Autonomous Robots 6, 131–146 (1999)
© 1999 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Traversability Analysis and Path Planning for a Planetary Rover

DONALD B. GENNERY
Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California 91109-8099
gennery@robotics.jpl.nasa.gov

Abstract. A method of analyzing three-dimensional data such as might be produced by stereo vision or a laser range finder in order to plan a path for a vehicle such as a Mars rover is described. In order to produce robust results from data that is sparse and of varying accuracy, the method takes into account the accuracy of each data point, as represented by its covariance matrix. It computes estimates of smoothed and interpolated height, slope, and roughness at equally spaced horizontal intervals, as well as accuracy estimates of these quantities. From this data, a cost function is computed that takes into account both the distance traveled and the probability that each region is traversable. A parallel search algorithm that finds the path of minimum cost also is described. Examples using real data are presented.

Keywords: mobile robots, sensor fusion, terrain mapping, obstacle avoidance, uncertainty

1. Introduction

Since the 1970s, research on planetary rovers has been conducted at various times at JPL. In the 1980s, it was planned to use a highly capable Mars rover to traverse long distances, perhaps aided by high-resolution (roughly one-meter) stereo pictures taken from orbit, so that only occasional commands from Earth would be needed (Randolph, 1986; Mankins, 1987; Wilcox and Gennery, 1987; Gennery, 1989). More recently, the emphasis has been on simpler vehicles needing more frequent help from Earth, such as the Pathfinder lander and rover that arrived on Mars in July, 1997 (Shirley and Matijevic, 1995; Matthies et al., 1995a). However, because of the propagation delay from Earth, considerable autonomy in a Mars rover is highly desirable, especially if it must drive far between sample sites. Also, because of the difficulty in communication, autonomy may be needed for a vehicle on the far side of the Moon. This paper describes some techniques that will be useful in path planning by such vehicles, and are improved from earlier versions that were briefly described previously (Gennery, 1991).

The vision system of a planetary rover may include a laser range finder, stereo vision, and other techniques such as shape from shading. The results from all of these and any other information sensing the surface at any one rover position, such as the wheel positions, can be combined to produce an unequally-spaced height (elevation) map, with accuracy estimates. So far the research described here has used only stereo vision (and the wheel positions). A stereo vision method and a way of matching local maps produced by the rover vision system and, if available, a global map produced from an orbital camera or other source, so that they can be merged, have been described previously (Gennery, 1989), and have been used to generate the examples in this paper. However, the techniques described herein are essentially independent of the origin of the height map.

Before a path can be planned, appropriate information about the traversability must be extracted from the vision data and represented in a suitable form. Many approaches to this task are possible. For example, individual obstacles could be detected and represented geometrically (Gennery, 1979; Hebert et al., 1988), the free space available to the vehicle could be represented by a simple mathematical description (Brooks, 1982; Giralt et al., 1979), or the ground surface could be represented by triangular planar surfaces identified as either traversable or nontraversable (Stephens et al., 1989). In such cases, standard graph searching algorithms can be used to find the shortest path. These approaches have the problem that only the length of the path is minimized; other information, such as the ease of traversing a given area or the uncertainty in its properties has been discarded.

Potential fields have been used by some (e.g., Brooks, 1986) as a path planning method. In the common form of this approach, the goal produces an artificial attractive force and obstacles produce repulsive forces. The direction of the net force on the vehicle determines how it should move. However, such techniques do not produce an optimum path for a vehicle, and they can get stuck in local minima. Simulating fluid flow (Feder and Slotine, 1997) eliminates the local minima, but the path is still is not optimum, and the computations may become very involved for complicated objects.

grid model

Another approach is to represent the area of interest by a regular grid (two-dimensional for a surface vehicle). The grid points cover the area densely, and each grid point contains information about the traversability and possibly uncertainty. As long as the spacing of the grid is sufficiently small, all of the information important for path planning has been preserved. Usually, the grid is a rectangular array, with equal spacings in the two dimensions. The connectivity of the grid points can be considered to be either four-neighbor or eight-neighbor. The four-neighbor approach can result in poor accuracy, since a straight line at a 45° angle relative to the grid would have a length differing by a factor of $\sqrt{2}$ from the nearest path along the grid. With the eight-neighbor approach, the maximum error is about 8%. An alternative would be to use an hexagonal array with six-neighbor connectivity, in which case the maximum error is about 15%. Because of its better accuracy, a rectangular grid with eight-neighbor connectivity has been extensively used (e.g., Thorpe, 1984) and is used here, with the modest cost of having to consider two kinds of neighbors.

Several pieces of information are desired at each grid point. The magnitude and direction of the slope of the surface are important in determining whether a given vehicle can drive through this point and how much time and energy it will expend in doing so. Of necessity, slope represents an average value smoothed over some area. Also of interest is the roughness of the surface, which represents how much deviation from the smoothed surface exists on a smaller scale. The height of each point also may be needed if it is desired to investigate individual wheel placements. The accuracy estimates of all of these quantities are very important, since, while the nominal values of these quantities may indicate a navigable region, the accuracy of the vision system might be such that there could be considerable doubt. Section 2 describes a method of computing smoothed height, both components of slope, the roughness, and variances and covariances of these quantities at equally spaced points, from the unequally spaced position data from a vision system. Section 3 describes a cost function, computed from these, that a path planner would try to minimize over a planned route. This cost includes the probability that a given grid point is nontraversable by a vehicle with specific capabilities.

Given such a cost function defined on a grid, many methods are possible to plan a path from the current vehicle position to a goal position. One method is A* search (Nilsson, 1971). Several researchers have used it to find the optimum path among the nodes of the grid (e.g., Thorpe, 1984), by the following method. The cost is integrated forward from the start point or backward from the goal, to produce a partial array of integrated costs. In some cases, an array of pointers to the adjacent node that produced the path to each node with minimum cost also is maintained. At the end of the integration, the path can be traced by means of the pointers, or by incrementing the integrated cost of the adjacent nodes by the cost of moving there from a node on the path and finding the minimum. A related method is D* (Stentz and Hebert, 1995), which is A* with the additional ability to replan partially when previously unknown obstacles are encountered. Slack and Miller (1987) developed a method that can operate in environments that change over time.

A* for pathing

Witkowski (1983) developed a method of planning a path on a grid that is similar to A* as described above, with two differences. First, it uses a breadth-first search, instead of an ordered (best-first) search such as A*, which uses heuristics to decide which nodes to pursue first. Therefore, it usually requires more nodes to be searched than does A*. This makes no difference if it is run on a parallel computer with one computational node for each node in the grid. On an ordinary computer, this extra search slows down the computations. On the other hand, the Witkowski method is very simple, whereas the extra computations in ordering the nodes to find the best one slows down the computations in A*. Which method is faster depends on the size of the grid array, the properties of the computer,

but doing a breadth-first search has lower computing costs – important on rover hardware and the nature of the particular data. (There is a variation on the method of potential fields for use on a grid (Ratering and Gini, 1995), that is similar in this respect to Witkowski's method and avoids the problems with potential fields mentioned above.)

The second difference is that Witkowski's method performs both the forward and backward integrations, resulting in two arrays of integrated costs. The sum of these two arrays then indicates the optimum path by means of those points with the minimum value. Although this is simpler than the above methods, it may be slower because of the two integrations. However, there is one potential advantage to Witkowski's method, related to the approximations involved in representing a continuous world by a grid of discrete points. Namely, Witkowski's method produces not only the optimum path on the grid, but also, for each node in the grid, the minimum cost of a path from start to goal that is constrained to path through that node. Therefore, it is easy to make changes to the path at a limited additional cost in traversability without much additional computation. (Apparently, this advantage was not utilized by Witkowski, who was mainly interested in the advantages on a parallel computer.)

The benefit from this feature of Witkowski's method (as generalized below) is that it makes it easy to make adjustments to produce a smoother path and to reduce wandering caused by the grid approximation. For example, consider the hypothetical cost function shown in Fig. 1 for a small 9×5 array, with the starting point in the lower-left corner and the goal in the upper-right corner. The search process produces the minimum cost path on the grid, shown as a dashed line, with a total cost of 966. However, the actual optimum path is shown by the solid line, with a total cost of 901. One of the paths that are closest to the optimum path is shown as a dotted line, with a total cost of 972. Since this differs from 966 by less than 1%, well under the 8% accuracy of the grid approximation, a reasonable threshold on the sum of the integrated arrays would permit the path to be adjusted to the solid, straighter line. A way of doing this is described in Section 4.

