover designs interacting with synthetic environments were executed in parallel in order to quickly compute performance statistics and risk estimates.

1. INTRODUCTION

The traditional flight project development approach results in many independent, but nevertheless interacting, design decisions for each of the various mission components and these are usually each made on a qualitative, best educated judgement basis. Using synthetic environments provides a way to perform Monte Carlo experiments on very large numbers of design combinations early in the mission cycle in a wide variety of possible mission environments. This can provide an understanding of design decisions and design interaction on the whole system. Moreover, related Monte Carlo experiments can easily provide perturbation analysis of key mission design parameters which may be difficult or impossible to perform analytically. The critical point here is that numerical estimates of performance can be used statistically to predict probability of mission success, thereby providing a way to make important mission design decisions quantitatively rather than qualitatively. Simulation of the target environment is key.

Current technology allows for far greater simulation and modeling early in the design cycle. The ability to run and test designs, algorithms and interfaces is indisputably critical to mission success. When the simulation capability is agile and can be run rapidly on parallel machines, additional value is gained in numbers and types parameters that can be explored early in the process. For example, a single scenario of a rover of a given size, with a given navigation algorithm, and a synthetic terrain with an expected rock density is a good system test. It can be compared to a mechanical rover performing on earth in a rocky playground. First, the ability to run the simulation many times with different starting and ending locations will give better confidence in the performance results. However, many simulation runs on terrains with rock densities from across the distribution of possible rock densities will give better confidence in the variability of the performance with environmental variations. The size and the navigation algorithms used can also be varied. Agile simulation allowing exploration of the design space, within a synthetic environment that can vary across the environmental uncertainty space will greatly enhance the mission design and decision making.

The advanced engineering environment we have built generates synthetic terrain on parallel, high-performance computers and serves it at near- interactive speeds. The terrain generation software realistically and consistently models physical phenomena such as rocks, and craters at arbitrary resolutions, typically ranging from 0.01 to 10 meters. Furthermore, those parameters and others such as the rock and the crater density can be set by the user of this software, which allows a client to tailor the synthetic terrain for its own requirements.

Early applications of synthetic environments has been to rover design since there are obvious problems in the design of space borne rovers which make it an ideal candidate. There is no way to test a Martian rover in the actual Martian environment before the mission. Testing can be performed on Earth before launch by finding similar physical areas or constructing sites which are believed to approximate Martian conditions. However, there will, in general, be only

[1] 0-7803-6599-2/01/$10.00 © 2001 IEEE

a few such physical test sites. Moreover, hardware always appears late in the mission cycle which greatly limits the time the rover can be tested. Moreover, there are never very many of these expensive devices to test. As a result, a design problem which appears late in the mission cycle may be very hard to correct, especially since it is notoriously hard to unbend metal. It is simply very difficult to revisit and modify early design decisions late in the development process precisely when much more has been learned about the functioning of the entire system.

The first client to use this environment was a simulation of the Sojourner Rover, which created and navigated on a simulated Martian terrain using the actual Sojourner flight code. Large numbers of identical rovers having different initial conditions executed in parallel and the resulting Monte Carlo experiments provided performance statistics. The rovers were asked to traverse terrain with different levels of rock and crater density chosen to correspond to the known conditions encountered by actual missions such as Viking.

A second client was a newer navigation algorithm designed originally as a replacement for the Sojourner but applicable to a wider class of rovers. This simulation was far more robust and included numerous runtime parameters such as critical rover dimensions. Using Monte Carlo experiments, this system easily supported perturbation studies of rover design elements interacting with different simulated rover environments.

Moreover, the environment well suited the new algorithm since the terrain information included soil composition, in this case either sand or rock. Simulation could, as a result, be performed without introducing any specific sensor models that would themselves affect the behavior of the navigation algorithm.

