JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 25, No. 6, November–December 2002

Lidar-Based Hazard Avoidance for Safe Landing on Mars

Andrew E. Johnson,* Allan R. Klumpp,† James B. Collier,‡ and Aron A. Wolf§
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109

Hazard avoidance is a key technology for landing large payloads safely on the surface of Mars. During hazard avoidance, sensors and computing onboard the lander are utilized to detect hazards in the landing zone, autonomously select a location free of hazards, and then generate a trajectory that retargets the lander to the safe landing site. Algorithms are described for scanning lidar (light detection and ranging)-based hazard detection, safe site selection, and powered landing guidance designed for Mars landing. The performance of these algorithms is quantified using a closed-loop simulation of hazard avoidance, which includes a synthetic Martian terrain generator, a scanning lidar model, and the required hazard avoidance and powered landing guidance algorithms. Preliminary results show that the proposed hazard avoidance algorithms are effective at detecting hazards and guiding the lander to a safe landing site.

Introduction

SURFACE hazards in the form of rocks and craters exist over the entire surface of Mars.[1] To guarantee safe landing, Mars landers are designed to be robust to surface hazards. Safe landing can be achieved by either of two design approaches. One is hazard tolerance, in which the spacecraft is designed to withstand impact with whatever terrain is expected in the targeted landing ellipse; airbags (employed by Mars Pathfinder) are an example. For the large landers currently under study for the Mars 2009 and 2011 launch opportunities, hazard tolerance can only be achieved by prohibitively massive ruggedized landing systems. The second approach (and the focus of this work) is hazard avoidance in which the spacecraft uses onboard sensors to detect hazards in the landing zone, selects an alternate landing site, and then maneuvers to the new site. One particularly promising sensor for hazard avoidance is scanning lidar (light detection and ranging). This paper describes algorithms for scanning lidar-based hazard avoidance and powered landing guidance for retargeting.

Design of a system for hazard avoidance requires trade studies to investigate sensor requirements and mission design. Analysis of these system trades is greatly facilitated by simulation. An integrated simulation tool has been developed and is described here, along with some illustrative results from a preliminary trade study. Our simulation contains four modules that interact according to the block diagram shown in Fig. 1.

Mars Terrain

Topographic terrain data are needed to model the hazards (rocks, cliffs, craters) likely to be encountered during landing. A large database of high-resolution Martian terrain is not available, and so a method for synthetically generating realistic terrain is needed. Because the results generated from the simulation will be useful only if the underlying terrain is realistic, we take advantage of an existing method for generating Martian terrain that is based on geophysical processes.[2,3] The terrain is generated by populating an initial surface, which comes from coarse orbiter topography data, with rocks and craters in a way that models the aging of the Martian surface. Once generated, the terrain is interrogated by the lidar model to generate measurements of surface topography. A reduced resolution rendering of a nominal Martian terrain is shown in Fig. 2a.

Scanning Lidar

We have designed our hazard avoidance algorithms to use data from a scanning lidar because it is the terrain-sensing instrument base lined for the large Mars lander mission currently under study at the Jet Propulsion Laboratory (JPL). A scanning lidar senses the three-dimensional topography within its field of view by raster scanning a pulsed laser beam across the targeted surface. When the time of flight of the laser pulses reflected from the surface is measured, the range to the surface can be determined for each scan. When combined with measurements of the angular position of a mirror that directs the scan, a three-dimensional point or sample can be generated for each laser pulse. The output of the lidar is a cloud of three-dimensional points that convey the topography of the scanned surface. Assumed parameters for a landing lidar are a 10×10 deg field of view with 10,000 samples scanned in 1 s and a maximum range of 2 km with a range resolution of 2 cm. We have built a model of a lidar into our simulation that incorporates pointing errors and range sensing errors due to measurement noise, as well as pulse stretching by the scanned terrain. The lidar model uses efficient ray tracing algorithms from computer graphics to generate 10,000 samples in less than 1 s, and so it can be used for real-time simulation of landing. The samples generated by the lidar model are output to the hazard detection and avoidance algorithms that compute safe landing sites.

Hazard Avoidance

A particular patch of terrain presents a hazard to the spacecraft during landing if the slope of the patch is too steep or the patch contains rocks or other protuberances that are taller than a certain terrain height. To determine quantitatively if a patch is hazardous, the slope and terrain variation over the patch must be measured. We have developed algorithms that estimate the location of surface hazards given scanning lidar data. These algorithms build an elevation map from lidar samples, estimate local slope and roughness using the elevation map, and then determine areas that exceed constraints on surface slope and roughness given the footprint of the lander. Images of nominal parameter maps are given in Fig. 2b. Of the remaining safe places to land, the hazard avoidance algorithm selects the location with the minimum slope and roughness. This new safe landing site is then passed to the powered landing guidance module that uses it to compute a trajectory to the new landing site. A safe landing map with previous and new landing sites selected is shown in Fig. 2c.

Received 10 September 2001; revision received 23 May 2002; accepted for publication 24 May 2002.

*Senior Member of Technical Staff, Section 348, Mail-Stop 125-209, 4800 Oak Grove Drive; aej@jpl.nasa.gov. Member AIAA.

†Senior Systems Engineer, Section 341, Mail-Stop 198-138, 4800 Oak Grove Drive; Allan.R.Klummp@jpl.nasa.gov. Member AIAA.

‡Senior Member of Technical Staff, Section 367, Mail-Stop 126-104, 4800 Oak Grove Drive; James.B.Collier@jpl.nasa.gov.

§Entry, Descent and Landing/Aero Applications Group Supervisor, Section 312, Mail-Stop 301-140L, 4800 Oak Grove Drive; Aron.A.Wolf@jpl.nasa.gov.

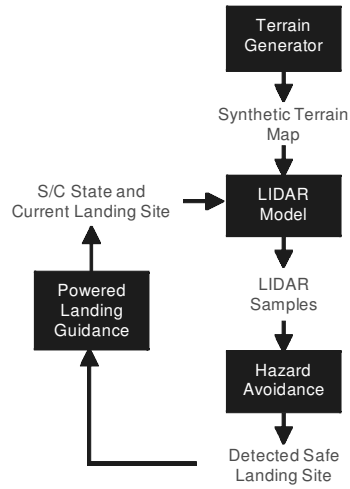

Fig. 1 Block diagram for simulation.