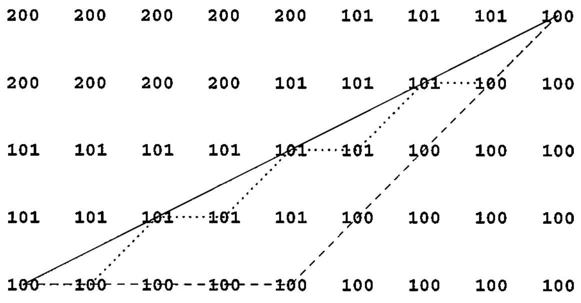


Figure 1. Example of cost function, showing effect of grid approximation. Solid line: the optimum real-world path, cost 901. Dashed line: the optimum grid path, cost 966. Dotted line: a grid path closest to the optimum real-world path, cost 972.

Witkowski's method used a binary obstacle map (with cost either directly proportional to distance or infinite). It is generalized in Section 4 to use a more general cost function, which necessitates changes in the growth phase of the algorithm, and the adjustments to the final path mentioned above are made.

It is assumed here that when a path is executed, it is possible to monitor the vehicle's position so that it can be controlled to follow the desired path. Several navigational techniques can be used for this purpose, including dead reckoning, feature tracking (Moravec, 1980), inertial navigation (Roberts and Bhanu, 1992), and terrain matching (Gennery, 1989). If the chosen technique has significant error, it can be allowed for in the cost function of the path planner by expanding obstacles.

2. Computation of Smoothed Height, Slope, and Roughness

The result from the vision system and possibly other sources is a set of unequally spaced points, each consisting of $\mathbf{r} = [x\ y\ z]^\top$ (where z is the vertical coordinate), its covariance matrix, denoted here by $\boldsymbol{\Sigma}_{\mathbf{rr}}$, and possibly the probability of correctness p_c (the absence of a gross error, as opposed to the usually small errors distributed according to the covariance matrix). It is desired to produce an equally spaced (in x and y) height map by means of smoothing (to combine points that are closer together than the desired spacing) and interpolation (to bridge gaps between points), to produce estimates of slope at the same spacing (by in effect differentiating the height map), and to produce estimates of surface roughness (by measuring deviations from the smoothed surface). Areas with small slope and roughness then can be considered safe for driving when a coarse analysis is done, or heights at a fine resolution could be used to determine safe placements of each wheel.

The above goals can be met by fitting (by means of weighted least squares) planes to small areas around each desired output point, using the coefficients of each plane for the height and slope, and using the residuals of the fit to estimate roughness. Two issues to decide are what is to be minimized in the fits and what weights are to be used.

Since we are considering z to be a function of x and y, the vertical distances from the points to the plane will be minimized. Thus, the solution is a linear least-squares adjustment for each fit, which is simpler computationally than minimizing the three-dimensional distances from the points to the plane. (The solution for a vertical wall then is zero slope and infinite roughness instead of infinite slope and zero roughness, but either combination indicates a nontraversable region.)

The weights used in the fits are a result of three considerations: the less accurate points (as indicated by their covariance matrix) should have less weight, the roughness of the terrain has the effect of less accuracy in the data as far as the fit is concerned, and points further from the center of the fit should have less weight (a Gaussian function is appropriate here). Because the vertical component of discrepancy is being minimized, the weight considering only the first effect should be the reciprocal of the variance of z. However, allowance must be made for the fact that the vertical discrepancy is influenced by horizontal error also, if the fitted plane is sloped. The proper thing to do is to compute an augmented variance of z for this purpose by using the current estimate of the slope $\mathbf{s}$ (=$[s_x s_y]^\top$), as follows:

$$\begin{aligned}\sigma'_{zz} &= [-\mathbf{s}^\top \quad 1]\, \mathbf{\Sigma_{rr}} \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \\ &= \sigma_{zz} + s_x^2\sigma_{xx} + s_y^2\sigma_{yy} - 2s_x\sigma_{xz} - 2s_y\sigma_{yz} + 2s_x s_y \sigma_{xy}\end{aligned}$$

(For any arbitrary variable u, the symbols σ_u^2 and σ_{uu} are used interchangeably here to denote its variance.) Since the above equation uses the slope to obtain the weights for computing the slope, an iterative process is required. However, by considering a constant typical slope magnitude s_τ (for which 0.4 has been used) and random slope direction, the following approximation results:

$$\sigma'_{zz} \approx \sigma_{zz} + \frac{1}{2}s_\tau^2(\sigma_{xx} + \sigma_{yy})$$

The implemented program uses this approximation on the first iteration, and each subsequent iteration uses weights computed according to the exact equation, using the slopes from the previous iteration. When speed is very important, just one iteration can be used, so that the weights are based directly on the approximation.

The effect of roughness is to add another component of variance to the height data. Therefore, a further augmented variance is computed by using the current estimate of roughness r (which represents the standard deviation of the actual surface about a local plane), as follows:

$$\sigma''_{zz} = \sigma'_{zz} + r^2$$

As before, the exact equation requires an iterative solution, with the roughness above being obtained from the previous iteration. On the first iteration, the implemented program uses the following approximation:

$$\sigma''_{zz} \approx \sigma'_{zz} + (\rho_\tau \sigma_g)^2$$

where σ_g is the standard deviation of the Gaussian function used for smoothing and ρ_τ is a proportionality constant (for which 0.3 has been used) giving the typical roughness in terms of the size of the Gaussian function. (The roughness tends to increase as the area over which it is measured increases, because natural surfaces usually have a fractal nature.)

Next to consider is an artificial extra component of variance σ_{ss} that causes the smoothing by increasing from zero at the point being evaluated to large values far away, so that nearby points get almost the full weight but far points get little weight. This is derived, somewhat arbitrarily, from the reciprocal of a Gaussian function, minus the center value, as follows:

$$\sigma_{ss} = (\kappa\sigma_g)^2 \left(\exp \frac{(x - x_g)^2 + (y - y_g)^2}{2\sigma_g^2} - 1 \right)$$

where κ is a constant scale factor (for which 0.3 has been used), and x_g and y_g denote the grid point for which output is desired. In order to prevent significant aliasing, the standard deviation of the Gaussian function σ_g should be at least half of the grid spacing. (We have used 0.8 times the grid spacing in many of our experiments, to produce some additional smoothing.) Ideally, when a path planner which considers the vehicle as a whole, such as that in Section 4, is used, σ_g should be about 0.4 times the size of the vehicle (since the area under a one-dimensional Gaussian function of unit height is $\sqrt{2\pi}\sigma \approx 2.5\sigma$), so that the computed slope represents an average slope over a region corresponding roughly to the vehicle, with the roughness representing deviations on a scale smaller than the vehicle. However, when individual wheel placements are considered, higher resolution is needed.

Then, if all of the input points are reliable, the weights for the plane fits are given by

$$w = \frac{1}{\sigma''_{zz} + \sigma_{ss}}$$

(This includes all of the above components of variance, but only $\sigma'_{zz} + \sigma_{ss}$ is used below for propagating error estimates into the results, since the roughness is a true variation in the surface to be fit, not a source of error.) However, if the stereo program or other source of the data has produced estimates indicating the probability of the absence of a gross error for each point, and if such errors have not been corrected manually or by a terrain matcher that compares the data to a standard such as corrected data from pictures from orbit, these probabilities should be included in the weights. Doing this in a statistically rigorous way would be complicated, but an intuitively reasonable approximation is just to multiply the above weight by the probability p_c for that point, to obtain

$$w = \frac{p_c}{\sigma''_{zz} + \sigma_{ss}}$$

and this is what the implemented program does when requested.

