Computer modelling: a structured light vision system for a Mars rover

John J. Nitao † Donald H. Cronquist, Jr. *

† Consultant, FMC Corporation

* FMC Corporation, Corporate Technology Center
1205 Coleman Ave, Box 580, Santa Clara, CA 95052

ABSTRACT

A computer model has been developed as a tool for evaluating the use of structured light systems for local navigation of the Mars Rover. The system modeled consists of two laser sources emanating flat, widened beams with a single camera to detect stripes on the terrain. The terrain elevation extracted from the stripe information goes to updating a local terrain map which is processed to determine impassable regions. The system operates with the beams and cameras fixed except, now and then, the beams are vertically panned to completely refresh the local map. An efficient surface removal algorithm determines the points on the terrain surface hit by rays in the bundle. The power of each reflected ray that falls on each pixel of the camera is computed using well-known optical laws.

1. INTRODUCTION

Some of the non-navigable features on Mars may not be detectable from orbiting satellites such as the Mars Observer. In order to avoid obstacles in its path the Mars Rover must either be tele-operated or possess some degree of semi-autonomy Because of the long communication delay from Earth tele-operation will be very time consuming. However, semi-autonomy will require on-board obstacle detection system, intelligent processing, and control. An example of an obstacle detection system under consideration uses a structured light vision system to obtain range data which is used to extract terrain features. Evaluation of a sensor system is complicated by its interaction with rest of the vehicle system. Examples of other subsystems are inertial navigation, vehicle suspension, and autonomous control; the effect of the environmental conditions on Mars is also extremely important. Computer simulation is an effective tool for evaluating system performance from an overall systems standpoint. A simulator including vehicle-terrain interaction, sensors, control, and processing is being developed by the FMC Corporation to test Rover design concepts In this report we describe the structured light vision system (SLVS) model.

2. STRUCTURED LIGHT SYSTEM

As envisioned by S. Harmon of Robot Intelligence International, Figure 1 shows a possible SLVS for the Mars Rover. It consists of two laser beams mounted above a camera. The laser beams are spread by a cylindrical lens.

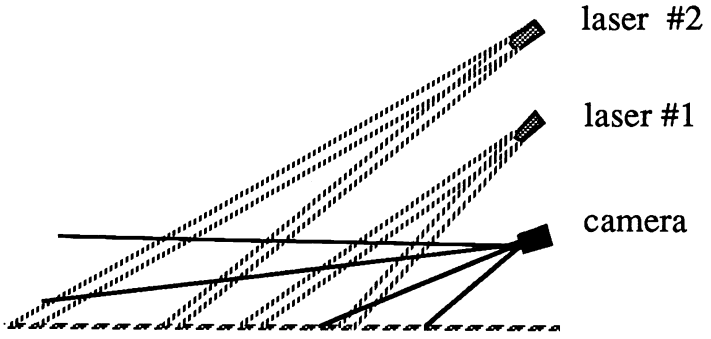

Figure 1. SLV System

The camera and the beams have their own local coordinate systems defined with respect to a local vehicle coordinate system, which, in turn, is relative to a global coordinate system. The attitude of the camera and beam sources are usually kept fixed to the vehicle so that the scanning of the beam across the terrain is due entirely to forward vehicle movement. Under some

circumstances, if the vehicle is at rest, the beams can be panned vertically in front of the vehicle. This scan mode may be necessary from time to time to correct an error build-up in the local map caused by de-synching of stripes taken at different times. Camera pictures of the beam are processed at a fixed rate. Light stripes are extracted from the image and their elevations go into updating a local terrain map.

3. OVERVIEW OF THE SLVS MODEL

We now describe how the SLVS system was modeled. At each camera frame the model must produce an image of the light stripes from the location and orientation of the vehicle, the geometry of the lighting system, and the camera and beam parameters. The terrain information is in the form of a data base holding computer generated rocks, crevasses, and craters.[2] These features are placed on a surface computed from stochastic interpolation of actual low resolution Mars data. All of the surfaces and features in the terrain data base are approximated by lists of triangles. The laser beam is decomposed into "ray pencils" through an imaginary grid in front of the laser (Figure 2). A hidden surface algorithm determines the intersection of each ray with the terrain point closest to the beam source. If this point is in camera view, a ray will reflect from the point and into a camera pixel. For each reflected ray the light loss is computed as a function of the distance from the laser to the surface and from the surface to the camera. The amount of light reflected from the surface will be a function of surface type, angle from the laser source to the surface normal vector, and angle from camera direction to surface normal vector. In this way the intensity of the stripes is built up on the camera image plane. This image, except for spatial sampling effects, is the ideal image found by geometrical optics.