Future work includes generating synthetic terrain that is an elaboration of an actual location being considered as a candidate mission target site. The latest available scientific data typically has resolutions of at best 30m while rover and landing simulations require resolutions on the order of 2cm. However, this observational data can be used to constrain the synthetic terrains which are generated so that they are consistent with the known data. For example, the spacecraft Mars Global Surveyor (MGS) provides image data, which can be used to establish an upper bound on craters and rocks, altimeter data, used to establish a sparse web of accurate elevations, and albedo data. Image data from both MGS and Viking can also be used with photoclinometry algorithms to fill in estimated elevations and slope variations.

The synthetic terrain environment will also expand to provide a rich set of simulation services and easy programming interfaces to make adopting this environment an attractive and even dominant choice for a wide spectrum of system modelers and designers.

2. Theory of Synthetic Terrain Generation

2.1. Basic Structure

The artificial surfaces provided by the terrain server are characterized by maps which give the elevation and albedo at regularly spaced surface points. Other information, such as dust characteristics and coverage, could be easily included. For example, one additional map is used to keep track of the fractional rock coverage, so that rocks need only be added in the final stages of construcion.

Surfaces are built sequentially. A preexisting surface is acted upon by a module representing some geological process such as cratering, to produce a new surface. That surface can then be acted upon by additional modules.

At any given resolution, the surface properties sampled at a point are representative of that point, and not averaged over a bin. This is an essential ingredient in the construction. Higher resolution surfaces retain the subset of lower resolution points, resulting in a consistent zooming behavior.

Each module alters the surface by proceeding from lower resolution to higher. The surface points are classified according to levels, with higher level points spaced further apart. Each point, except for those at the highest level, has a set of four higher level "parent" points which participate in its alteration.

This computational structure[1] was designed so that a portion of the surface could be constructed without building the entire surface. Originally, this was required so that small patches of surface, such as those seen by a rover, could be computed "on the fly" at a resolution of a few millimeters. The same structure allows for complete parallelization of the construction.

2.2. Specific Processes

Surface texturing is accomplished by using stochastic interpolation. The height values at the parent points are averaged to give an initial height for the child, which lies at the center of the four. To that initial height is added a random variation, the amplitude of which is a function of the spacing of the parents and of a random variable. The precise form of the amplitude function determines the roughness character of the surface.

Each point of the surface has its own random number generator for producing these variations. This guarantees that a higher resolution surface can be generated which is entirely consistent with an embedded lower resolution one. Although it has not been tried, there is no reason why the parameters of the roughness function could not themselves have a spatial dependence, allowing different areas of the surface to have different values for initial roughness.

A totally artificial surface begins with a functional form for the variation amplitude, and the heights at the four bounding corners. The basic spacing of the points is arbitrary. An interpolated surface begins with digital elevation and albedo maps which define the basic scale to within a factor of a power of two.

The initial surfaces are acted upon sequentially by modules representing various geological processes, a good example of which is cratering.

The primary cratering distribution is taken to be a power law,

$$N=N_0(D/D_0)^{-A}$$

where N is the number of craters per unit area with diameters greater than D, and A ~ 2. Secondary cratering can also be included, with parameters and cutoffs determined by the primary diameter. The power law is steeper for secondaries, with A ~ 3.5. Secondaries were included to facilitate the study of saturation equilibrium in cratering[2,3,4].

The first step in the cratering process is the assignment of impact points, diameters, and times of impact. The level structure of the surface plays an important role in this construction. The array of points with levels $\geq 2k+1$ form a grid with spacing 2^k basic units. The subset of points with level equal to $2k+1$ divides the surface into diagonally oriented squares with 2^{k+1} unit diagonals and centered at those points. Each of these squares is cratered independently, with craters ranging from some diameter D_k to $2D_k=D_{k+1}$. The diameter bin defined by D_k is somewhat arbitrary, with larger values speeding up the overall calculation. The only requirement is that the largest crater, plus its ejecta blanket, should not be able to extend past the adjoining squares.

The random variable associated with the central point determines the distribution of diameters, impact centers and impact times. Once these have been assigned, the craters are ordered according to time, and the actual cratering can begin. This is done one crater at a time, so that later craters can impact on earlier ones.