The computation of the smoothed, interpolated height $\bar{z}$ and the two components of slope s_x and s_y then proceeds as follows for each grid point (equally spaced values of x_g and y_g over the desired area), by using the usual formulas for weighted least-squares adjustments (Mikhail, 1976), with **B** here representing the coefficients in the condition equations, **N** and **C** representing respectively the coefficients and "constants" in the normal equations (partially reduced, in the nomenclature of Mikhail), and $\mathbf{N}_0$ representing weight on a priori zero values:

$$\mathbf{B} = [1 \quad x - x_g \quad y - y_g]$$

$$\mathbf{N} = \mathbf{N}_0 + \sum_{\text{points}} \mathbf{B}^\top w \mathbf{B}$$

$$\mathbf{C} = \sum_{\text{points}} \mathbf{B}^\top w z$$

$$\begin{bmatrix} \bar{z} \\ s_x \\ s_y \end{bmatrix} = \mathbf{N}^{-1}\mathbf{C}$$

where the summations are over all points sufficiently near the given grid point so that the Gaussian function allows them to have appreciable weight. (A distance limit of approximately $3.5\sigma_g$ is used currently.) The inclusion of $\mathbf{N}_0$ helps to constrain the solution to reasonable values in regions of very poor or sparse data. The implemented program uses the following:

$$\mathbf{N}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_{s_0}^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_{s_0}^2} \end{bmatrix}$$

where σ_{s_0} is a specified a priori standard deviation of slope about zero. Thus, there is some tendency to force both components of slope towards zero, but height is left free. The covariance matrix of $\bar{z}, s_x$, and s_y cannot be computed in the usual way as $\mathbf{N}^{-1}$, because the weights here are not the inverse of the error variances, but include the effects of roughness. Since, according to the above, the coefficient of each measured z value in the solution is $\mathbf{N}^{-1}\mathbf{B}^\top w$ and the error variance is $\sigma'_{zz} + \sigma_{ss}$, the covariance matrix is obtained as follows:

$$\mathbf{M} = \mathbf{N}_0 + \sum_{\text{points}} \mathbf{B}^\top w^2 (\sigma'_{zz} + \sigma_{ss}) \mathbf{B}$$

$$\begin{bmatrix} \sigma_{\bar{z}}^2 & \sigma_{\bar{z}s_x} & \sigma_{\bar{z}s_y} \\ \sigma_{\bar{z}s_x} & \sigma_{s_x}^2 & \sigma_{s_x s_y} \\ \sigma_{\bar{z}s_y} & \sigma_{s_x s_y} & \sigma_{s_y}^2 \end{bmatrix} = \mathbf{N}^{-1}\mathbf{M}\mathbf{N}^{-1}$$

The roughness r and its standard deviation σ_r then are computed as follows for the same grid points, by using a general method for variance adjustment (Gennery, 1980) and treating probability in the same way as above:

$$v = z - \bar{z} - (x - x_g)s_x - (y - y_g)s_y$$

$$k = \frac{1}{1 - w\mathbf{B}\mathbf{N}^{-1}\mathbf{B}^\top}$$

$$\omega = \frac{1}{2(\sigma''_{zz} + \sigma_{ss})^2} \quad \text{or} \quad \omega = \frac{p_c}{2(\sigma''_{zz} + \sigma_{ss})^2}$$

$$r^2 = \max\left(\frac{\sum_{\text{points}} \omega (k v^2 - \sigma'_{zz})}{\omega_0 + \sum_{\text{points}} \omega}, 0 \right)$$

$$\sigma_{r^2}^2 = \frac{\omega_0 + 2\sum_{\text{points}} \omega^2 k^2 (\sigma'_{zz} + \sigma_{ss})^2}{(\omega_0 + \sum_{\text{points}} \omega)^2}$$

As before, $\sigma'_{zz} + \sigma_{ss}$ represents error to be propagated into the resulting accuracy estimate, whereas $\sigma''_{zz} + \sigma_{ss}$ includes the effect of roughness and is used for deriving the weight. (Here υ represents the residuals from the above plane fits, ω represents the weights for the variance adjustment, ω_0 represents the a priori weight for zero roughness, and k represents the ratio of the variance of each observation to the variance of its residual in the fits.) Notice that, since roughness is defined here as the standard deviation about the fitted plane, the square of roughness, representing the variance, is solved for in the above variance adjustment. The roughness itself is obtained merely by taking the square root. Obtaining the variance of roughness, however, is a little more complicated. Because the standard deviation of the roughness can be larger than the roughness, the usual linear approximation $\sigma^2_{r^2} = 4r^2\sigma^2_r$ cannot be used. By using the fact that the fourth moment of a Gaussian distribution is $3\sigma^4$, a better approximation is $\sigma^2_{r^2} = 3\sigma^4_r + 4r^2\sigma^2_r$. (This is exact in the two limiting cases of $r = 0$, if the distribution is Gaussian, and $r \gg \sigma_r$, for any distribution.) Solving this equation for σ^2_r produces the following:

$$\sigma^2_r = \frac{1}{3}\left(-2r^2 + \sqrt{4r^4 + 3\sigma^2_{r^2}}\right)$$

which is used in the implemented program to obtain the variance of the roughness from the variance of the roughness squared produced above.

Setting r to zero in the approximation for $\sigma^2_{r^2}$ and taking the reciprocal produces the following equation for obtaining the a priori weight of zero in the roughness adjustment:

$$\omega_0 = \frac{1}{3\sigma^4_{r_0}}$$

where σ_{r_0} is the standard deviation about zero of the a priori roughness.

It remains to describe the criteria for terminating the iterations. First consider convergence. A good criterion is to declare convergence for any point if the change in roughness on any iteration is no more than a certain small portion α of its standard deviation and if the change in smoothed height and the two components of slope is within the $\alpha\sigma$ error ellipsoid determined by the covariance matrix of these three quantities. However, a precise test is not very important, so in order to save computing time, the implemented program uses a simple, crude approximation to the error ellipsoid test. First, only the two components of slope are used instead of including height (since height does not affect the weights). Second, the **N** matrix is used as an approximation to the inverse of the covariance matrix. (This is exact if the roughness is zero.) Third, only the main-diagonal terms in **N** are used. (Using their average instead of each eigenvalue in the direction of each eigenvector is exact when the error ellipse is a circle, but allows the change in values to be up to $\sqrt{2}$ times the semi-minor axis of the ellipse when the ellipse is very eccentric.) The resulting convergence test is as follows:

$$\left(\Delta s^2_x + \Delta s^2_y\right)(n_{22} + n_{33}) \le 2\alpha^2$$

and

$$\Delta r^2 \le \alpha^2\sigma^2_r$$

where Δ denotes the change from the previous iteration, n denotes an element of the **N** matrix, and currently $\alpha = 0.1$. This test is made starting on the second iteration.

Because of conflicting data, some points may never converge. In order to avoid wasting time with these, starting on the fourth iteration a test is made to see if the magnitude of the change in slope is decreasing. If it is not (that is, if $\Delta(\Delta s^2_x + \Delta s^2_y) \ge 0$), this point is declared to be nonconverging. In this case, an adjustment is made to the variances for this point to allow for the additional uncertainty in the unconverged value, by adding $\Delta\bar{z}^2$, Δs^2_x, Δs^2_y, and Δr^2 to $\sigma^2_{\bar{z}}$, $\sigma^2_{s_x}$, $\sigma^2_{s_y}$, and σ^2_r, respectively. (In principle, this test could be made starting on the third iteration, but one more iteration is allowed to give the points more chance to start converging.)

In regions of poor or nonexistent data, the **N** matrix may be singular or so nearly singular that numerical errors prevent a good inverse from being obtained. If tests (that currently check to see if **N** is within about 1 part in 10^6 of being singular) detect such a condition, singularity is declared for this point, the height, slope, and roughness are set to zero, and their variances are set to the a priori values (including a very large value for the variance of height).

Each point keeps iterating until it has been terminated by one of the above three conditions (convergence, nonconvergence, or singularity); otherwise, it is still active. However, if the number of active points stops decreasing (after at least ten iterations), or if the specified maximum number of iterations has been reached, the entire process stops, and (if more than one iteration has been allowed) the variances are augmented in the same way as for nonconvergence, using the changes on the last iteration.