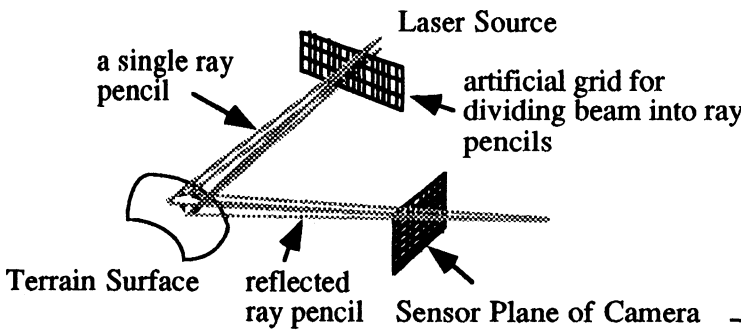

Figure 2. Ray Tracing from Laser Source to Camera Sensor Plane

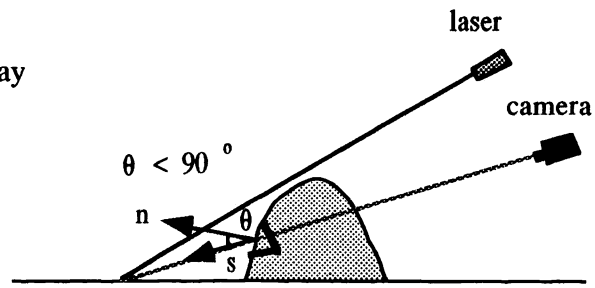

Figure 3. Obstruction by a Triangle Pointing Outward from Camera View

It is necessary to consider the occlusion of stripes by terrain features (Figure 3) A hidden surface removal is performed from the camera view point. If a point on another spot of the terrain is closer to the camera than the stripe, the stripe intensity is zeroed out for the appropriate camera pixel

After occlusions have been considered, errors such as lens aberration and lens dust, can be added to the image by spatial filters. Errors due to thermal and quantization noise can be added to the image by superimposing random noise satisfying a statistical distribution. Each pixel value in the image is adjusted to account for the dynamic range and bias level of the camera. It is, then, digitized to the level of precision of the A/D converter. Finally, thresholding or edge detection is performed to extract the light stripes from the image. To summarize, the basic steps in the algorithm are as follows,

1. For each ray pencil from laser source, find nearest point on the terrain by hidden surface removal
2. Compute the light intensity at each point on the stripe based on distance from the laser along the point's ray pencil
3. Follow the reflected ray pencils from the light stripe on the terrain to the corresponding pixels on the camera
4. Compute the intensity of each ray pencil that hits each pixel; intensity is based on the distance from surface along the ray and the reflectance from the surface
5. Perform hidden surface removal from camera's viewpoint over the terrain triangles that face away from the camera in order to remove the occluded portions of the light stripe
6. Filter camera image to account for aberrations and dust on lens
7. Add random sensor noise
8. Adjust dynamic range of each pixel and digitize
9. Extract light stripes from image

Currently, a version of the SLV simulation program has been coded and tested that includes steps 1-5 and a simple implementation of step 9. This version is appropriate for integration into the system simulator. In the following sections we will give a more detailed description this version. A version including steps 6-8 is planned for detailed off-line simulation of the SLV system.

4. HIDDEN SURFACE REMOVAL

Since the algorithm is used for both beam propagation and camera imaging, the description will be somewhat generic. The main goal of the algorithm is to compute a range raster map from a sensor to the closest points on the terrain surface. The hidden surface algorithm receives a list of triangles from the terrain data base for the region in front of the vehicle. Each triangle is projected to a sensor projection plane (SPP) (see Figure 4). That part of the SPP that is visible by the sensor is called the sensor screen (SS). Hidden surface removal for the camera uses the perspective transformation while the beams use a special cylindrical transformation described later. The projected triangles are scanned horizontally along the rows of the raster map to compute the range at each point . The steps in the algorithm are as follows:

1. skip the triangle if it is obviously not in the sensor map, i.e. skip those triangles with:
    a. all vertices in front of the SPP
    b. the back side of the triangle is facing the sensor (triangles have a convention such that when viewed from the front side, the vertices are in counter-clockwise order)
2. transform the vertices of the triangle to the SPP coordinate system;
3. skip triangle if:
    a. all the vertices are to the left, right, above, or below the SS.
    b. all the vertices are in the SS but their range exceeds the maximum range of the sensor
    c. none of the vertices or the sides of the triangle intersect the the SS and the center point of the SS is not in the triangle
4. scan the triangle
    a. break up triangle into two triangles each having a common horizontal side (see Figure 4)
    b. perform a horizontal line scan of the pixels on the sensor map from the left-most pixel lying on the triangle to the rightmost point; at each pixel we set the range map equal to the minimum of current distance to the point on the current triangle with the value that is already in the map (the range map is initialized to the maximum range of the sensor)
5. go to step 1