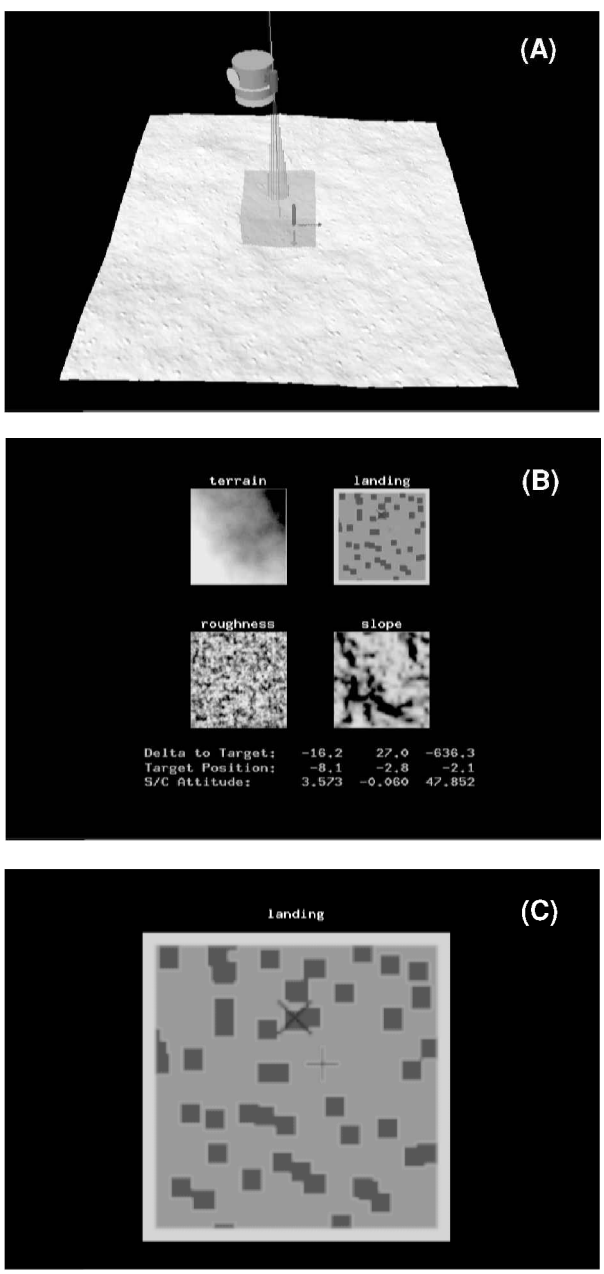

Fig. 2 Hazard avoidance simulation visualization: a) terrain and lidar scan, b) parameter maps used to detect safe landing sites, and c) safe landing site map with hazards shown in dark gray, safe zones shown in light gray, previous landing sites shown as a black *X*, and the selected landing site shown as a +.

Powered Landing Guidance

We use an algorithm for powered landing guidance that can retarget at any time to a new point specified by hazard avoidance. Our guidance algorithm, adapted from that flown on Apollo, transfers the lander from any current state (position and velocity) to touchdown in two phases, called approach and vertical descent. Each phase has a target point where the position, velocity, and acceleration are all specified. The approach-phase target is 5 or 10 m directly above the landing site, and the vertical-descent-phase target is on the surface. Specifying zero for the horizontal components of target velocity and acceleration causes the lander to arrive at each target point with zero transverse velocity and in an erect attitude, regardless of any maneuvering en route to avoid hazards. The vertical components of approach-phase target velocity and acceleration are chosen to provide a fuel-efficient transfer with a safe thrust margin and a vertical rate at the target that is the same or close to the constant value flown in the vertical-descent phase. In the vertical-descent phase, the lander descends at around 1 m/s until the engines are shut off upon or just before contacting the surface.

Initially the lander is traveling along a predetermined trajectory. Each time a lidar scan is taken, hazards are detected and a (possibly) new safe landing site is selected. This site is then passed to the guidance module, which computes a new trajectory to the desired landing site. This process is repeated until the lander has landed safely on the surface. The next section describes the (minimal) related work in the area of hazard detection and avoidance, and the rest of the paper describes in detail our algorithms and simulation for lidar-based hazard avoidance.

Related Work

All unmanned planetary landing systems to date have followed the mechanical hazard tolerance approach to safe landing. Only recently has there been interest in autonomous hazard detection and avoidance technology, and so related work is limited to preliminary studies on the viability of different approaches to safe landing. In the early 1990s Charles Stark Draper Laboratory conducted a study to design a hazard detection and avoidance system for NASA Johnson Space Center.[4] Their design for a hazard detection and avoidance system utilized a passive imager and a laser scanner to detect hazards in a two-stage approach. In their approach, candidate landing sites are first selected with a passive imager by finding regions of minimum intensity variation. These sites are then scanned with a higher resolution laser sensor to find the landing site with minimum hazards. To motivate the need for safe landing technology, Charles Stark Draper Laboratory also conducted a study of the Viking I and II landing sites that showed that the probability of landing failure, given a three-legged lander, was as high as 20% (Ref. 5).

The system presented in this paper utilizes the same basic concept of safe landing as that presented in the Charles Stark Draper Laboratory report, but the components used in our simulation are more realistic, which results in a more complete simulation. In particular, input into our algorithm are the terrain maps generated by up-to-date methods for generating synthetic Martian terrains, the lidar model used in our simulation models the range detection electronics used in a real sensor, our hazard detection and avoidance algorithms measure slope and roughness independently (as opposed to a coupled adjacent pixel measurements), our hazard avoidance algorithms find all safe landing sites in the terrain, and finally the loop is closed around sensing by using an adaptive guidance algorithm that recomputes trajectories after each retargeting.

A process for generating hazard sensing and avoidance requirements has been developed by Halbrook et al.[6] Their study showed that a there is a tradeoff between hazard tolerance and hazard avoidance. In their assessment of hazard detection sensors, only scanning lidar/ladar sensors were found to be effective at detecting both roughness and slope hazards. This assessment supports our choice of a lidar as the hazard detection sensor used in our simulation. Although scanning lidars are very effective hazard detection sensors, they do have limitations in terms of maximum range and spatial resolution when compared to passive imagers. Recent work has shown that is it possible to efficiently assess landing site safety in terms of surface roughness and slope using a passive imager and onboard

processing of pairs of images.[7] Given this new capability, future studies are needed to compare the hazard detection performance of scanning lidar and passive imagers.

Synthetic Terrain Map Generation

A requirement for any landing simulation is a physically accurate model of the landing surface. This model must contain enough fidelity that the sensor models that interact with the surface can produce measurements that are realistic. The simulation described in this paper uses a sophisticated environment for modeling Martian terrain that has been developed at JPL over the last decade.[2,3]

The terrain generator software builds Martian surfaces by sequentially applying geological processes to the surface. These processes are realizations of the models used by planetary geologists to describe the surface of Mars. Of particular interest to safe landing is the statistical model for rock size and density. In our simulation, the user has control over the rock density parameter, and so terrain of varying difficulty for landing can be generated.

The terrain generator begins with an initial coarse surface. Fractal surface generation is then used to fill in the terrain. This fractal surface is then acted on by a cratering process and a rock generation process. These processes can be repeated to generate a surface that has the appearance of one that has been created over time. The user has complete control over the terrain generation parameters, so that surfaces of many types can be generated.

The Martian surface terrain generator has been implemented on a 128-node SGI supercomputer. The interface to the terrain generator is a terrain server. The safe landing simulation requests a piece of terrain of a particular size, resolution, and rock density from the terrain server. The terrain server then partitions the generation of the terrain into multiple parts and passes the computationally intensive terrain generation off to multiple processors on the supercomputer. The terrain pieces are generated, and the terrain server puts them together and sends the complete terrain to the simulation. An example of a synthetically generated terrain is given in Fig. 3.