An improvement that may be made in the future is to make the size of the Gaussian function for smoothing variable, so that in regions of sparse data it automatically becomes larger.

we may need a cost f^n

3. Cost Function

A cost function suitable for use by a path planner, such as the one in Section 4, will now be described. It uses the data produced by the slope and roughness computation in Section 2, computed on an equally-spaced grid over a specified area.

The cost function represents the cost of driving through each grid point. In general, this function could take into account many factors, such as the energy expended in going uphill and the slower speed needed over rough ground, for example, and could contain directional components. It could even include negative costs, to represent the desirability of gaining a closer look at something that appears to have scientific interest. However, so far the cost function used here is computed from only two components: the distance traveled and the probability that the slope or roughness may be too large to allow the vehicle to pass safely. Of course, the cost of any nonzero probability could be made infinite, but doing this may be too conservative because of the appreciable uncertainties that may exist in the sensor data (and unrealistic, because complete certainty can never be achieved), as long as the mistakes can be detected before the vehicle actually attempts to do something dangerous. A tentatively planned path can be partially executed, and then new images can be taken, which may allow the vision system to remove previous uncertainties in the remainder of the path because of the closer view. Also, a rover should have some kind of proximity sensor, such as mechanical feelers, that will enable it to detect obstacles, and inclinometers, that will detect tilt before it becomes excessive. Therefore, the cost due to probability allows for the cost of backtracking (and possibly waiting for help from Earth) in case an obstacle is not noticed at first.

The probability that any given grid point is traversable is the product of the probabilities that the slope is less than the maximum permissible slope and that the roughness is less than the maximum permissible roughness, assuming that the errors in these are independent. These probabilities are approximated by assuming that the errors have the Gaussian distribution. For the roughness, the probability then follows simply from the integral of the one-dimensional Gaussian function, which is easily computed to high accuracy by standard algorithms. However, the situation for slope is more complicated, since slope is two-dimensional and the uncertainty can be as large as the slope itself. The correct thing to do would be to integrate the two-dimensional Gaussian function over a circle whose radius is the maximum permissible slope. To avoid this difficulty, at present as an approximation a Gaussian distribution is assumed for the magnitude of the slope ($\sqrt{s_x^2 + s_y^2}$), with a standard deviation equal to the greater of the standard deviations of the two components of slope (thus ignoring the correlation of the two components).

Since the cost function will be summed over a route, whereas probabilities of success should be multiplied, the logarithm of probability is actually used. Specifically, the cost per unit distance currently is defined to be

$$c = \frac{1}{a} - \frac{\ln p_t}{b}$$

but we don't have all these variables

where p_t is the above combined probability of traversability, b is the distance over which this probability is assumed to be highly correlated (normally about the size of the vehicle), and a is a scale distance that determines the trade off between the length of the path and probability of failure. Because $p_t \leq 1$, $-\ln p_t \geq 0$. Therefore, the portion of cost due to probability is always positive (or zero if the point is certainly traversable).

4. Path Planning

The method presented here uses a cost function such as that in Section 3, as computed on an equally-spaced grid over a specified area, and finds the best path through this grid by a fairly simple parallel process. This process is derived from the method of Witkowski (1983), but as discussed in Section 1, it is generalized to arbitrary cost functions, and some adjustments to the final path are made.

Before the cost function is used below, it can be expanded to allow for the width of the vehicle and uncertainty in path execution (although the smoothing done in the slope computation has a similar effect, so that this may not be necessary). The expansion is done by rounding the desired semiwidth to the nearest multiple of the grid spacing and then performing this number of expansion iterations, alternating between eight-neighbor growth and four-neighbor growth. On each iteration, each cost is replaced by the maximum of itself and its neighbors. Thus, an octagonal expansion is produced, as an approximation to the desired circular expansion.

The main computation then consists of a forward growth from the vehicle starting position in one array and a backward growth from the goal position in another array, in order to integrate the cost from the start point to each point in the grid and from each point to the goal point.

The two arrays are initialized to contain infinity everywhere except at the start point in the forward array and the goal point in the backward array, which contain zero. Then on each iteration each point in each of the two arrays is replaced by the minimum of its old value and the eight values obtained by adding to the old value of each of its eight nearest neighbors the cost of moving from the neighbor to this point (in the forward array) or from this point to the neighbor (in the backward array). This cost of moving is taken to be the average of the cost function at the two points times the distance between them, where this distance is equal to the grid spacing for the four nearest neighbors or $\sqrt{2}$ times the grid spacing for the diagonal neighbors. (In the general case, this would include the effect of the direction of the move.) The iterations could continue until there is no further change. Actually, since negative costs currently are not used, the iterations stop when the smallest new value that is inserted in the backward array on any iteration is greater than the backward value present at the start position, since the value at this position then can never change. (Equivalently, the forward array and goal position could be used for this test, except that the goal might be outside of the array, as described below.) Note that a point will change as soon as the shortest route (by "chess king" distance) reaches it, but it will change again if a cheaper route reaches it later. (This does not happen in the Witkowski algorithm.)

At the end of the iterations, at each grid point, the forward array contains the total cost of moving from the start point to this point by the cheapest route, and the backward array contains the total cost of moving from this point to the goal point by the cheapest route. The sum of these two arrays then is the cost of moving from start to goal through this point. Therefore, the minimum value of this sum occurs along the optimum path from start to goal, and this minimum value is equal to the value in the forward array at the goal point and to the value in the backward array at the start point.

In order to prevent excessive wandering in the path (caused by the approximate representation of the real world by the grid) and to provide an easy means of generating reasonably path long segments for steering the rover, all points in the sum array within a given tolerance from the minimum are detected, and the result is thinned to a path with a thickness of one grid point. Then a segmented path is generated suitable for sending to the rover steering program. At present, this consists of straight-line segments. These are generated by iterative end point fits (Duda and Hart, 1973) to the thinned path. In this method, a straight line is fit to the two end points, the line can be divided at the point of maximum deviation of the data (here the thinned path) from the straight line, and the process repeats recursively on each new segment. The criterion used here for stopping the recursion is that each straight line segment must be entirely within the unthinned acceptable minimum points. In the future, curved segments may be generated, instead of leaving this to the rover steering program, to steer better through narrow passages.

However, the above produces only a tentative path, and at some point along the path we may want to take another look with the vision system and to replan before executing the rest of the path. For this purpose, the probability portion of the cost function $(-\ln(p_t)/b)$ is integrated along the tentative path. When this integrated probability cost exceeds a threshold (or the probability of failure at any grid point reached exceeds a smaller threshold), the path is terminated. (Actually, the path terminates at a specified distance before the threshold is reached, since the rover may not be able to see the ground directly in front of it. Similarly, a minimum distance after which this integration and test start can be specified, in case the rover starts in a poorly known area. Also, a maximum path length can be specified.) Execution of only the portion of the path up to this termination point is actually attempted. (Also, normally the line segment fits actually are done only for this portion.) However, the remaining portion of the tentative path is examined to determine a good direction for the vehicle to look for its next view, since the most likely next path is roughly along this tentative path.

In some scenarios, the goal may be very far from the present rover position. Of course, intermediate goals can be generated from approximate paths that may be sent from Earth, but it may be desirable to have even these outside of the grid that is generated, since this area needs to be small in order to keep the computing time small. If the goal is outside of the grid area, the backward array is initialized by inserting in its border (on the one or two sides facing the goal) the integrated cost from each of these points to the goal, computed by using the Euclidean distance and a point probability of $1/2$ (with a and b as usual).

Although an algorithm such as the above is most suitable for a parallel processor such as an SIMD (single instruction multiple data) machine or a pipelined image processor, for small areas it runs reasonably fast on an ordinary computer. However, on a single processor the computation time increases roughly as the cube of the linear dimensions in terms of the grid spacing, whereas on a parallel machine with one node for each grid point it increases roughly as the first power. One speedup done in the present version of the program, since it runs on a single processor, is to note which grid points do not change during the forward and backward growth, so that no computations need be done for their neighbors on the next iteration.

In the future, more elaborate path planners may be used that take into account the position of each wheel as bumpy terrain is traversed, so that a more accurate traversability analysis can be done. However, a method such as the above using slope and roughness smoothed over approximately the vehicle size can produce a tentative route for refinement by such analysis.