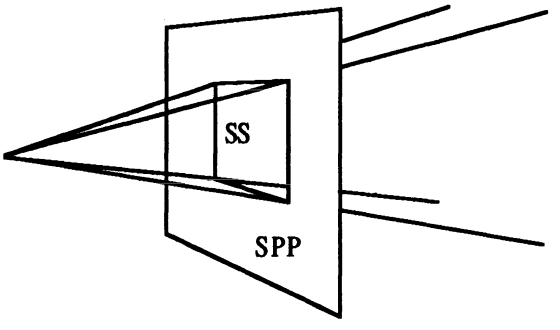

Figure 4. Sensor Screen (SS) and Sensor Projection Plane (SSP)

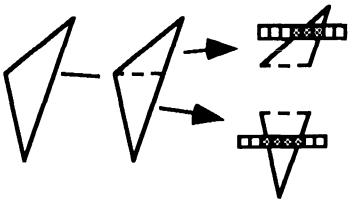

Figure 5. Line Scanning of the Triangle

5. CONSTRUCTING THE GEOMETRICAL OPTICS IMAGE

The camera image is found by following the rays from beams to the camera pixels and computing the power falling on each pixel. Except for the obvious spatial sampling errors the resulting image is that obtained through geometrical optics. Effects of diffraction, lens aberration, misfocusing, and lens dust can be treated by performing a convolution of the point spread function [3] on an image with finer resolution than the camera image. Calculations show that diffraction effects are not important compared to camera pointing errors arising from vehicle inertial navigation sensors. Further work needs to be done to analyze the other sources of errors.

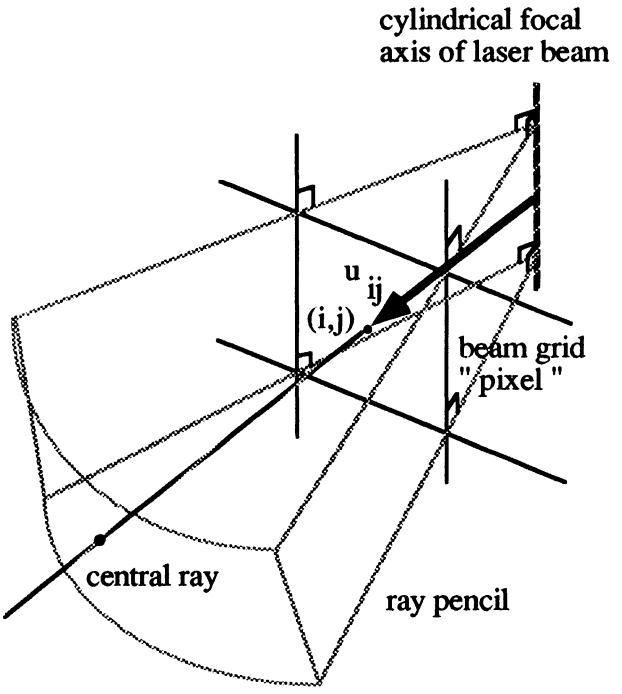

Figure 6. Definition of Ray Pencils Emanating from Laser Beam (Pencil Thickness is Exaggerated)

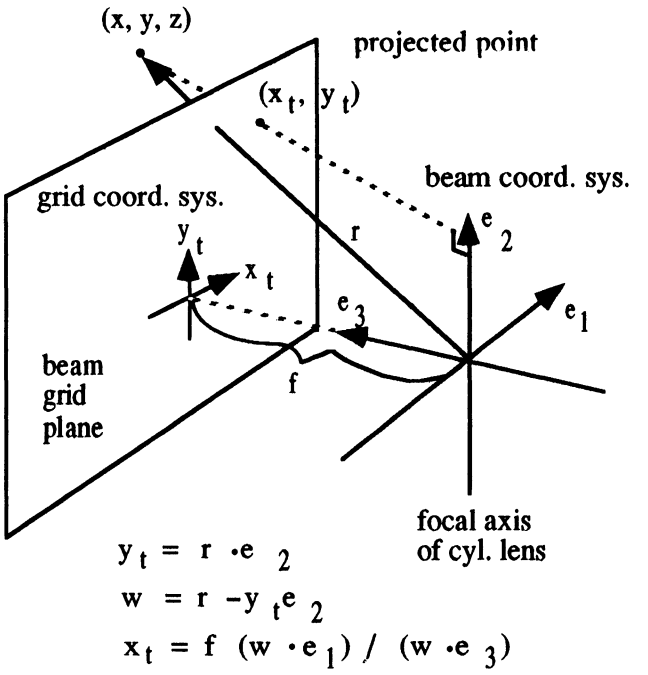

Figure 7. Projection of a Point (x,y,z) to the Beam Grid Plane

Each laser beam has a beam grid which splits the beam into ray pencils (Figure 6). The vertical lines of this grid are parallel to the focal axis of the beam. It is assumed that the beam emanating from the laser is passed through a cylindrical lens in order to make it spread. The focal axis in the figure is the focal axis of this lens. The grid can be considered to be made up of rectangular pixels that define each pencil. Ray pencils are useful because they have a finite thickness, and, therefore, a well-defined radiant flux (power / time). They are used to compute the irradiance (power / (time x area)) of each ray and the radiant flux into each camera pixel. In our discussion we assume that diffraction of the laser beam is negligible for our propagation distances. Therefore, the beam and its ray pencils have constant thickness in parallel to the focal axis.