The actual cratering process begins with the annihilation of the affected area. Some high level points are retained, and a new surface with its own roughness function is constructed. The rock potential in the actual crater and most of the ejecta blanket is zeroed out. The surface heights are then adjusted according to a crater function that is bowl shaped with a depth of 20% of the diameter, which rises above the initial surface near the rim, and then decays in an ejecta blanket that drops in thickness like the inverse cube of the distance from the center[5]. The total volume of the ejecta blanket is equal to the volume excavated from the bowl. As the diameter of the craters become larger, the bottoms become flatter and their depths never exceed 6 km.

The fringes of the ejecta blanket are feathered into the surrounding unaffected terrain. Finally, the rock potential is enhanced to reflect those rocks produced by the cratering. An exponential dropoff with distance is used, which provides a good fit to the data[6].

The addition of rocks is similar to cratering, except that large rocks are added first, no attempt being made to time-order their creation. The fraction of the surface covered by rocks of diameter greater than D (in meters) is taken to be [7]

$$f=f_0*\exp(-qD)$$

where $q=1.79+0.152/f_0$, and where f_0, the quantity carried by the rock potential, represents the fractional coverage by all rocks. This leads to a number density per km^2 of

$$N=(4\times10^6/\pi)*q^2*f_0*\left(\frac{\exp(-qD)}{qD}-E1(qD)\right)$$

for circular rocks. The actual rocks are oblong, and randomly oriented, so the true fractional coverage may be different from the input value. They have their own surface texture and albedo. They tend to be higher when f_0 is larger, reflecting that less of the rock is covered by dust.

2.3 Stereo-photoclinometry

The goal of the current work is to enhance a preexisting topographic and albedo map of a real Martian surface with consistent subresolution terrain. To some extent, the trends in roughness vs. scale can be extrapolated to smaller scales, but this would ignore important structures, such as rocks, which appear only in the very highest resolution images. To some extent, thermal inertia data can give an estimate of the fractional rock coverage. Still, it is essential that appropriate geological input be available to limit the processes that might be contributing.

The high resolution real surfaces which will form the starting point for the subresolution extrapolations will be created using a form of enhanced stereo-photoclinometry

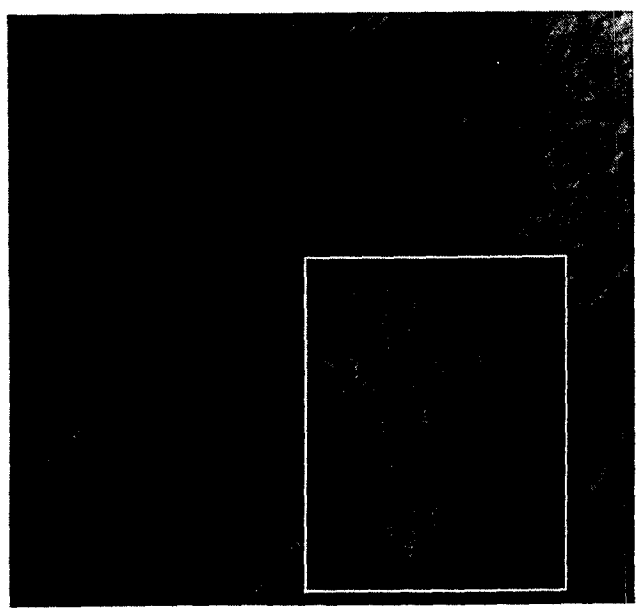


Figure 1

which has been developed over the last few years. The initial work was done during Mars precision landing studies, but the procedure has been refined in small body analysis for optical navigation. In figure 1, an image made from digital elevation and albedo maps derived from Viking imagery is compared with a portion of a MGS wide angle image in Gusev crater.