5. Results

Four examples of the use of the methods described above will now be presented. The first two examples use data from previous projects, and involve the combined use of both aerial photographs (simulating a planetary orbiter) and images taken from a vehicle on the ground. The third example uses data from a current project, with images only from a ground vehicle. The last example is from another current project, and it uses only overhead views (simulating the descent of a planetary lander).

Previously, our experiments were performed in natural terrain in the arroyo beside JPL. In order to simulate the data from orbital images that might be used in a future mission, a stereo pair of aerial photographs of the arroyo were used. These were processsed by a stereo program (Gennery, 1989), whose output was manually corrected where needed. The result was a height map with points at approximately one-meter horizontal spacing in most places, with some large gaps, but with no gross errors. This is similar to the kind of data that might be produced by the combined computer and human processing on Earth of orbital images of Mars.

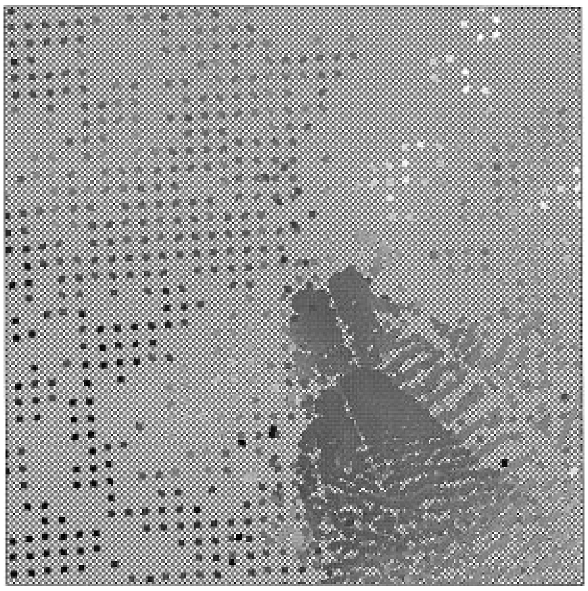


Figure 2. Height data from rover vision (at two positions of large rover) and aerial photographs, shown in a horizontal plane, 32 m $\times$ 32 m. Each data point is shown as a gray-level square, where black $=0$ and white $=3$ m relative elevation.

Figures 2–4 show some results for a large rover that was called “Robby” (Weisbin et al., 1992). It was 3.9 m long and 1.6 m wide, and was fully instrumented. These figures show an example of processing the data collected at two successive positions of this vehicle in the arroyo.

Three stereo pairs of images from the rover cameras covering a panorama of roughly 70° at each position were processed to obtain three-dimensional data, and the results were automatically translated, corrected, and merged with the data from the aerial photographs, by methods previously described (Gennery, 1989). Figure 2 shows the merged height map. The area covered by the figure is 32 m $\times$ 32 m. In the figure, east is to the right, north is up, and height (altitude) is represented by the gray level, which varies linearly upwards from black to white over a 3-m interval. Each point is shown as a 0.3-m square superimposed on a crosshatched pattern. (Included with each data point is

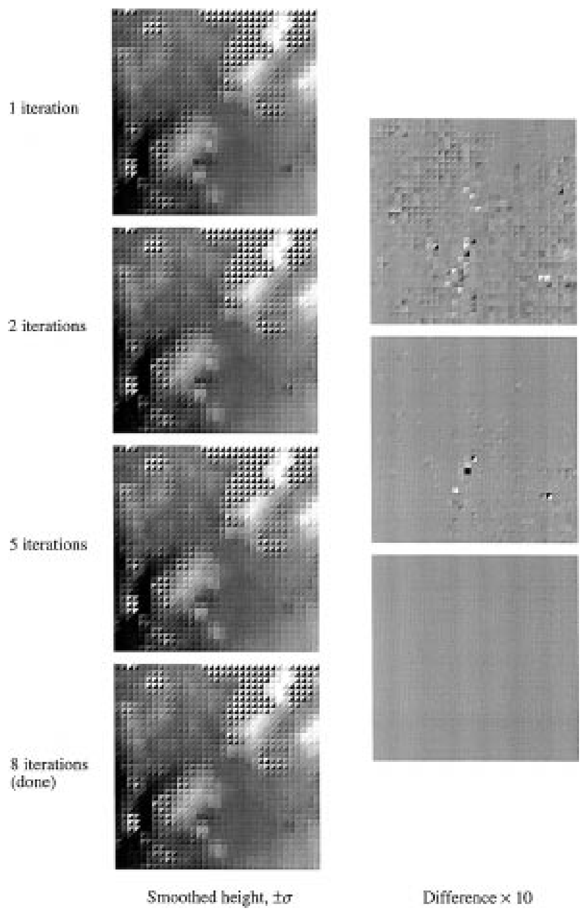


Figure 3. Smoothing and interpolating for large rover example, generating data at 1-m horizontal intervals. The gray scale is as in Fig. 2, except that each square is divided to indicate plus and minus one standard deviation.

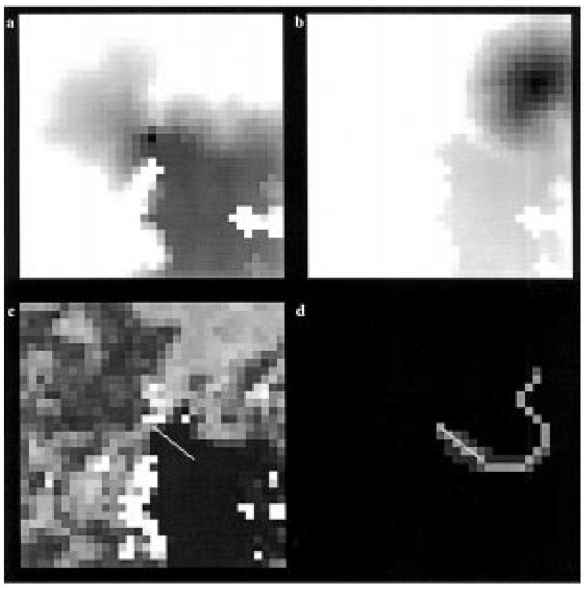


Figure 4. Path planner for large rover example: (a) forward cost integrated from start position (black represents zero cost); (b) backward cost integrated from goal; (c) cost function used in integrations, obtained from data in Figs. 2 and 3, with final path (white line) superimposed; and (d) preliminary path with final path superimposed.

a covariance matrix and a probability, but these are not shown in the figure.) At its second position the rover is near the center of the figure, looking towards the lower right. (In its previous position it was a little more to the upper left, looking in approximately the same direction.) The data from the rover's vision system can be seen as the dense points towards the lower right. The points from the aerial pictures are mostly roughly equally spaced (about a meter apart), with some large gaps. (These points generally are less accurate than those points from the rover that are near to the rover.) Six points at each rover position corresponding to the wheels of the rover are included, since these are known points in contact with the ground.

Figure 3 shows the result of computing the smoothed, interpolated heights from this data using the method described in Section 2. A grid spacing of 1 m was used in generating the smoothed points, over an area $32 \text{ m} \times 32 \text{ m}$ (producing 33 by 33 points). The standard deviation of the Gaussian function for smoothing was 0.5 m. In the figure, each point is plotted as a 1-m square, divided diagonally to indicate the height plus and minus one standard deviation. (If the values are outside of the 3-m height range, they appear as pure black or white, as appropriate.) In the portions where there was no input data, the standard deviations are infinite, so the squares are divided into black and white triangles. In portions of high accuracy, the squares appear uniform. The left column of the figure shows the results after selected iterations, with the bottom one being the final result, and the right column shows the difference between these, with the contrast increased by a factor of 10, and with the zero level shifted to medium gray, so that differences of either polarity can be seen. Since the changes are fairly small, the computations could be limited to one iteration where speed is important and very high accuracy is not needed. This computation (which used 7032 input points to generate 1089 output points) took 0.66 s for the first iteration and 3.27 s total for all iterations, on a Sun Microsystems Sparc Ultra 1 workstation.