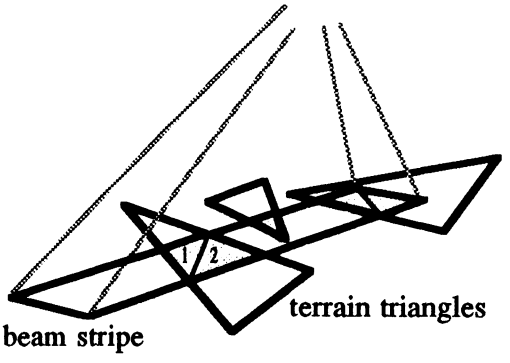

Figure 8. Clipping of Terrain Triangles to Beam Stripe

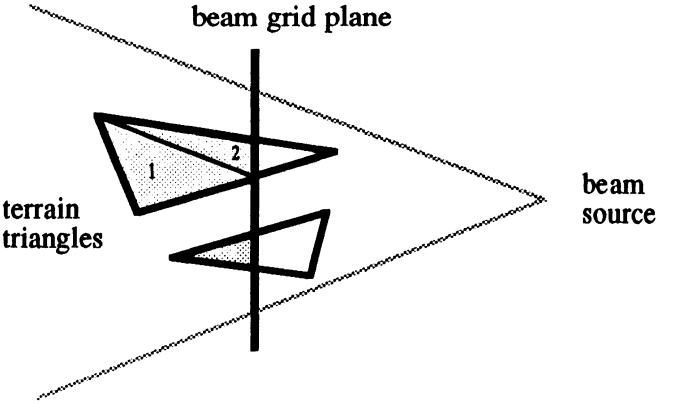

Figure 9. Clipping of Triangles to be in Front of Beam Grid Plane

To determine the point on the terrain hit by ray pencils, only central rays are considered. The point nearest to the beam source will be the actual point of ray incidence. Determination of this point requires the hidden surface removal algorithm described in the previous section. The vertices of the terrain triangles are transformed to the beam grid plane using the projective transformation defined in Figure 7 for a beam passing through a cylindrical source. But before this step, we clip off the portions of the triangles lying outside the beam as shown in Figure 8. The transformation, unlike the perspective

transformation, does not map terrain triangles onto the beam grid plane as triangles; except for vertical and horizontal lines, lines do not map to lines. By clipping, we force the transformed triangles to have vertices within the beam grid,and since beam thickness is small, the non-linearity of the triangle sides becomes insignificant. Transformation of triangles to the grid plane is defined only if the triangles lie in front of the plane (i.e., in back of the plane from the beam's perspective) . If a terrain triangle intersects the grid plane, the triangle has to be clipped to form triangles that are entirely in front (Figure 9). If two of the vertices lie in back of the grid plane, the clipping results in a smaller triangle lying in front. If one vertex lies in back, two triangles are created, and these are both processed.

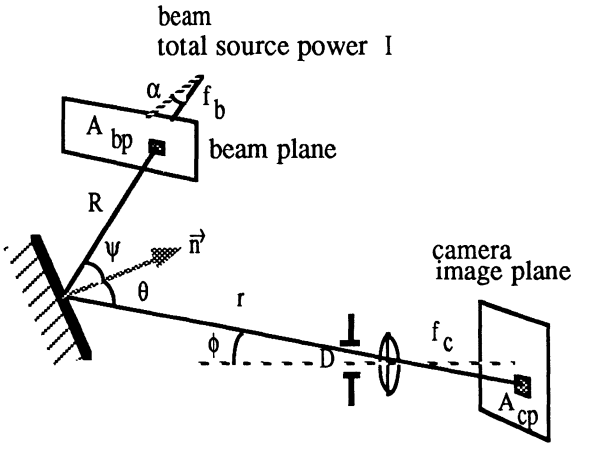

Figure 10. Calculation of Pixel Intensity

Construction of the camera image requires the power incident on each camera pixel. Figure 10 shows the propagation of the k-th ray emanating from the laser beam, reflecting off terrain, and onto a pixel. The following equation describes the radiant flux (power/time) entering the pixel coming from this beam.

$$I_k = \left[I_b r_d \frac{\pi}{4} \left(\frac{D}{f_c}\right)^2 C_{loss} F(\phi) A_{cp} \left(\frac{A_{bp}\cos\alpha}{A_b}\right) \frac{\cos^4\phi \cos\theta \cos\psi}{R r^2} e^{-\sigma r} e^{-\sigma R} \right]_k \qquad (1)$$