Lidar Model

The safe landing simulation requires a model of a scanning lidar so that realistic lidar measurements can be generated from synthetic terrain. This model should also be computationally efficient so that it can be used in real-time simulations. To this end, we have developed an efficient high-fidelity model that emulates the physical processes within a time-of-flight dual-axis gimbaled mirror lidar.

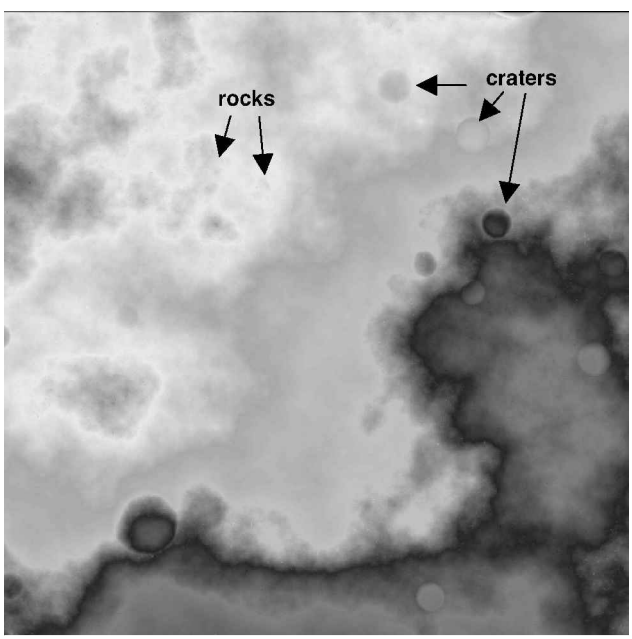

Fig. 3 Example synthetic terrain map populated with craters and rocks (image intensity scaled from high to low).

Principal of Operation

A lidar, when pointed at a surface, measures the range to this surface using laser light. A common range measurement principal used in long-range lidar is time of flight; a laser pulse is emitted, the pulse is reflected from the surface, and the time of the returning pulse is recorded. This time is then converted to a range measurement using the speed of light. Because the laser beam has a finite divergence (angular width), the laser beam will be reflected from a patch of the surface, and a continuum of ranges will result (pulse spreading). The actual range measured by the lidar will be the range at which the integrated return light energy passes a threshold defined in the detector electronics. Usually this threshold is defined in terms of the signal to noise of the measurement.

A scanning lidar emits a continuous stream of laser pulses, and optics are used to sweep the laser beam across a scene. Often single- or dual-axis gimbaled mirrors are used to direct the laser beam. In this case, a scanning lidar measurement, called a sample, consists of the detected range to the surface and the measured angular position of the mirror(s) when the laser pulse was emitted. Mirror encoders are not perfect, so that noise will be introduced into the angle measurements.

A scan pattern defines the angular position of each sample. Modifiable parameters of a scan pattern are type of pattern, for example, raster or spiral, the angular spacing between samples, and the field of view of the scan.

A parameterized function of mirror kinematics is used to convert each sample into a three-dimensional point in a Cartesian coordinate system attached to the lidar. The result is a set of three-dimensional points that convey that shape of the scene being scanned.

Measurement noise will be introduced into the three-dimensional points generated from lidar samples in multiple ways: range error from the integration of ranges within the footprint of the laser and range detector noise, mirror angle measurement noise, and errors in mirror kinematics parameters. A high-fidelity model of a lidar should take all of these noise sources into account. Currently JPL is building a time-of-flight dual-axis gimbaled mirror lidar. For this reason, a model of such a sensor has been developed for the simulation described in this paper. First, the efficient model used to generate range measurements given a lidar beam direction will be described. This will be followed by a description of the model used for laser beam scanning.

Range Measurement Model

Input into the range measurement model is a terrain map, a three-dimensional ray (origin and direction) describing the true direction of the laser pulse impinging on the terrain and the divergence of the laser pulse.

To account for the divergence of the laser, the pulse is modeled as bundle of rays centered on the pointing direction of the laser pulse (Fig. 4). A portion of the energy in the laser pulse is given to each ray, and each of the rays is intersected with the terrain map (described subsequently) to get a range. To model the integration of laser energy occurring in the lidar detector, the energies of the rays are added in order of range (closest to farthest). When the cumulative energy exceeds a threshold, the range at that energy is the ideal detected range. Detector noise is introduced by adding a

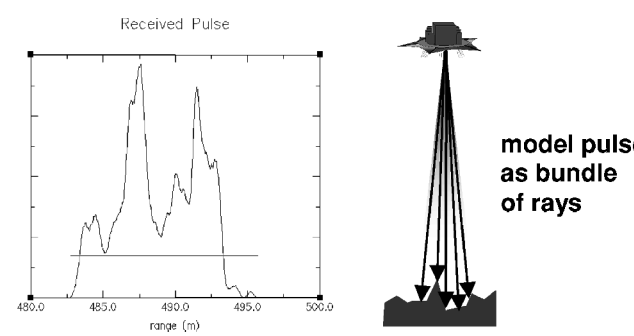

Fig. 4 When laser pulse is modeled as bundle of rays, effect of pulse stretching can be simulated.

Gaussian distributed range error to the ideal range, and this is the output of the range measurement model.

The most computationally expensive portion of the lidar model is ray tracing: intersecting a ray with the synthetic surface to determine range. To alleviate this problem, an efficient ray tracing algorithm has been developed for ray/terrain map intersection. Let a ray be defined as $\boldsymbol{r}(t) = [r_x(t), r_y(t), r_z(t)] = \boldsymbol{a} + \boldsymbol{b}t$ with origin $\boldsymbol{a} = (a_x, a_y, a_z)$ and direction $\boldsymbol{b} = (b_x, b_y, b_z)$, $\|b\| = 1$. First, the maximum $z_{\max}$ and minimum elevation $z_{\min}$ in the terrain map are determined before ray tracing. Then for a given ray, the intersection of the ray with the maximum and minimum elevation planes are given by $\boldsymbol{r}(t_{\max})$ and $\boldsymbol{r}(t_{\min})$ where

$$t_{\max} = (z_{\max} - a_z)/b_z, \qquad t_{\min} = (z_{\min} - a_z)/b_z$$

Suppose the synthetic terrain map $T(r, c)$ has width H and sampling between grid cells of S; then a three-dimensional point $\boldsymbol{x} = (x, y, z)$ can be projected vertically into the grid cell (r, c) in the terrain map using the operator $P(\boldsymbol{x})$,

$$P(\boldsymbol{x}) = P(x, y, z) = (r, c) = (y/S + H/2, x/S + H/2)$$

As shown in Fig. 5, the intersection of the $\boldsymbol{r}(t)$ with the terrain map must occur along the line segment in the two-dimensional row/column space of the terrain map between

$$P[\boldsymbol{r}(t_{\max})] = (r_{\max}, c_{\max}), \qquad P[\boldsymbol{r}(t_{\min})] = (r_{\min}, c_{\min})$$

To find the intersection, a search along this line segment and ray is used. The search starts at the grid cell $(r_s, c_s) = (r_{\max}, c_{\max})$ and ray position $\boldsymbol{r}(t_s) = \boldsymbol{r}(t_{\max})$, where $T(r_s, c_s) \le r_z(t_s)$. The elevation at each grid cell $T(r_s, c_s)$ is compared to the ray elevation $r_z(t_s)$ at that grid cell until the grid cell where $T(r_s, c_s) > r_z(t_s)$ is found. At this transition, the ray has passed though the terrain map, and so the intersection has been found. Linear interpolation between the intersection and previous ray points and terrain elevations is used to get a subgrid cell estimation of terrain map elevation z_i at the intersection, and z_i is converted into a range ρ using

$$\rho = (z_i - a_z)/b_z$$

The procedure for ray intersection is used to get a range for all rays in the bundle describing the laser pulse. The energies associated with the rays are summed from closest to farthest, and when the cumulative energy exceeds a threshold, that range is the error-free range for the pulse ρ_n. Next, a Gaussian error $G(\sigma_\rho)$ is added to ρ_n to simulate detector noise. This range is then discretized to the resolution of the range detection electronics δ_ρ to get the range measurement ρ.