The starting point for this procedure is a set of images which have been registered to subpixel accuracy, a good camera model, and a good knowledge of body-fixed spacecraft (camera) location or of camera pointing. The very procedures used for stereo-photoclinometry were originally developed for alignment of landmarks, typically to 0.2 pixels rms[8], and the registration of images, typically to a few hundredths of a pixel.

Once the images have been aligned, portions of the surface can be extracted from the imaging data and projected into the same coordinate system, usually south, east and vertical in a body-fixed frame. Of course, since the vertical location of the surface patch is unknown, some initial work must be done to align the centers, just as was done for the landmarks.

The next step is the determination of the height of each point of the extracted map relative to the defining coordinate system. Using the brightnesses of each point in each projected image, the slopes and albedos are determined in a least squares fit. The slopes are integrated, gradually winding out from the center, to determine the height of each projected point. Up to 450 paths are used in each integration to average down the errors. As the height estimates improve, further projections of the images are made. The solutions for slope and albedo become sharper, and the integrations more reliable.

If the imaging data were perfect, and if the photometric properties of the surface were precisely known, integration of the slopes would probably be sufficient. Unfortunately, this is not the case, although a 35% Lambertian, 65% Lommel-Seeliger model seems to mimic the surface photometrics pretty well. The slopes and albedos determined during the previous stage can be used to predict the images seen in each camera and sun location. Significant portions of these predicted images can be correlated with the actual images to determine the locations of those portions stereographically. The heights of these stereo points are used to constrain the slope integration.

This technique allows one to obtain elevation and albedo maps at a resolution nearly as good as the imaging itself, much bettrer than stereo alone. Because albedo is also determined, at least three images at different camera and/or sun angles are needed. Since the stereo pairs provided by Mars Global Surveyor are not sufficient, they will be supplemented by Viking Orbiter frames. In addition, MOLA data will be used to augment the stereo points during the high resolution integrations.

3. Synthetic Environment Package

The synthetic terrain environment package consists of three software components: a terrain generator, a terrain server, and a terrain client. The generator, based on the theoretical work in the previous section generates synthetic terrain. The server manages these terrains and serves portions of them on demand to remote simulation clients such as a rover simulator.

3.1 Terrain Generator

The most important component of the synthetic terrain environment is the terrain generator software system whose theory is described in section 2. Its features include detailed modeling of physical phenomena, fractal-based algorithms, and parallelized execution for rapid generation

The user may control the terrain generator by specifying the physical phenomena which should occur to shape the surface, the order in which these phenomena occur, and critical parameters constraining these phenomena. For example, the caller could request:

1. a base terrain sloped toward the southeast, then
2. rocks randomly distributed with a density of 17.6% (the same as the density at the Viking II landing site), and finally
3. craters having diameters between 10 and 1000 meters.

The algorithms of the generator are based on fractals. This results in any piece of terrain having consistent features regardless of the resolution. For example, at a resolution of 0.01 meters, the terrain generator may place a 2-meter rock at 5 meters north and 3 meters east of the origin. If so, the terrain generator would still place a 2-meter rock at that location even if re-run at a resolution of 1.0 meters, though the rock would occupy fewer pixels.

The original version of this program was a suite of multiple programs written by Dr. Robert Gaskell. These have been integrated into a single program which executes in parallel. This parallel version executed on a multiprocessor supercomputer can result in a reduction of the terrain generation time by up to 98%.

3.2 Terrain Server

The first use of the terrain generator as part of a synthetic environment was for Monte Carlo simulations of the Mars Pathfinder rover. This predated the creation of the terrain server and client. The simulations of the rover's autonomous navigation system ran successfully but quite slowly on a single processor workstation. For each discrete rover motion (which cannot exceed 65mm), the terrain generator was executed even though the rover's position stayed well within the previously generated patch.

The terrain server was created to reduce the execution time for simulations by eliminating the recomputation of terrain.

When a client requests a terrain patch, the server checks the directory for the library of already generated terrain. This directory is referred to as the metadata. If the terrain already exists, the requested patch is returned. If not, then the requested patch is generated and added to the the metadata directory before being returned.