A_b	total area of beam grid
A_{bp}	pixel area on beam grid associated with incident ray
A_{cp}	pixel area on camera image grid associated with reflected ray
C_{loss}	optical loss coefficient for camera
D	camera aperture diameter
$F(\phi)$	optical loss function due to optical filtering
f_c	camera focal length
k	index referring to k-th incident ray emanating from laser beam
I_b	total laser beam power output
I_k	power incident on camera pixel receiving reflected ray from the k-th incident ray
R	distance from beam source to terrain
r	distance from terrain to lens plane
r_d	reflectivity coefficient of terrain, assumes Lambertian surface
α	angle between incident ray and beam plane normal axis
ϕ	angle between reflected ray and camera normal axis
ψ	angle between incident ray and normal to terrain triangle
σ	atmospheric attenuation coefficient
θ	angle between reflected ray and normal to terrain triangle

The factor $(A_{bp}\cos\alpha / A_b)$ represents the proportion of the total beam flux radiating from the incident beam. Explanation of most of the other terms can be found in standard references in optics.[1] Many of the terms in the equation depend only on

the pixel location, not on the incident terrain point, and can, therefore, be fetched from pre-computed tables. For example, tables of $F(\phi)$ are computed for each camera pixel before simulation starts. The cosines and attenuation factors are also pre-computed as well as the camera and beam pixel areas. The distances R and r are computed without using square roots by taking advantage of the special properties of triangles.

6. REMOVING STRIPE SECTIONS OCCLUDED BY TERRAIN

By tracing the rays from each camera pixel, the ray tracing algorithm can be used to remove those portions of the stripe that are occluded by the terrain. If the ray intersects the terrain at a point closer to the camera than the point on the line stripe the pixel intensity is zeroed. Performing this procedure for every pixel on the camera image would be too time consuming. Checking only pixels that "see" the stripe is one way of lowering the cost. However, our hidden surface algorithm cannot be restricted to specific regions on the camera image because, for the sake of efficiency, it works in a backward fashion, taking each triangle of the terrain data base and finding all the raster points that see the triangle. One can, however, reduce the computation by reducing the number of triangles that are checked. We make the observation that a ray from the camera to a stripe is occluded by the terrain if the ray passes through a triangle on the back side of the terrain. Such triangles that face away from the camera have the property that their outward surface normal vector (the ordering of the vertices of each terrain triangle is counterclockwise when viewed from the outward side) makes an angle of less than 90 degrees with respect to the camera axis vector, or, equivalently, the dot product of these two vectors is a positive number (see Figure 3). We may further reduce the number of triangles by considering only triangles with a least one vertex not past the furthest extent of the upper laser beam.

Extraction of the stripe edges is currently performed by simple thresholding. The edge pixels are transformed from camera coordinates to local vehicle coordinates by triangulation based on the mounting position and attitude of the camera and beams. The position and orientation of the vehicle from inertial navigation sensors are used for transformation to three dimensional global coordinates including elevation.

7. BUILDING THE LOCAL OBSTACLE MAP

Current plans for the real system are for the edges to be extracted every tenth of a second. Since only two lines of stripe data are produced by the SLVS every cycle, a complete picture of the terrain around the vehicle requires saving data onto a "local terrain map" (LTM). The LTM is needed for finding impassable regions. This information is converted into the "local Navigator map" used by the intelligent control module, the Navigator, which pilots the vehicle around obstacles.

The LTM is a square grid, with a nominal spacing of 0.1 meters between grid points. The map has 201 by 201 grid points and covers a region 20 by 20 meters. Each point on the map has two data fields: the global elevation value and a key. The key has one of two values:

's' the elevation at this grid point is not known
'u' the elevation at this grid point is known

All of the grid points are initialized to 's' at the start of the run. Each time the map is updated, the central grid point of the map is moved with the vehicle but only in increments of the grid spacing. The global coordinate of the central grid point is updated so that the global coordinates of the other grid points can be computed. The map does not rotate but the x and y directions are always parallel to the x and y global axes. The central grid point at the start of a mission is at the local vehicle origin.

The map is updated at every camera frame. By extracting the edges of the stripes, the camera pixels that correspond to edge points are known. A table holds the local vehicle coordinates of the edge plane point seen by any given camera pixel, i.e. the point where a ray passing through the camera pixel intersects the lower edge plane of the light beam; thus, the local vehicle coordinates of edge points falling on the terrain surface can be quickly computed. After converting the edge points to global coordinates, the horizontal x and y coordinates of the LTM points are translated so that the central grid point of the LTM is the xy origin. The grid point having x,y coordinates closest to an edge point inherits the elevation of the edge point and its key field is set to 'u'. Transformation to global coordinates requires knowledge of the global position of the vehicle. The x and y coordinates can be obtained directly from inertial guidance sensors and corrected, possibly, by periodic updates from satellite sightings of the Rover's beacons. The pitch, roll, and yaw angles of the vehicle can also be measured from on-board sensors. Accurate elevation of the vehicle is more difficult to obtain, however. In our implementation, the vehicle determines its global elevation by adding its measured elevation above the ground surface to the elevation of its current

location on the LTM. Wheel sinkage into the ground surface makes wheel radius an unreliable measure of elevation. We therefore assume that a proximity sensor is mounted under the vehicle to measure the distance from the local vehicle origin to the ground surface. This distance is used to correct the vehicle elevation read from the LTM. Since this LTM elevation was obtained at previous camera frames, errors may accumulate as this process continues. Although, to a degree, only relative elevations are important, error accumulation can lead to inconsistency between points taken at different frames. Thus, the vehicle should stop periodically in order to perform a complete vertical scan of the terrain to obtain a fresh LTM.