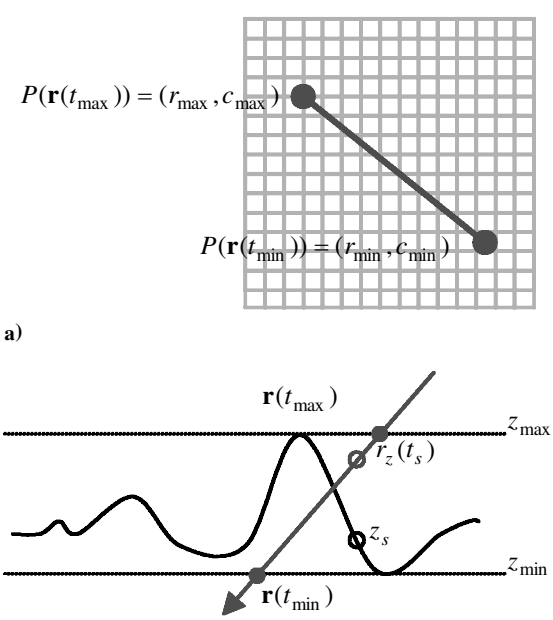

Fig. 5 Intersection of a ray with a terrain map: a) ray intersects a terrain map along a segment and b) elevation view of the search for the ray/terrain map intersection.

Scanning Model

The scanning model is fairly simple because it models the pointing of the laser during scanning using a two-axis gimbaled mirror. The angles of the mirror are (θ, ϕ). The scan model has parameters for field of view f, resolution n (number of samples within field of view), and scan pattern. Currently the model only supports raster scans.

For a raster scan, the field of view and number of samples within the field of view create an array of equally spaced angles (θ_n, ϕ_n) that define the direction of the laser beam for each sample. Because pointing of the mirror is not perfect, the actual mirror angles will not be exactly the same as those defined by the scan pattern. This effect is simulated in the scanning model by adding a Gaussian distributed offset $G(\sigma_\theta, \sigma_\phi)$ to each (θ_n, ϕ_n) pair to get $(\theta_{\text{off}}, \phi_{\text{off}})$. Together $(\theta_{\text{off}}, \phi_{\text{off}})$ and the position and attitude of the lidar determine the ray that is used to generate range for each sample. To model sampling of angle measurements by the mirror encoders, (θ_n, ϕ_n) are discretized to the angular resolutions of the encoder $(\delta_\theta, \delta_\phi)$ to get the measured angles (θ, ϕ). The discretized angles and the measured range constitute the lidar sample (ρ, θ, ϕ) output for each pointing angle. The lidar samples for all angles in the scan pattern constitute the output of the lidar model for a single scan.

In summary, the range measurement model has multiple parameters including the divergence of the laser pulse d, the number of rays used to model the pulse, the link analysis parameters of the laser beam used to set the threshold on return laser energy (laser power, pulse width, etc.), the detector error added to the measured range σ_ρ, and the range resolution δ_ρ of the detector. The parameters input into the scanning model are field of view f, number of samples within field of view n, the pointing error distribution $G(\sigma_\theta, \sigma_\phi)$, and the mirror encoder sampling $(\delta_\theta, \delta_\phi)$.

Computational Issues

It takes less than 50 μs to intersect a ray with a terrain map on a 174-MHz R10000 SGI O2 workstation. For a 100×100 sample scan with 1 ray per sample, this takes 0.5 s, which is sufficient for real-time simulation given that the JPL lidar under development is being designed to take a 10,000 sample scan in 1 s. With greater fidelity in the range measurement model, for example, 10 rays per sample, a scan can be generated in 5 s. Obviously, a model of this level of fidelity cannot be run in real time, but it will be useful during offline Monte Carlo simulations.

Hazard Avoidance Algorithms

The inputs to hazard avoidance are lidar samples and a vector describing the geodetic normal of the surface, both in sensor coordinates. Also input are the safe landing parameters: lander diameter, maximum lander incidence angle, and maximum surface roughness. The hazard avoidance algorithms process these inputs to select a safe landing site. Hazard avoidance takes part in three stages: elevation map generation, hazard detection, and safe site selection.

Elevation Map Generation

Elevation map generation is the process by which range samples are projected into a grid to form a two-and-one-half-dimensional surface representation (regridding). Scanning laser rangefinders generally have spherical or perspective projection models. Also, scan patterns are not always regular raster scans; spiral and triangle scans are common when minimizing scanner power. Nonlinear projection models and irregular scan patterns create an irregular sampling of the surface. If the range samples are used directly, a time-consuming hazard detection algorithm that accounts for the irregular spacing between samples is needed. However, by resampling the range samples from each scan to a regular grid in Cartesian space, hazard detection can be accomplished by applying fixed local operators to the resampled grid. Resampling greatly simplifies the underlying

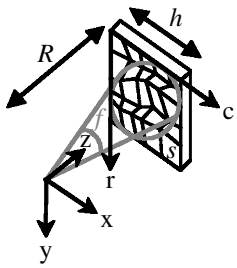

Fig. 6 Sensor and terrain map coordinates.

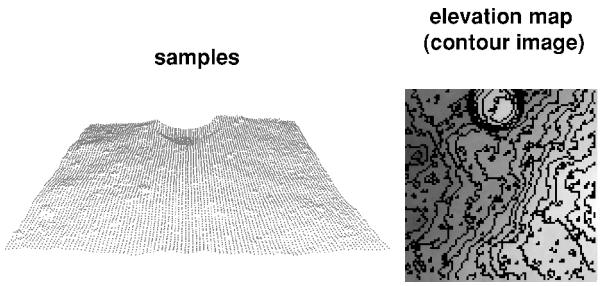

Fig. 7 Elevation map generation.

algorithms and data structures, so that a more efficient algorithm results.

An elevation map is a function $Z(r, c)$ that encodes surface elevation on a regular grid. To generate an elevation map, the horizontal size of each grid cell, s, and horizontal extent, h, of the elevation map must be determined. As shown in Fig. 6, these parameters can be determined from the scanner field of view f, the average of scan samples across the scene n, and the average range to the scene being imaged R. In general we set these parameters as follows:

$$h = 2R\tan(f/2), \qquad s = h/n \tag{1}$$

With these settings, the elevation map will cover roughly the same extent as the scanned data and each grid cell will contain approximately one sample.