Besides returning existing terrain, the server can also perform some simple image processing in order to do the following: bi-linearly interpolate between pixels, rotate or flip terrain, and down-sample pixels. Interpolation has been used by rover simulators that can physically move in smaller increments than the resolution of their sensors. Rotating and flipping is used by clients that request different coordinate systems. Down-sampling has been used by a spacecraft entry, descent, and landing simulation (EDL), in which the resolution of the spacecraft sensors changes based on its altitude.

3.3 Terrain Client Interface

A client application accesses the synthetic terrain environment through a standard library of functions or API. These functions automatically connect the client to the remote terrain server and manage the communication with this server.

The primary goal was to keep the interface simple. A client can recover most of the terrain information it needs through calling a single function, getDEM(), which takes location, size, and phenomena as inputs and provides a digital elevation map (DEM) as output. There are additional entry points such as one to return the albedo.

If a client requests a patch that the terrain server does not have metadata for, the terrain server executes the terrain generator for that patch and a large surrounding area. This may take minutes, so the library offers the function pregen(), which allows a client to tell the server to generate terrain in advance.

3.4 Future Work

The most important work would be to incorporate future work by Dr. Gaskell into this environment, such that the environment as a whole would still appear to be automated and seamless to any client.

The terrain data that a client can access currently consists of an elevation map, an albedo map, and an optional grayscale image. The terrain generator currently generates all three. Since generating the image consumes most of the computation time, the parallelization of this component my be significantly modified. An alternative is to create models of specific terrain sensors such as a laser ranger or a CCD. In order to simulate the scientific activity of a rover it may become important to return additional information about the terrain surface such as a classification map; for example, the geologic composition of the terrain pixel by pixel.

4. Rover Simulations as Example Clients

Ideal applications for the synthetic environments described above are the simulations for the study of rover navigation algorithms on the Martian surface are The actual flight code or candidate flight code can be used directly in the simulation. Error sources can be controlled or entirely eliminated from sensor input so that the "pure" navigation algorithm can be studied. Similarly, error sources in the rover motion can also be controlled or entirely eliminated.

4.1 Performance metric

Rovers are placed on the synthetic terrain and given another location as a goal to reach. As a metric of performance for this task we used the linear distance of the goal from the start location divided by the actual distance traveled by the rover to reach this goal. This is a measure of rover efficiency. Clearly efficiency is always some value between 0 and 1 where 1 is the best possible performance and 0 is the worst. This is a useful metric since rover power is limited as is the expected life time of the rover in the martian environment.

4.2 Sojourner algorithms

The first application of synthetic environments was a test of the autonomous Martian surface operation of the Sojourner behavior-based[9–12] rover navigation algorithm developed at JPL by Jack Morrisson.
The high level structure of the rover simulation system is the following:

1. The rover requests a small patch of synthetic terrain immediately around the rover using Chen's Terrain Server. The size of this patch depends on the sensing distance of the rover. For Sojourner this distance is a few meters. A terrain resolution of 2 cm was selected as being adequate for describing a surface appropriate for driving a rover.
2. A model of a sensor senses the surrounding synthetic terrain. For Sojourner this was a set of 20 laser ranging devices and a tilt sensor.
3. The input sensor data is processed to create an internal model of the surrounding terrain. This model usually includes an estimate of a digital elevation map for the surrounding terrain. Salient characteristics are extracted from this internal terrain model for the use of the navigation algorithm. For Sojourner the orientation of the rover received from the tilt sensor and the 20 ranges comprised the data required by the navigation algorithm.
4. A rover navigation algorithm takes the sensed terrain characteristics as input, combines this with its current internal state, and computes a new state. The new state may include the output of a move command for the external rover. The new state could also be the determination that the rover has arrived at its goal state, in which case the rover stops, or it could be a call for

help sent to a remote operator on Earth. In general, the output is a command to turn right or left and/or move forward or backward. For the Sojourner simulation, the navigation algorithm used was the actual Mars Pathfinder flight code.