In some cases, gaps between adjacent stripe edges may occur on the map due to sensor noise, errors in edge extraction, turning of the vehicle, etc. They are also caused by occlusions from low lying rocks or other terrain features not large enough to be obstacles. These are more serious since they can be of larger area. For now, we have attempted to compensate by extrapolating the elevation of each new data point to each of its neighboring grid points that is not itself a new data point. This solution only works for gaps that are less than or equal to one grid spacing. A more general solution, which we have not yet implemented, is to allow the vehicle to go over the gaps that are less than one wheel thickness wide. To test if the gap contains elevations that are too high for the vehicle, an upper estimate of the maximum elevation of the gap region can be computed from the height of the occluding terrain feature or by using elevations from successive stripes bracketing the gap region. More work needs to be done to solve the "gap problem" in a computationally efficient manner.

The obstacle regions are not extracted from the LTM every camera frame but at after a fixed number of the frames. Determination of the obstacle regions is based upon a radial grid that is overlain over the LTM. Each point on the radial grid maps to the point on the LTM closest to that point. Points on the radial grid are tested as to whether they belong to an obstacle region in order of increasing radii along each of the sectors. A point Q is considered to be a member of the obstacle region if maximum slopes are exceeded between various point pairs on the terrain map. The point pairs are relative to an imaginary vehicle whose front axle midcenter is at the point Q being tested. The orientation of the vehicle is in the direction of the radial grid lines. The point pairs for slope testing are:

a) two front wheel centers
b) two rear wheel centers
c) the point midway between front wheel centers and the point midway between the rear wheel centers

In addition we test whether the terrain hits the midpoint between the rear and front wheel centers. If the rear wheels lie on points with unknown LTM elevation, instead of testing with the points in c) above we test with a pair of points closer to the center of the vehicle. The slope tests are made on all point pairs whose LTM elevation is known. If any of the slope tests exceed the maximum slope for that point pair, point Q is labelled an obstacle point and testing for that sector stops. If all the slope tests are satisfied, but if one of the slope points has unknown LTM elevation, the point Q is labeled as unknown, and testing for that sector stops. Otherwise, if all tests are passed and all points have known elevation, the point is labeled as being clear, and testing for that sector continues unless that point is the last one of the sector, in which case, the point is labelled as a visibility limit. Therefore, each sector has a label and point, the label being either obstacle, unknown, or visibility.

The Local Navigator Map is a list of labelled global vectors, and is used by the intelligent control and piloting module, the Navigator. The radial grid is converted into the map by basically connecting the labelled points on each sector of the radial grid by vectors. If a vector joins two points with the same label, that vector is labelled with that same label. The label of a vector joining two points is 'obstacle' if one of the points is labelled 'obstacle' and if the difference in radial distances between the two points is not large. Otherwise, if one of the points is labelled 'visibility' and the radial difference between the points is small than the vector is labelled 'visibility.' If none of these conditions hold, the vector is labelled 'unknown.'

8. SAMPLE CALCULATIONS

We now present two separate scenes where the SLVS was applied. The first scene consists of multiple pyramids. The vehicle starts at 7.5 meters from where the view from the camera in Figure 11 was taken. Figure 12 shows a portion of the LTM when this final point is reached The open areas represent regions where the elevation is unknown due in this case to 'shadowing' from the pyramids. Figure 13 is the resulting obstacle map. Figures 14 to 16 are for a terrain consisting of about 256 triangular facets. The light shaded lines on the LTM figure denote areas where the elevation is unknown. The LTM has a lot of unknown areas because the vehicle was forced to travel its path regardless of unknown areas and obstacles; hence there is a lot of shadowing. Processing of a single camera scene with 256 triangles requires about 8 seconds on a Mac II not including the graphics overhead. The processing of a single scene in the vertical scan mode takes considerably longer since tables must be regenerated at every camera and beam configuration The computation can be speeded up in the future by converting the code from floating point to integer arithmetic.

9. REFERENCES

1. Born, M. and E. Wolf, Principles of Optics, Sixth Edition, Pergamon Press, 1980.
2. Cronquist, Jr., D.H., J. J. Nitao, and L.S. McTamaney, "Development of a Martian Surface Model for Simulation of Vehicle Dynamics and Mobility," this Proceedings, 1989.
3. Goodman, J.W., Introduction to Fourier Optics, McGraw-Hill, 1988.