Once the elevation map parameters are established, the procedure for elevation map generation is as follows. First, each range sample is converted from scanner angle and range coordinates to Cartesian coordinates (x, y, z). Next, the (x, y) coordinates of the sample are used to determine the floating point coordinates (r, c) that the sample projects to in the grid cell,

$$(r, c) = (y/s + h/2, x/s + h/2) \tag{2}$$

The coordinate relationship between sensor and elevation map coordinates is shown in Fig. 6. In general (r, c) will fall between discrete grid cells; thus, to prevent aliasing, bilinear interpolation is used to update the elevation map. Two arrays are used to perform bilinear interpolation: the elevation accumulator $E(r, c)$ and the bilinear weight accumulator $W(r, c)$. For each sample, the four grid cells surrounding (r, c) are updated using

$$\begin{aligned} p &= r - \lfloor r \rfloor, & q &= c - \lfloor c \rfloor \\ E(r, c) &+= (1-p)(1-q)z, & W(r, c) &+= (1-p)(1-q) \\ E(r+1, c) &+= p(1-q)z, & W(r, c) &+= p(1-q) \\ E(r, c+1) &+= (1-p)qz, & W(r, c) &+= (1-p)q \\ E(r+1, c+1) &+= pqz, & W(r, c) &+= pq \end{aligned} \tag{3}$$

where $\lfloor\ \rfloor$ is the floor operator. After all samples have been accumulated, the elevation Z at each grid cell is determined using

$$Z(r, c) = E(r, c)/W(r, c) \tag{4}$$

Because of the irregular sampling by the scanner, it is possible that a grid cell did not have a sample projected into it and consequently does not have an elevation value. For efficiency, it is important that the elevation map be free of holes, especially near the center of the map. A simple interpolation scheme is used to fill any holes. First, hole cells are detected by finding cells that do not have an elevation but are surrounded by cells with elevation. Next, each hole cell is assigned the average elevation of all neighboring cells that have elevation values. When this process is repeated until all hole cells have an elevation value, the holes in the elevation map are filled incrementally. Figure 7 shows a typical range scan and an elevation map after hole filling. Figure 8a shows an elevation map (as a contour map) constructed for a 400×400 sample raster scan of synthetic Martian terrain.

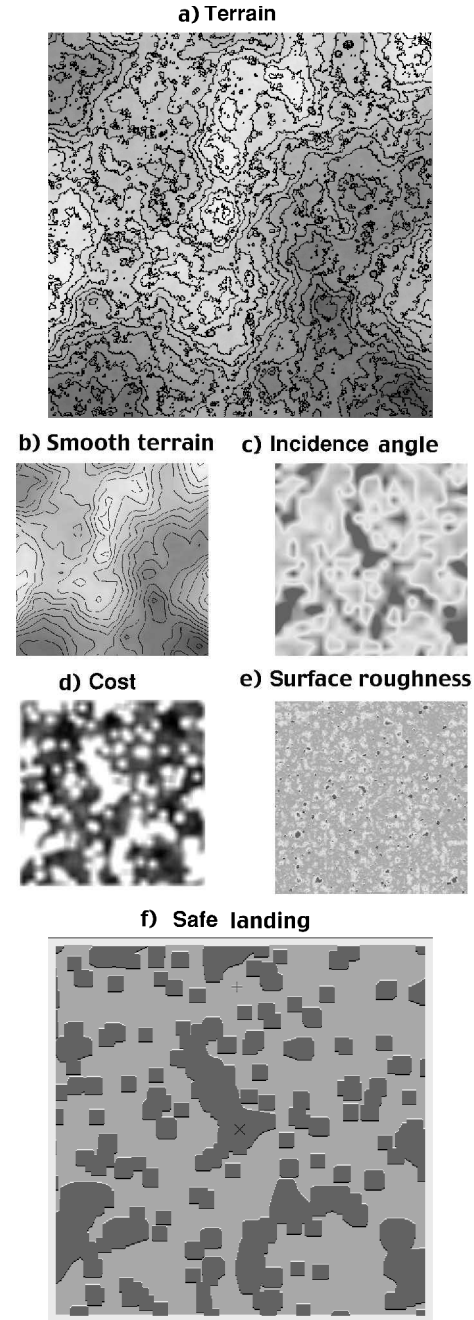

Fig. 8 Hazard avoidance maps; for incidence angle surface roughness and safe landing maps, dark gray indicates hazard, light gray indicates safe, and white indicates unknown or partially safe.

Hazard Map Generation

The next step after generating the elevation map from the lidar samples is the generation of landing incidence angle and roughness maps that will be used to select safe sites for landing. To achieve this, the surface must be separated into two components: the underlying smooth surface from which landing incidence angle is computed

and the rocks and pits embedded in this surface from which surface roughness is computed. First the low-frequency surface is computed; deviations from this surface are then determined by subtracting it from the original elevation map.

Robust Plane Fitting

Because the lander will be approaching the surface vertically, landing incidence angle is defined as the angle α between the local surface normal $\boldsymbol{n}$ and the geodetic normal of the surface $\boldsymbol{n}_g$. Local surface normal at a terrain grid cell is computed by fitting a plane to the terrain in the neighborhood of the cell; the computed local surface normal depends on the size of the neighborhood over which the plane parameters are estimated. For safe site selection, the normal of interest is the one that is computed with a patch that is the size of the lander footprint because this normal will indicate the resting angle between the lander and the surface. If the lander footprint is L (meters) then we approximate this size in units of grid cells as $l = \lceil L/s \rceil$ where $\lceil\ \rceil$ is the ceiling operator.

A plane is represented by the equation

$$\boldsymbol{n} \cdot \boldsymbol{x} + d = 0$$

where $\boldsymbol{n}$ is the surface normal of the plane and d is the plane intercept. The neighborhood of a grid cell can be represented by a set of three-dimensional points:

$$N = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$$

Plane fitting finds the best-fit plane $(\boldsymbol{n}, d)$ to these points given some appropriate error metric. In many cases, Mars terrain can be characterized by scattered rocks on top of relatively smooth terrain. The plane that is desired for estimating local surface normal is the plane fit to the surface in which the rocks are embedded. Given this domain specific insight, estimating the plane of the underlying smooth surface is a problem best solved by robust statistical methods, where rocks are treated as outliers from the underlying smooth surface.