5. The rover executes the output rover motion command and is now in a new location on the synthetic terrain. The cycle can now repeat with step 1.

It is important to realize that while all other parts of the system are simulations, the navigation component is the actual flight software. Thus this approach provides a way to perform orders of magnitude more trials on flight software than could possibly be achieved using physical rovers.

A standard test used in our early simulations was to start all rovers distributed evenly about a unit circle of radius 25 meters with the center of the circle as the common goal. An initial single processor implementation executing one rover case at a time and generating synthetic terrain as needed required approximately 2 months to perform a Monte Carlo experiment sufficient to produce meaningful statistics. The introduction of rapid synthetic terrain on demand using the Synthetic Terrain Environment server and the parallel execution of all rovers simultaneously on a 256 node supercomputer can now execute a 100 rover Monte Carlo experiment in about 30 minutes.

When executed on Viking II rock density, 17.6%, rover efficiency varied from 0.280 (89.3 meters traveled) to .0277 (902.5 meters traveled) with an average efficiency of 0.0893 (280 meters traveled). It should be remembered that the Sojourner algorithm was never designed for autonomous execution and the autonomous experiment planned for the end of the mission was canceled due to loss of contact with the rover.

We also performed similar experiments using rock densities of 6.9%, 8.8%, and 11.8%. The qualitative assessment that rovers will perform better on a lower rock density was certainly confirmed but the quantitative evidence indicated that the variance in rover behavior was unacceptably high except for the 6.9% density surface.

We next performed simulation experiments using a second generation navigation algorithm designed by Marcel Schoppers of JPL as a replacement for the Sojourner algorithm.

The high level structure of the Marcel rover system is very similar to the Sojourner rover system. Functionally, the only differences are in steps 2 and 3. It is assumed here that there is some "virtual sensor" which can see a rectangle of terrain about 1m × 2m in size in front of the rover where the 1m is in the forward direction. This rectangle is subdivided into four roughly equal rectangles from left to right in front of the rover. The "virtual sensor" data is processed so that each of these four rectangles is classified as either "safe" or "unsafe" for the rover. These four bits of information are the only data input to the navigation algorithm.

This classification process requires not only the digital elevation map of this 1m X 2m rectangle but also its material composition pixel by pixel. The synthetic terrain classifies each pixel as either sand or rock and returns this information as part of the albedo classification. This means that the "virtual sensor" processing performed to compute these four bits of input to the rover navigation algorithm can be achieved with perfect accuracy. As a result, we can study the performance of this navigation algorithm in insolation from any sensor error.

Monte Carlo experiments with this rover design on an 8.8% rock density terrain resulted in an average efficiency of 0.89 (averaging 44.9meters to reach a goal 40meters away). The user also has access to certain design parameters such as critical rover dimensions and minimum hazard clearance as well as certain navigation tuning parameters. This allows a much wider scope for quantitative perturbation studies of the effects of design changes on performance. In addition, it is worth noting that there was an unexpected benefit of this simulation system. When the first version was demonstrated to the designer of the navigation algorithm, Marcel Schoppers, he was immediately able to detect several anomalies in the expected rover behavior and correct them within a few minutes.

ACKNOWLEDGEMENTS

The authors would like to thank Marcel Schoppers for his navigation algorithm, as well as guidance and energy. Thanks also to Bob Shishko and his team who motivate and interpret the Monte Carlo results into risk assessments.

This research we carried out at the Jet Propulsion Laboratory, California Institute of Technology. Under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] R. W. Gaskell, "Martian Surface Simulations," *Journal of Geophysical Research-Planets, 98:(E6), 11099-11103,* June 25, 1993.

[2] W. K. Hartmann, R. W. Gaskell, "Planetary cratering 2: Studies of saturation equilibrium," *Meteoritics & Planetary Science, 32:(1), 109-121,* January 1997.