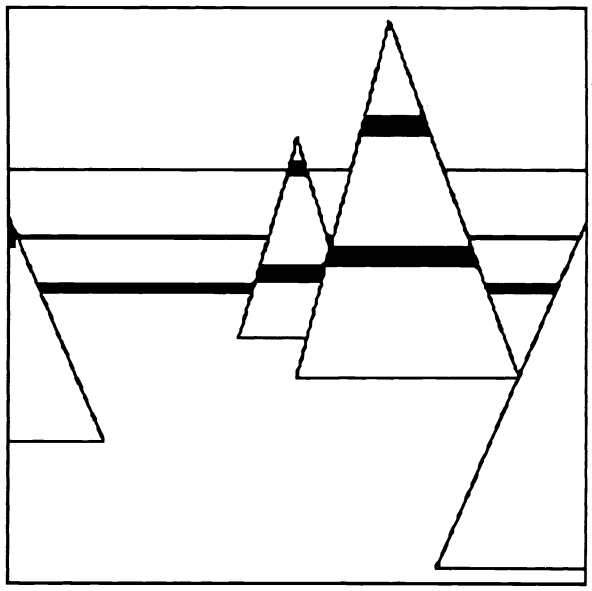

Figure 11. Camera View of Pyramid

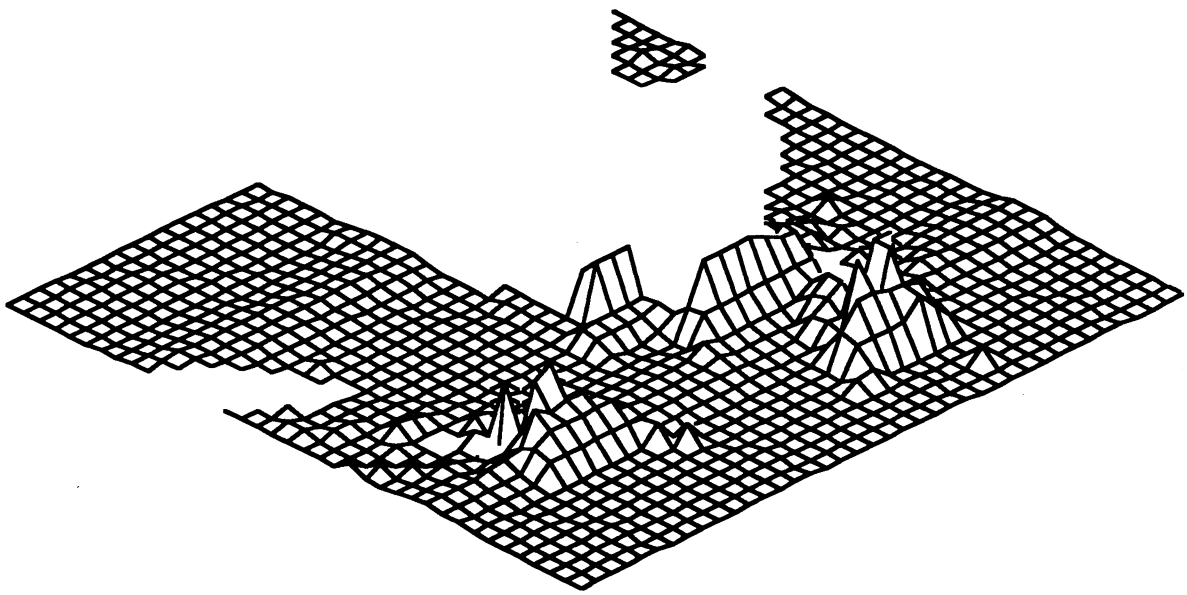

Figure 12. Local Terrain Map

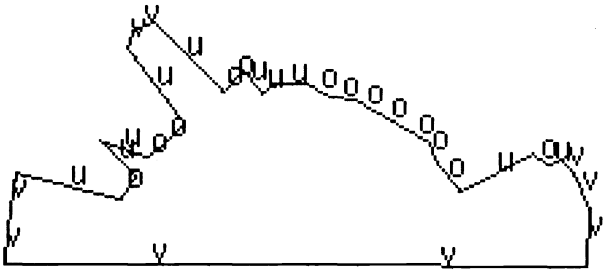

Figure 13. Obstacle Map