The robust statistical method used to fit a plane to the underlying terrain is least median square (LMedSq) estimation. The LMedSq algorithm uses the following principal: Three points define a plane, investigate multiple triples of three-dimensional points in N, and eventually a triple that is free of outlier (rock) points will be found. The number of triples t that must be investigated is based on the expected percentage o of rock points in N and the desired probability P of obtaining a triple without outliers:

$$t = \ell n(1 - P)/\ell n[1 - (1 - o)^3]$$

LMedSq estimation algorithms follow a standard form.[8] The LMedSq algorithm specialized to plane fitting is as follows. Repeat the following steps for t triples. First, select a triple of noncollinear points $(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}_c)$ randomly from N. Next, compute the parameters $(\boldsymbol{n}, d)$ for the plane that is defined by the three points using

$$\boldsymbol{n} = (\boldsymbol{x}_b - \boldsymbol{x}_a) \times (\boldsymbol{x}_c - \boldsymbol{x}_a), \qquad d = -\boldsymbol{n} \cdot \boldsymbol{x}_a \tag{5}$$

Given this plane, the square plane errors r_i are computed for each of the remaining point in N:

$$\{r_i\} = \{(\boldsymbol{n} \cdot \boldsymbol{x}_i + d)^2\}$$

If the median of the described square plane errors is less than the median square plane error computed for all previous planes r_{med}, the current plane parameters becomes the best encountered so far $(\boldsymbol{n}_{\text{best}}, d_{\text{best}})$. The process is repeated for all of the t triples. This procedure finds the plane that minimizes the median square plane error, hence, LMedSq.

Next, the robust standard deviation

$$\sigma_r^2 = (1.4826\{1 + [5/(t - 3)]\})^2 r_{\text{med}}$$

is computed. When σ_r is used, a neighborhood point $\boldsymbol{x}_c$ is considered an outlier, and consequently eliminated, if

$$r_i = (\boldsymbol{n}_{\text{best}} \cdot \boldsymbol{x}_i + d_{\text{best}})^2 > \sigma_r$$

Fig. 9 LMedSq vs a priori LSq plane fitting.

The final step in robust plane fitting is to fit a least-squares (LSq) plane to the remaining inlier points in the neighborhood. To fit a plane to multiple points, we use the standard LSq plane-fitting algorithm based on finding the eigenvector of minimum eigenvalue of the scatter matrix of the points.[9]

The LMedSq algorithm eliminates the points in the neighborhood that correspond to rocks and then fits a LSq plane to the remaining points. If a LSq plane was fit to the all of the points in the neighborhood, then the plane would be skewed in the direction of rock points, which would reduce the accuracy of the slope computation needed for hazard detection. A comparison of LMedSq and LSq plane fitting is given in Fig. 9.

Map Generation

The robust plane $(\boldsymbol{n}, d) = (n_x, n_y, n_z, d)$ fit at a grid cell $\boldsymbol{x} = (x, y, z)$ intersects the underlying smooth rock free elevation z_s. Here z_s is defined using the robust plane parameters as

$$z_s = -(n_x x + n_y + d)/n_z \tag{6}$$

The robust plane also defines the landing incidence angle at the grid cell,

$$\alpha = \cos^{-1}(\boldsymbol{n} \cdot \boldsymbol{n}_g / \|n\| \|n_g\|) \tag{7}$$

If a robust plane is fit at every grid cell, then the underlying smooth surface can be generated by computing z_s at every grid cell. Plane fitting is a relatively computationally expensive operation. Given the real-time requirements of landing, computing a robust plane at every grid cell is infeasible. Furthermore, robust plane parameters will not change drastically between grid cells, and so computing them at every cell is wasted effort.

To alleviate the computational burden, a robust plane is only fit to grid cells separated horizontally by the footprint of the lander l. Furthermore, the size of the neighborhood for each plane fit is set to a square regions centered around the grid cell of width l. The result is that robust planes are fit to a coarse grid of cells with nonoverlapping neighborhoods that are the size of the lander. At each of these grid cells, z_s and α are computed. In between grid cells, bilinear interpolation, similar to that described earlier, is used to fill in the gaps between grid cells. The end result is two maps, $Z_s(r, c)$ and $A(r, c)$, that describe the underlying smooth surface and the landing incidence angle, respectively, at each grid cell in the elevation map. Figures 8b and 8c show the smooth surface and landing incidence angle for the elevation map (Fig. 8a).

Once the smooth surface has been generated, the roughness map of the surface $R(r, c)$ is simply computed as the difference between the smooth surface $Z_s(r, c)$ and the elevation map $Z(r, c)$. Only absolute deviations are needed to characterize roughness. Thus,

$$R(r, c) = |Z(r, c) - Z_s(r, c)|$$

Figure 8e shows the roughness map for the elevation map (Fig. 8a).

Safe Site Selection

The strategy behind the safe site selection is to generate a landing cost map that first keeps the lander away from detected hazards and then, from the remaining terrain, selects the landing site that has minimal landing incidence angle and roughness.

Hazard map generation creates maps that define landing incidence angle and roughness at every grid cell in the elevation map. With use of the landing system constraints on maximum surface roughness $R_{\max}$ and landing incidence angle $A_{\max}$, safe sites in the terrain are

selected as follows. First, the regions of the terrain that are hazardous to the lander are detected by identifying grid cells (r, c) where

$$R(r, c) > R_{\max} \qquad \text{or} \qquad A(r, c) > A_{\max} \tag{8}$$

More specifically, a landing cost map $C(r, c)$ is constructed. If a grid cell (r, c) violates Eq. (8), then $C(r, c)$ is set to 1.0. Furthermore, the lander cannot intersect any grid cell that violates Eq. (8), and so if (r, c) is within l of a grid cell that violates Eq. (8), then $C(r, c)$ is also set to 1.0. The remaining unassigned grid cells are assigned the normalized product of landing incidence angle and roughness.

$$C(r, c) = [R(r, c) \times A(r, c)]/(R_{\max} \times A_{\max})$$

$C(r, c)$ will be at or near 1.0 around grid cells that violate the landing constraints; $C(r, c)$ will be near zero for regions that have small roughness and landing incidence angle. When the minimum of $C(r, c)$ is found, the best landing site will be found. However, in addition to minimizing the landing cost, the landing site selected should also be near regions of similarly low cost. This concept can be implemented by smoothing the cost map and then finding its minimum. Specifically, the cost map is smoothed by setting $C(r, c)$ to the average of all costs in a square neighborhood of size l centered around (r, c). The best safe landing site is then selected at the terrain coordinate (r, c) that minimizes the smoothed $C(r, c)$. Figure 8f shows a landing map with detected hazards and buffer zones of size l, and Fig. 8d shows the landing cost map for the elevation map (Fig. 8a).

Powered Landing Guidance

Hazard-avoiding landing guidance was first developed for Apollo,[10,11] and the guidance algorithm planned for Mars landings is close to that of Apollo. In manned lunar landings, the commander identified landing hazards and manipulated a joy stick to redesignate the selected landing site to a hazard-free area of the lunar surface. In Mars landings, hazard detection and avoidance hardware and software take the place of the commander. Landing guidance cannot distinguish between human- and machine-derived landing-site redesignation commands. The most significant difference between landing on the moon and landing on Mars is that the lunar module approached the landing site along a nearly horizontal trajectory, whereas the Mars lander will approach the site along a nearly vertical trajectory. This difference affects the guidance algorithm, as will be explained.