The slope and roughness information corresponding to the final heights in Fig. 3 were used as described in Section 3 to compute the cost function shown in the lower-left portion of Fig. 4, where black represents

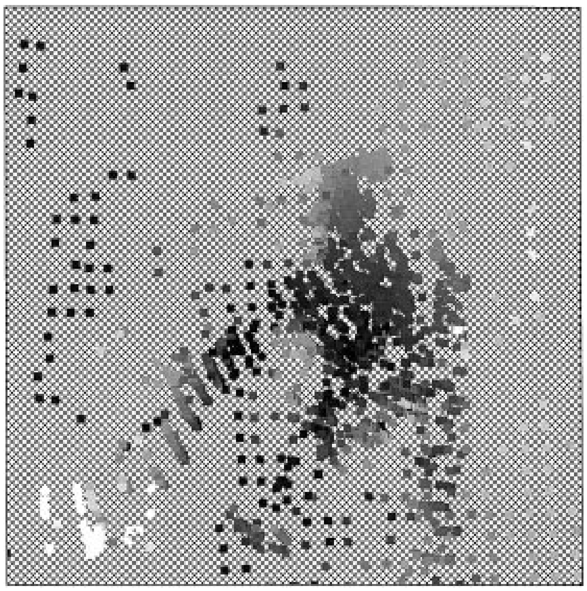


Figure 5. Height data from rover vision (at one position of small rover) and aerial photographs, shown in a horizontal plane, 25 m × 25 m. Each data point is shown as a gray-level square, where black = 0 and white = 2 m relative elevation.

zero cost. This cost function was used in the path planner described in Section 4. The forwardly integrated cost (after the final iteration) is shown in the upper-left portion of Fig. 4 (the start point is the darkest), and the backwardly integrated cost is shown in the upper right (the goal point is the darkest). In the lower right, the preliminary path computed by the path planner is shown with dark gray before thinning and light gray after thinning. The final path to be executed is shown as a thin white line superimposed both on this preliminary path and on the cost function in the lower left portion. (These figures show the same area as Fig. 2.) For this run, the slope threshold was 0.5, the roughness threshold was 0.2 m, the probability tolerance for the minimum was 0.1, the point probability tolerance along the path was 0.01, the integrated probability tolerance was 0.1, and the actual vehicle width of 1.6 m was used. This computation, including reading the data from a file, computing the cost function, performing the search, and generating and writing the display for the figure and other output, took approximately 1.3 s on a Sparc Ultra 1.

Some older experiments used a smaller, simpler vehicle (the "Blue Rover") that was slightly modified from one developed for another project (Holmes et al., 1986). This vehicle was 1.8 m long and 0.8 m wide.

Figures 5–7 show similar data for this vehicle as that shown in Figs. 2–4 for the previous example, except

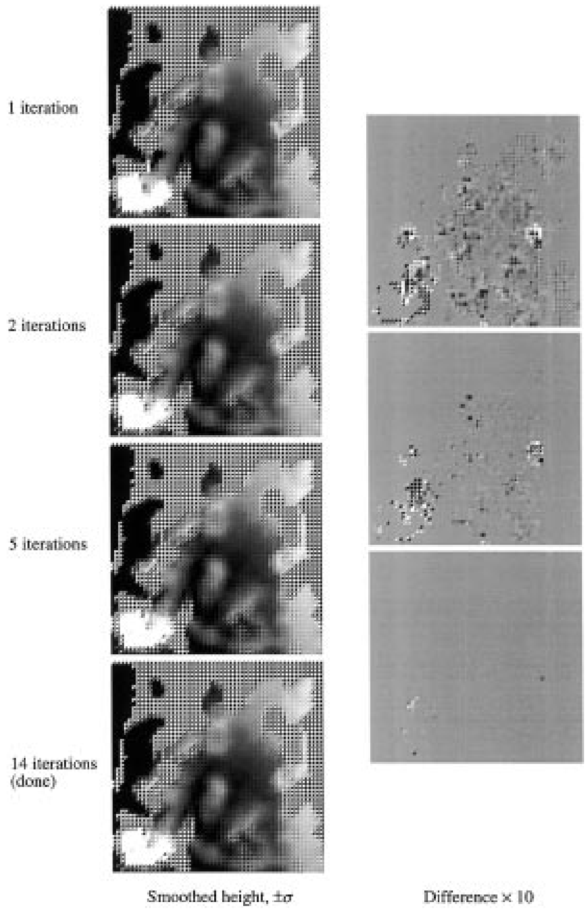


Figure 6. Smoothing and interpolating for small rover example, generating data at 0.5-m horizontal intervals. The gray scale is as in Fig. 5, except that each square is divided to indicate plus and minus one standard deviation.

that these figures cover an area of 25 by 25 m (with a 2 m range in height from black to white), the interpolated and smoothed data is produced at an interval of 0.5 m, and only one rover position was used in this case. The rover is in the upper right portion of the figures and is looking approximately towards the lower left. Figure 5 shows the combined data from the aerial photography and a panorama of three stereo views from one position, Fig. 6 shows the iterations to obtain the smoothed and interpolated data, and Fig. 7 shows the results of the path planner. (The standard deviation of the Gaussian function for smoothing was 0.4 m, and the roughness threshold in the path planner was 0.1 m, because of the smaller vehicle.) The slowly converging high-elevation area in the lower left is a tree, which is a difficult object for stereo vision, especially since it looks so different from the ground and from the air.

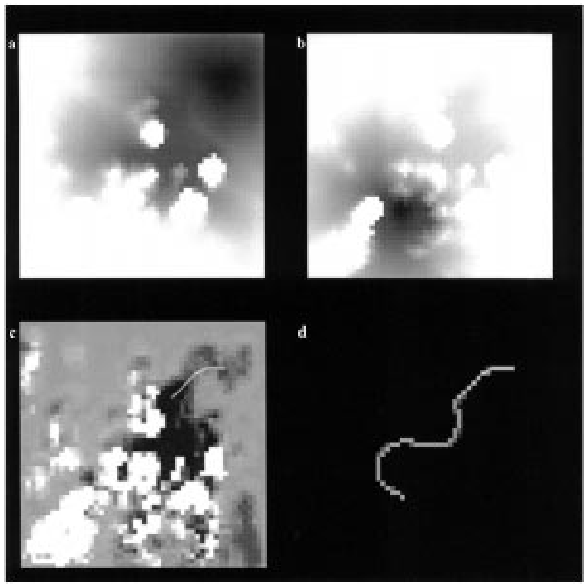


Figure 7. Path planner for small rover example: (a) forward cost integrated from start position (black represents zero cost); (b) backward cost integrated from goal; (c) cost function used in integrations, obtained from data in Figs. 5 and 6, with final path (white line) superimposed; and (d) preliminary path with final path superimposed.

Figure 8 shows a stereo pair taken from a HMMWV ("Humvee") used as an autonomous vehicle in a current research project for the Army (Matthies et al., 1995b). These images were processed by the same stereo and smoothing-interpolating programs used above, and the results are shown as a height map at 1-meter intervals in the horizontal plane in Fig. 9, which covers an area 30 m wide, 35 m in range, and 3 m in height from black to white, with the standard deviations shown as before. Since there is no aerial data in this case, the areas to the sides of the camera view have no data, and therefore have infinite standard deviation. These therefore are areas to be avoided, along with the steep or rough areas along the edge of the road, such as the tree and the bush.

The path planner was run on the data in Fig. 9. The vehicle starting position was at the center of the bottom edge of the array in the figure, and the goal was 1000 m straight ahead of the vehicle. (Therefore, all points along the top edge of the array in the figure have almost equal desirability as a goal for the path planner.) The computed path is overlaid on the height display in Fig. 9 as a white line. It is also projected into the images in Fig. 8, by using the smoothed height data (at the ends of the straight segments) for the height dimension. The portion that seems to be safe to execute is indicated by a solid line. At the end of this line, the vehicle should take another look before proceeding. The rest of the path (the tentative portion) is indicated by a dashed line. For this run, the slope threshold was 0.25 and the roughness threshold was 0.1 m. Varying the slope threshold from 0.15 to 0.5 and the roughness threshold from 0.07 to 0.5 m made no appreciable difference in the planned path, except for how much of it was deemed to be executable.