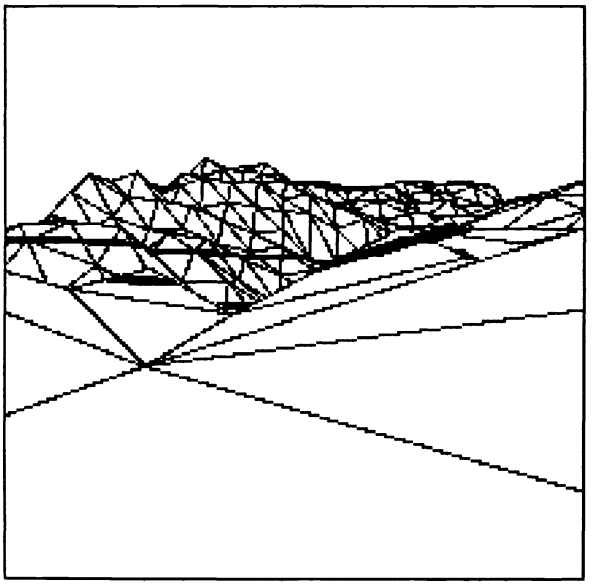

Figure 14. Camera View of Terrain Surface

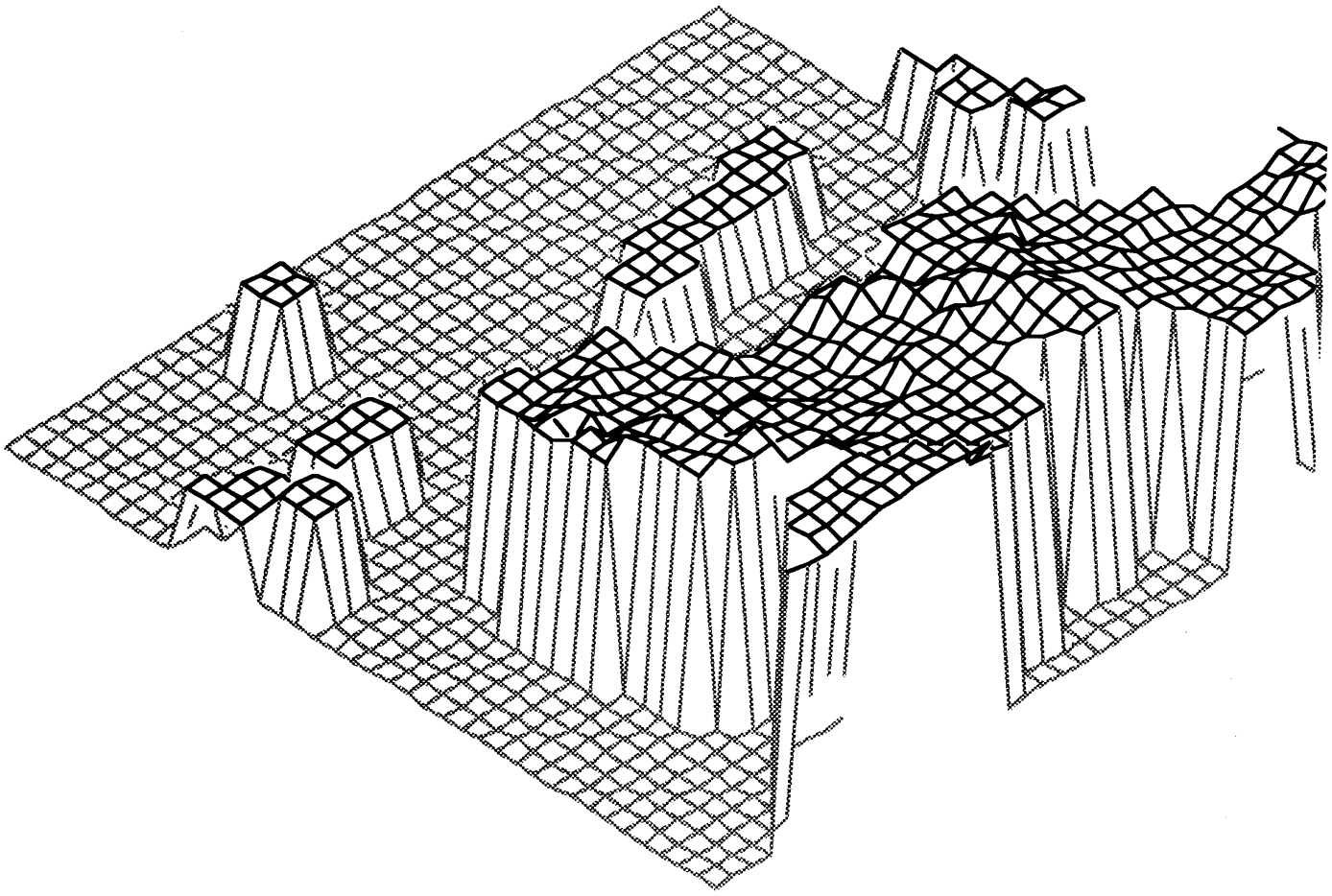

Figure 15. Local Terrain Map

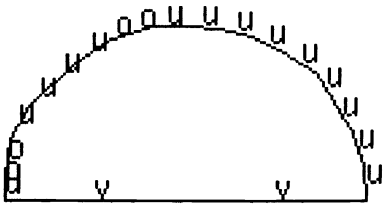

Figure 16. Local Navigator Map