Powered landing guidance transfers the lander from any current state (position and velocity) to touchdown in two phases, approach and vertical descent. Each phase has a target point. The approach-phase target may be 5 or 10 m above the surface, and the vertical-descent target is near the surface, where the descent engines are shut down. As the lander approaches a target, it nulls horizontal components of velocity and acceleration so that it arrives at the target moving vertically downward in an erect attitude. The vertical descent is at constant velocity until the engines are shut off at or slightly before touchdown. Shutting off the engines before touching down makes the lander less likely to tip over.

The guidance algorithm solves a two-point boundary-value problem. The boundary values are the initial state and the target conditions. Target conditions are the position, velocity, and acceleration at the target point. When zero horizontal components of target velocity and acceleration are reached, the lander arrives at the target point moving vertically and erect.

To solve the two-point boundary-value problem, the guidance algorithm fits a polynomial between the current state and the target conditions. Thus, the transfer trajectory is a polynomial in time. It expresses position and its derivatives, velocity, acceleration, jerk, and snap as functions of the independent variable time.

With time (which is zero at the target and negative en route) T, the position vector and its derivatives R, V, A, J, and S, and current and target conditions denoted by subscripts C and T, the two-point boundary-value problem is expressed by

$$R_C = R_T + V_T T + A_T T^2/2 + J_T T^3/6 + S_T T^4/24 \tag{9}$$

$$V_C = V_T + A_T T + J_T T^2/2 + S_T T^3/6 \tag{10}$$

Both current vectors and the first three of the five target vectors are given. The solution is the following matrix equation, which expresses the target jerk and snap vectors in terms of the five given vectors:

$$\begin{bmatrix} J_T \\ S_T \end{bmatrix} = \begin{bmatrix} 24/T^3 & -6/T^2 \\ -72T^4 & 24/T^3 \end{bmatrix} \begin{bmatrix} R_C - R_T - V_T T - A_T T^2/2 \\ V_C - V_T - A_T T \end{bmatrix} \tag{11}$$

In terms of the target jerk and snap vectors, the current acceleration vector is given by

$$A_C = A_T + J_T T + S_T T^2/2 \tag{12}$$

The guidance algorithm computes target jerk and snap until near the end of the phase, stopping short of the end to avoid dividing by powers of time as time approaches zero. It computes current acceleration until the end, using computed values of jerk and snap even when they are no longer being computed. The current acceleration is used for commanding engine thrust magnitude and direction.

This guidance algorithm solves the two-point boundary-value problem for any value of time. For Mars landings, time is chosen to produce a trajectory in which the vertical component of jerk is constant. This contrasts with Apollo guidance in which time was computed to produce a specified target value of the downrange component of jerk. In the Mars case, time is the solution of a quadratic equation, and in the Apollo case it was the solution of a cubic equation.

Simulation

The simulation ties together all of the previously described pieces. The simulation consists of three modules: the terrain server (TS), the powered guidance module (PGM) and the lidar model/hazard avoidance module (HAM). The TS communicates with the HAM through TCP/IP sockets, and the PGM and HAM communicate though named pipes.

First, the terrain server is started with a command to generate a piece of terrain large enough to cover all lidar scans expected during the simulation. Next, the PGM and HAM are started. The PGM starts with an initial state and then propagates this state according to the powered guidance equations. There are no navigation or control errors in the simulation.

At a rate of once per second, the PGM sends a command to the HAM to generate a lidar scan and detect the safest place to land. This command has the position and attitude of the spacecraft when the scan was requested and the current landing site. The HAM requests some terrain from the terrain server, generates a lidar scan, and detects hazards. If the current landing site is hazardous, then the HAM sends a new landing site back to PGM, and the PGM computes the new trajectory to the new landing site. If the current landing site is safe, then the HAM sends back the same current landing site, and the PGM keeps the current trajectory. This hazard avoidance loop repeats until the spacecraft has landed. Parameters on landing site position, spacecraft trajectory, and fuel consumption are recorded. If desired, images of hazard maps are recorded for each step of the simulation as well. Images recorded for five steps of a safe landing run are given in Fig. 10.

Simulation Results

To demonstrate the usefulness of the simulation and to prove the performance of the hazard avoidance algorithms, some Monte Carlo simulations were conducted. The first set of simulations quantifies the performance of the hazard detection and avoidance algorithms for a single lidar scan. The second set of simulations investigates the performance of the end-to-end powered landing system. The tests are preliminary and will be used to design more accurate and more robust guidance and hazard avoidance algorithms. The purpose of the tests is to demonstrate the variety of parameters that can be investigated with the simulation.

Table 1 Nominal parameters for the safe landing simulation

Parameter	Value
Terrain map: size	500 m
Terrain map: resolution	0.1 m
Terrain map: rock density	0.1
Lidar: field of view	10 deg
Lidar: resolution	100
Lidar: range error	0.02 m
Lidar: range resolution	0.02 m
Lidar: pointing error	0.001 deg
Lidar: pointing resolution	0.001 deg
Lidar: divergence	0.1 deg
Lidar: altitude	500 m
Hazard avoidance: lander base size	2.5 m
Hazard avoidance: roughness	0.5 m
Hazard avoidance: incidence angle	13 deg

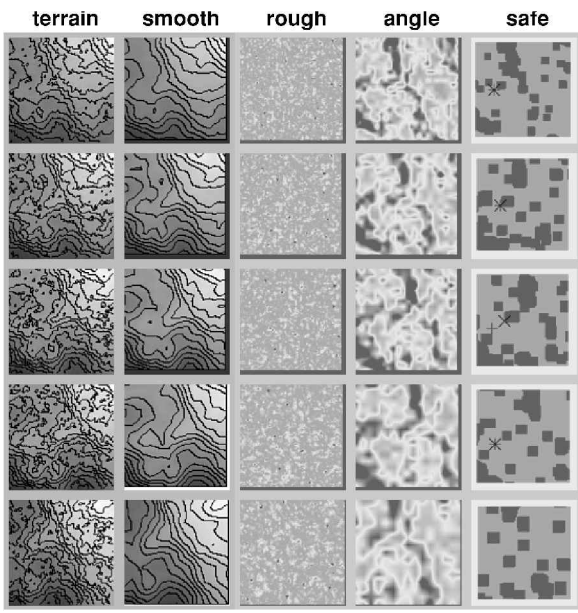

Fig. 10 Example of simulation output; each row corresponds to a single step of the simulation. Note that there is a redesignation in the third row of images.

Hazard Avoidance

Three lidar parameters were investigated for their effect on hazard detection: lidar range detection noise σ_ρ, lidar altitude a above the terrain, and lidar resolution n. Unless otherwise stated, the nominal parameters for the simulation are given in Table 1.

Figure 11 shows the results of the simulations in terms of safe site detection probability, that is, the percentage of truly safe sites that were detected by the lidar scan, and safe landing probability, that is, the number of landing sites selected by the hazard avoidance algorithm that were truly safe when compared to ground truth. Maximizing these probabilities is desirable, although a high safe landing probability is more important than a high landing site detection probability.