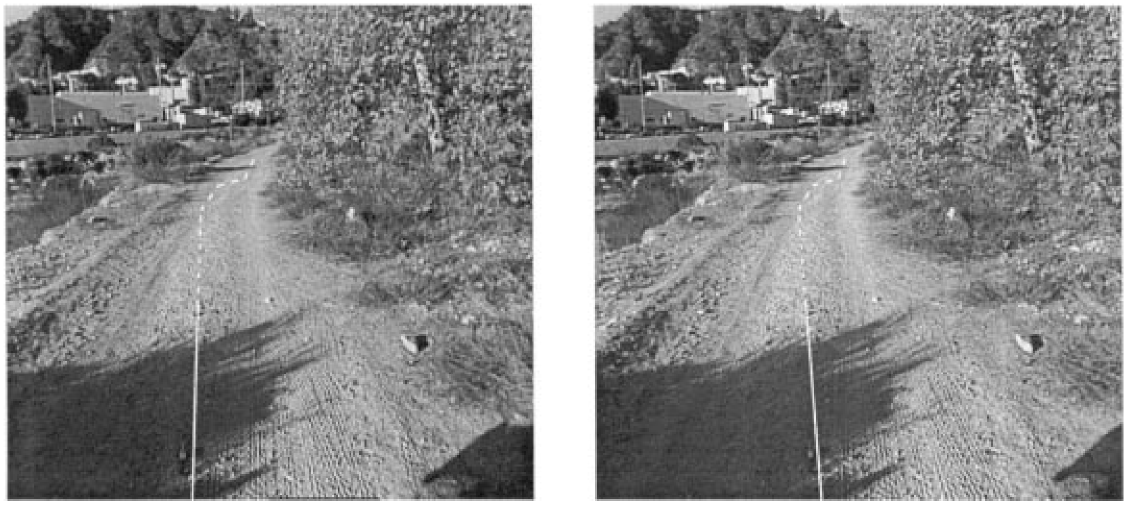


Figure 8. Stereo pair from HMMWV. (Left image is on the right.) The planned path is projected as a white line, solid for executable and dashed for tentative.

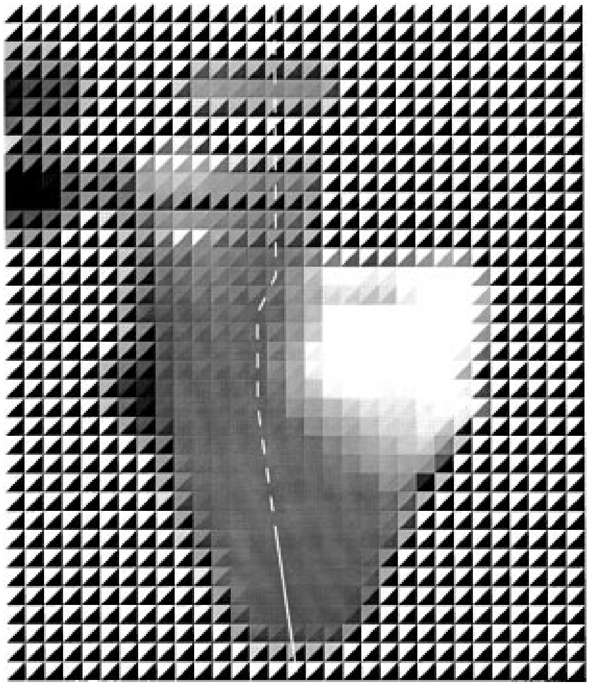


Figure 9. Smoothed and interpolated height map from HMMWV images in Fig. 8, shown in a horizontal plane, where black = 0 and white = 3 m relative elevation. Each point is shown as a 1-m square, divided diagonally to indicate height plus and minus one standard deviation. The planned path (from bottom to top) is superimposed as a white line, solid for executable and dashed for tentative.

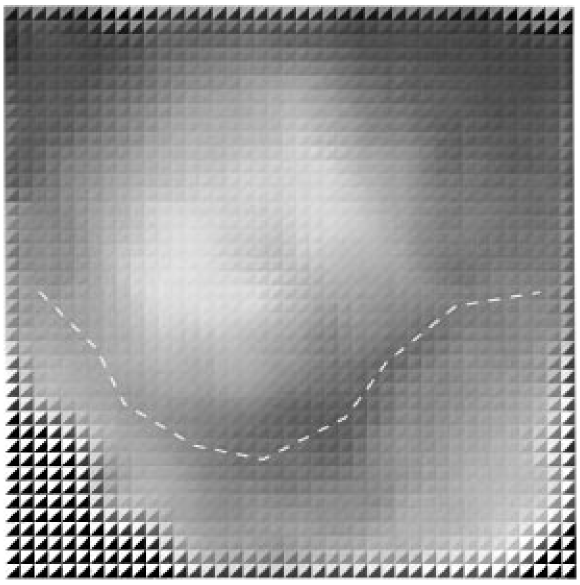


Figure 10. Smoothed and interpolated height map from Mars Yard, shown in a horizontal plane, where black = 0 and white = 2 m relative elevation. Each point is shown as a 0.25-m square, divided diagonally to indicate height plus and minus one standard deviation. The tentative planned path (from left to right) is superimposed as a dashed white line.

Figure 10 shows the results from images that were taken in the JPL Mars Yard, which is used for testing rovers (Matthies et al., 1997). A pair of images were taken looking straight down, from two positions that were almost in a vertical line, in order to simulate images that may be taken by the descent stage of a vehicle landing on Mars or some other astronomical body. The upper image was taken from about 14 m above the ground, and the lower image was taken from about 8 m above the ground. These images were processed as a stereo pair, and the smoothed and interpolated heights are shown in the figure as before, with the brightness values covering a range of 2 m in height, over an area 10 m $\times$ 10 m in the horizontal plane. The standard deviations are fairly large in the center, because it is on the stereo baseline (without the smoothing, they would be even larger), and they are infinite near some of the edges, where there is no data. The path planner was run on this data, with the starting point near the left edge and the goal near the right edge, and the result is shown by the white line in the figure. It can be seen that the path skirts around the mound in the center. However, it is shown all as a dashed line, since the accuracy of the data was not sufficient to have high confidence in the path. It should be considered as a long-range tentative path that a vehicle on the ground would refine with its own local views as it progressed. The scale of this data is such that a very small vehicle was assumed (a few centimeters); however, the results scale up to what might be obtained from a descent stage at higher altitudes.

6. Summary and Conclusions

A method has been presented of computing smoothed height, slope, and roughness from three-dimensional data such as that from a stereo vision system. By an interpolation mechanism inherent in the smoothing process, the results are produced at equally-spaced points in a horizontal plane, so as to be convenient for use by a path planner for a vehicle. The error propagation included in the algorithm produces accuracy estimates of these quantities. These accuracy estimates are very important in deciding whether the computed quantities indicate a safely traversable region, especially when trying to plan a path that extends a considerable distance from the location at which images were taken. For example, with stereo, the error in the distance measurements varies approximately with the square of the distance, and the error propagation indicates this degrading accuracy in its results. Also, at large distances, the data from stereo or a scanning laser rangefinder will tend to become sparser when expressed in a horizontal plane, and this sparseness of data results in the computed slope and roughness having poor accuracy, which is reflected in the resulting variance estimates.

The above computation of smoothed height, slope, and roughness is an iterative process, since the weights used in the adjustment depend on the adjusted values. However, the first iteration, using the nominal weights, usually produces results not much different from the final iteration. Therefore, the computations can be limited to one iteration when speed is important. On the other hand, other vision processing, such as stereo, may take even more time than the time taken by these computations, so in many cases the full iterative process might as well be used, for more accuracy.

The above quantities can be used to compute a cost function that a path planner would use to indicate the undesirability of driving through a given region. The accuracy estimates can be used to compute the probability that the maximum slope or maximum roughness capability of the vehicle would be exceeded. The energy and time expended in traveling a given distance, depending on the roughness and the steepness of a climb or descent, also can be extracted from the slope and roughness data and the vehicle specifications. Both of these types of information, and information from other sources indicating the desirability or undesirability of driving through a given region, can be combined to produce a cost function that includes all of the available information.