[3] W. K. Hartmann, R. W. Gaskell, "Confirmation of Saturation Equilibrium Conditions in Crater Populations," *Abstracts of the 24th Lunar and Planetary Science Conference, Houston, TX, p. 611,* March 15-19, 1993.

[4] W. K. Hartmann, R. W. Gaskell, "Crater Saturation Equilibrium in Ancient Uplands: Preliminary Results of

New Modelling," *Abstracts of the Lunar and Planetary Science Conference, volume 23, page 495*, 1992.

[5] H.J. Melosh, Cratering mechanics - observational, experimental and theoretical, Annual Review of Earth and Planetary Science, volume 8, 65-93, 1980.

[6]. Moore, et al., unpublished data, 1969.

[7] M. Golombek and D. Rapp, Size-frequency distribution of rocks on Mars and Earth analog sites: Implications for future landed missions, Journal of Geophysical Research, volume 102, 4117-4129, 1997.

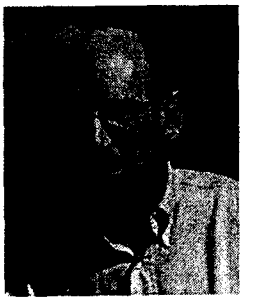
[8] R. W. Gaskell, "Digital Identification of Cartographic Control Points," *Photogrammetric Engineering and Remote Sensing, 54:(6), 723-727, Part 1*, June 1988.

[9] R. Brooks, "A layered control system for a mobile robot." *IEEE Journal of Robotics and Automation 2:1 14-23*, 1986.

[10] R. Brooks, J. Connell, and A. Flynn, "A mobile robot with onboard parallel processor & large workspace arm," *Proceedings AAAI National Conference* 1986.

[11] R. Brooks, "Intelligence without representation," *Artificial Intelligence* **47**, 1991.

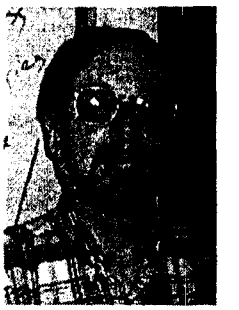
[12] R. Brooks, "Intelligence without reason," *Proceedings IJCAI 569-595*, 1991.



Bob Gaskell *received his ScB in Physics in 1967 from Brown University, and his PhD, also in Physics, in 1972 from McGill. After several years of postdoctoral work in high energy physics at Carleton University and the University of Toronto, Bob returned to McGill as a research associate. In 1978, he moved to Lafayette College as an assistant professor. During six years at Lafayette, his area of interest shifted to group theory, continuing his work with colleagues in Montreal.*

In 1984, Bob joined the Optical Navigation group at JPL, where he has been ever since. He divides his time between developing methods for the precise determination of planetary and small body shape and topography from optical imagery, and generating simulated shapes and surfaces for planets, asteroids and comets.

Bob is married, and has three children.



Jim Collier *is a Senior Member of the Information Sciences and Computer Science Staff at the Jet Propulsion Laboratory. He has extensive experience in software design and development including parallel applications, simulations, SAR and AVIRIS processing, and navigation. Early JPL work includes cognizant programmer for JPL mission navigation system resulting in numerous awards for design and development of innovative software. Mathematical research includes computer aided theorem proving.*

Laura Ekroot Husman *is a Technical Group Supervisor in the Exploration Systems Autonomy Section, Jet Propulsion Laboratory, Pasadena. She has been at the Lab since 1992 and has been working in the Advanced Laboratory for Parallel High-Performance Applications (ALPHA) group since 1995. She received her B.S. from Caltech in 1986 and her M.S. (1987) and Ph.D. (1991) degrees in Electrical Engineering from Stanford University.*

Richard Chen *received his BS in EECS from UC Berkeley in 1987 and his MS in CS from UCLA in 1991. He has worked at JPL since 1989. He was a member of the Parallel Applications Technology group from 1998 to 2000 and is currently a member of the Physical Oceanography Distributed Active Archive Center.*