The performance of hazard avoidance with respect to altitude is poor for very low altitude. At moderate altitude, the performance improves. Finally, at high altitude, the performance drops off. The poor performance at low altitude is caused by the variability of terrains used in the simulation. When at low altitude, it is possible that the scanned terrain does not contain a safe place to land, that is, all of the visible terrain is hazardous. This observation indicates that it is better to make hazard avoidance decisions when high above the surface. Because the laser footprint increases with altitude, hazard detection performance degrades with altitude. This is the reason for decreasing hazard avoidance performance at high altitude.

The performance of hazard detection with respect to scanning resolution is as expected. As scanning resolution increases, the horizontal spacing between samples decreases. This will result in more

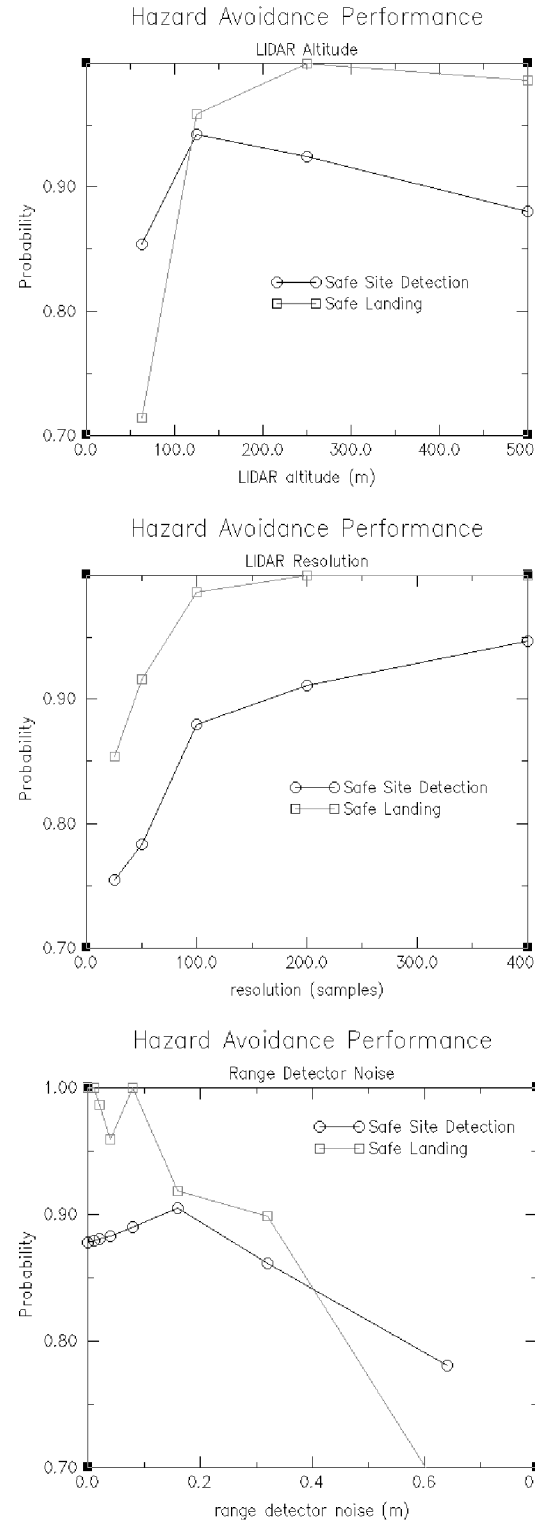

Fig. 11 Hazard avoidance performance.

hazards being detected and, consequently, better hazard avoidance performance.

Hazard avoidance performance decreases as range detector noise increases. As detector noise increases, more false hazards and safe zones will be detected, so that the probability of choosing a hazardous landing site increases.

Safe Landing

Safe landing probability for a complete run of the landing simulation (multiple scans and redesignations during descent) was investigated as a function of rock density. Approximately 100 Monte Carlo landings were performed, and the number of times that the

Table 2 Safe landing simulation performance as a function of rock density

Safe landing probability	Rock density
1.00	0.10
0.98	0.15
0.93	0.20

final landing site was safe, when compared to ground truth, was measured. For each Monte Carlo run, the terrain scanned by the lidar was varied. Each landing run started at 500 m with a purely vertical velocity. The results are given in Table 2. As expected, as rock density increases, the probability of ultimately landing at a safe site decreases. However, even for a high rock density (0.2) the safe landing probability is above 93%.

Conclusions

We have developed algorithms that process scanning lidar data to detect hazards and select a safe landing site during the terminal descent phase of Mars landing. In addition, we have developed a guidance algorithm that generates a trajectory to take the lander to the selected safe landing site. To test these algorithms, we have developed an end-to-end simulation of powered descent with hazard avoidance. This simulation includes the described algorithms and high-fidelity terrain generation and scanning lidar models. This simulator will be used during design of the next generation of Mars landers and will eventually be incorporated into a complete 12-degree-of-freedom simulation of all phases of Martian entry, descent, and landing. Types of questions the simulation will help answer are the following: What are the requirements for the lidar in terms of field of view and maximum range? How much fuel is consumed during landing? What control authority is needed to land safely given certain terrain statistics. What is the probability of safe landing?

Acknowledgments

The research described in this paper was sponsored by the Mars Technology Program at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA.

References

[1] Bernard, D. E., and Golombek, M. P., "Crater and Rock Hazard Modeling for Mars Landing," AIAA Paper 2001-4697, Aug. 2001.

[2] Gaskell, R., "Martian Surface Simulations," *Journal of Geophysical Research—Planets*, Vol. 98, No. E6, 1993, pp. 11,099–11,103.

[3] Gaskell, R., Collier, J., Husman, L., and Chen, R., "Synthetic Terrain Environments for Simulated Missions," Inst. of Electrical and Electronics Engineers, IEEE Paper 13.0406, March 2001.

[4] Pien, H., "Autonomous Hazard Detection and Avoidance," Charles Stark Draper Lab., Rept. CSDL-C-6303, Boston, MA, June 1992.

[5] Pien, H., Barton, G., Brand, T., and Suh, D., "Assessment of Blind Landing Risk on Viking Specific Terrain," Charles Stark Draper Lab., Rept. CSDL-R-2688, Boston, MA, July 1995.

[6] Halbrook, T. D., Chapel, J. D., and Witte, J. J., "Derivation of Hazard Sensing and Avoidance Maneuver Requirements for Planetary Landers," American Astronautical Society, Paper AAS-01-024, Jan. 2001.

[7] Cheng, Y., Johnson, A. E., Matthies, L. H., and Wolf, A., "Passive-Image Based Hazard Detection," International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS 2001), Paper AM028, June 2001.

[8] Stewart, C., "Robust Parameter Estimation in Computer Vision," *SIAM Review*, Vol. 41, No. 3, 1999, pp. 513–537.

[9] Duda, R., and Hart, P., *Pattern Classification and Scene Analysis*, Wiley-Interscience, New York, 1973, pp. 116–119.

[10] Cherry, G. W., "E Guidance—A General Explicit, Optimizing Guidance Law for Rocket-Propelled Spacecraft," Massachusetts Inst. of Technology Instrumentation Lab., Rept. R-456, Cambridge, MA, 1964.

[11] Klumpp, A. R., "Apollo Lunar Descent Guidance," *Automatica*, Vol. 10, 1974, pp. 133–146.