A path planner that uses a cost function such as the above (at present based on only probability and distance), has been described. This path planner computes the optimum path (minimum cost) of all possible paths that proceed from a given start point to a given goal on a horizontal grid. The equally-spaced grid on which these computations are done corresponds to that from the slope and roughness computation above. This optimum (on the grid) path, composed of "chess king" (eight-neighbor) moves on the grid, is refined into a more suitable (and sometimes more accurate) form for vehicle steering commands (at present, reasonably long straight-line segments, but possibly curved segments in the future). The path planner terminates the path when the portion of the cost due to uncertainty becomes too high. At that point, more images should be taken and the rest of the path replanned. However, a tentative path is computed beyond this point (all the way to the goal, if possible), which may be useful in subsequent planning or in directing the next view.

The amount of computation used by this path planner usually increases roughly as the cube of the linear dimensions of the grid. When it is desired to plan a path at high resolution over a very large area, a hierarchical coarse-to-fine approach could be used to save time (Moravec, 1988), in which a path planned at low resolution is successively refined by replanning at higher resolutions over smaller areas. (Of course, then there is no guarantee that the globally optimum path would be found.) However, for typical planetary rover applications, this situation would not occur, because the rover cannot see very far, and thus a high-resolution map is not available over a large area. The only likely similar situations are the following: the case where a long coarse path is planned from orbital or descent images and then is refined over the short distance that the rover can see from each position, and the case where a planned path is refined by considering individual wheel placements, as discussed in the next paragraph.

The results from the path planner described herein are suitable for driving a vehicle over regions where the roughness is small and where no sharp turns in narrow passageways are needed. However, if it is desired to drive through more difficult regions, a more detailed analysis may be necessary. In this case, first a tentative path could be computed by producing slope and roughness with smoothing comparable to the size of the vehicle, and using these in the above path planner. Then height could be computed with very little smoothing (comparable to the size of the wheels or less). The tentative path then could be verified and refined by a simulation of placing the wheels (or legs) in various positions along the tentative path, as limited by the turning ability of the vehicle. (Of course, this would work only if the original vision data has sufficient resolution, so it may require more frequent image taking, but the accuracy estimates from the smoothing computation indicates whether or not it does.) Such a process is beyond the scope of this paper. It may be attempted in future work, and certain aspects of such an approach have been included in work by others (e.g., Gat et al., 1990), but in any case the computation of smoothed height, slope, and roughness and the path planner, as described in this paper, would produce suitable inputs for this additional computation.

Acknowledgments

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and was partially funded through the Director's Discretionary Fund. Participants in the older rover experiments included Andrew Mishkin, Brian Wilcox, Brian Cooper, Steven Katzmann, Keith Lay, Tam Nguyen, Greg De Wit, and Jonathan Cameron. The newer data was supplied by Todd Litwin and Larry Matthies.

References

Brooks, R. 1982. Solving the find-path problem by representing free space as generalized cones. Massachusetts Institute of Technology, Cambridge, MA, AI Memo 674.

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23.

Doshi, R.S., Lam, R.K., and White, J.E. 1988. Region based route planning: Multi-abstraction route planning based on intermediate level vision processing. In *Proc. Sensor Fusion: Spatial Reasoning and Scene Interpretation*, SPIE, Cambridge, MA.

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

Feder, H.J.S. and Slotine, J.-J.E. 1997. Real-time path planning using harmonic potentials. In *Proc. IEEE International Conference on Robotics and Automation*, Albuquerque, NM, pp. 874–881.

Gat, E., Slack, M.G., Miller, D.P., and Firby, R.J. 1990. Path planning and execution monitoring for a planetary rover. In *Proc. IEEE International Conference on Robotics and Automation*, Cincinnati, OH, pp. 20–25.

Gennery, D.B. 1979. Object detection and measurement using stereo vision. In *Proc. Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, pp. 320–327.

Gennery, D.B. 1980. Modelling the environment of an exploring vehicle by means of stereo vision, AIM-339 (Computer Science Dept. Report STAN-CS-80-805), Stanford University.

Gennery, D.B. 1989. Visual terrain matching for a mars rover. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, pp. 483–491.

Gennery, D.B. 1991. Planning the route of a robotic land vehicle. *NASA Tech Briefs*, 15(3).

Giralt, G., Sobek, R., and Chatila, R. 1979. A Multi-level planning and navigation system for a mobile robot: A first approach to hilare. In *Proc. Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, pp. 335–337.

Hebert, M.H., Kanade, T., and Kweon, I. 1988. 3-D vision techniques for autonomous vehicles, CMU-RI-TR-88-12, Carnegie Mellon University, Pittsburgh, PA.

Holmes, K.G., Wilcox, B.H., Cameron, J.M., Cooper, B.K., and Salo, R.A. 1986. Robotic vehicle computer aided remote driving, Vol. 1, JPL internal report D-3282, Jet Propulsion Laboratory, Pasadena, CA.

Mankins, J.C. (Ed.) 1987. *Proc. Mars Rover Technology Workshop*. JPL internal report D-4788, Jet Propulsion Laboratory, Pasadena, CA.

Matthies, L., Gat, E., Harrison, R., Wilcox, B., Volpe, R., and Litwin, T. 1995a. Mars microrover navigation: Performance evaluation and enhancement. *Autonomous Robots*, 2:291–311.

Matthies, L., Litwin, T., and Kelly, A. 1995b. Obstacle detection for unmanned ground vehicles: A progress report. *International Symposium of Robotics Research*, Munich, Germany.

Matthies, L., Olson, C., Tharp, G., and Laubach, S. 1997. Visual localization methods for mars rovers using lander, rover, and descent imagery. *International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS)*, Tokyo, Japan.

Mikhail, E.M. (with contributions by F. Ackermann) 1976. *Observations and Least Squares*, Harper and Row.

Moravec, H.P. 1980. Obstacle avoidance and navigation in the real world by a seeing robot rover, AIM-340 (Computer Science Dept. Report STAN-CS-80-813), Stanford University.

Moravec, H.P. 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74.

Nilsson, N.J. 1971. *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill.

Randolph, J.R. (Ed.) 1986. Mars rover 1996 mission concept, JPL internal report D-3922, Jet Propulsion Laboratory, Pasadena, CA.

Ratering, S. and Gini, M. 1995. Robot navigation in a known environment with unknown moving obstacles. *Autonomous Robots*, 1:149–165.

Roberts, B. and Bhanu, B. 1992. Inertial navigation sensor integrated motion analysis for autonomous vehicle navigation. *Journal of Robotic Systems*, 9:817–842.

Shirley, D. and Matijevic, J. 1995. Mars pathfinder microrover. *Autonomous Robots*, 2:283–289.

Slack, M.G. and Miller, D.P. 1987. Path planning through time and space in dynamic domains. In *Proc. Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, pp. 1067–1070.

Stentz, A. and Hebert, M. 1995. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2:127–145.

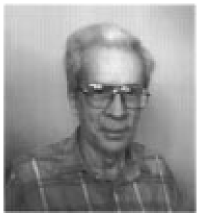
Stephens, M.J., Blissett, R.J., Charnley, D., Sparks, E.P., and Pike, J.M. 1989. Outdoor vehicle navigation using passive 3D vision. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, pp. 556–562.

Thorpe, C.E. 1984. Path relaxation: Path planning for a mobile robot. In *Proc. National Conference on Artificial Intelligence* (AAAI-84), Austin, TX, pp. 318–321.

Weisbin, C.R., Montenerlo, M., and Whittaker, W. 1992. Evolving directions in NASA's planetary rover requirements and technology. In *Missions, Technologies and Design of Planetary Mobile Vehicles*, Centre National d'Etudes Spatiales, France.

Wilcox, B.H. and Gennery, D.B. 1987. A mars rover for the 1990's. *Journal of the British Interplanetary Society*, 40:483–488.

Witkowski, C.M. 1983. A parallel processor algorithm for robot route planning. In *Proc. Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, pp. 827–829.



Donald B. Gennery has conducted research at JPL since 1980 in the fields of computer vision, robotics, and parallel computing. He has developed techniques for stereo vision, camera calibration, tracking moving objects, and rover navigation. In previous work for RCA at Cape Canaveral he developed techniques for digital filtering, image processing, and shipboard navigation systems. He has an M.S. degree in Physics from the Florida Institute of Technology and a Ph.D. degree in Computer Science from Stanford University.