CONNECTION MACHINE SYSTEM FOR PLANETARY TERRAIN RECONSTRUCTION AND VISUALIZATION

Mark Carlotto & Keith Hartt
The Analytic Sciences Corp.
55 Walkers Brook Dr.
Reading, MA 01867

ABSTRACT

An interactive system for computing terrain elevation maps and synthetic views of planetary scenes from a single panchromatic image is described. The system, implemented on an 8192 processor CM-2 Connection Machine, can generate an alternative view from an original (512 by 512) image in about 20 seconds. The system uses a shape-from-shading algorithm based on a numerical integration approach for computing relative elevations from the image and an oblique parallel projection/hidden surface removal algorithm for generating synthetic renditions of the scene. Both of these algorithms are implemented using scans and execute in constant time for a given image size. Results from Mars using Viking Orbiter imagery are presented.

1. INTRODUCTION

Scene rendering and animation have become important tools in computer graphics for terrain visualization, environmental impact assessment, and mission planning. In order to create synthetic views of terrain, a digital image of the terrain along with its corresponding digital elevation model are required. For most terrestrial applications this is not a problem given the availability of satellite and aircraft imagery and high resolution (typically 30 meter) digital terrain elevation data over much of the earth. For other planets like Mars, terrain elevation databases like that used in JPL's "Mars: The Movie" [1] are typically much lower in resolution and are not as readily available.

This paper describes a system that has been implemented on a Connection Machine[2] for computing planetary terrain elevation maps and alternative (oblique) views of the imaged scene from a single panchromatic image. The system uses a shape-from-shading algorithm based on a numerical integration approach[3] for computing relative elevations from the image and an oblique parallel projection/hidden surface removal algorithm for generating synthetic renditions of the scene. Both of these algorithms are implemented using scans[4] and execute in constant time for a given image size.

The organization of the paper is as follows: Section 2 summarizes the key features of the Connection Machine system and the *Lisp programming language. Section 3 describes the implementation of image restoration and enhancement, shape-from-shading, and scene rendering algorithms using scans. It also discusses the implementation of geometric transformations on the Connection Machine which are also used extensively in the system. Section 4 presents some results from Mars using Viking Orbiter imagery. Section 5 summarizes future work.

2. THE CONNECTION MACHINE AND *LISP

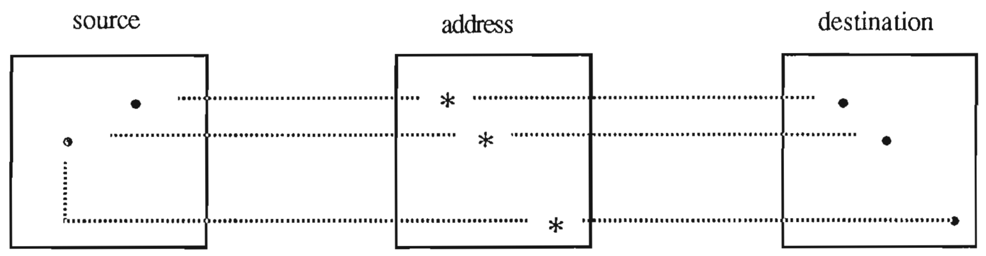
The Connection Machine (CM) is a data-parallel computing system containing up to 64K physical processors which can act like millions of virtual processors. The CM, originally conceived by Hillis[2], is built by Thinking Machines Corporation (TMC). A description of the CM system can be found in Ref. 5. The CM-2 contains 64K bits per physical processor and can perform 32 bit arithmetic at a rate of 2500 MIPs for a 64K system. The current system configuration at TASC is a 8192 processor CM-2 system with a Symbolics front-end processor and a frame buffer that allows the contents of the CM to be viewed at rates up to a gigabit per second.

*Lisp, a parallel dialect of Common Lisp, and PARIS, the assembly language of the CM are provided within the Symbolics software environment. *Lisp[6] is based on objects known as parallel variables or pvars which we shall denote in uppercase letters, e.g., A. Elements of pvars are processors that may be accessed by their cube address (i.e., relative to the hypercube) or their grid address, a(x,y). Elements of pvars may be signed and unsigned integers, variable precision floating point numbers, and booleans. The operation (!! a) returns a pvar in which the scalar a has been broadcast to all processors in the currently selected set. Macros such as *when, *cond, and *if select subsets of processors. For example the form (*if (=!! A B) (!! 1) (!! 0)) returns a pvar that contains ones in those elements in which A and B are equal and zeros elsewhere. Functions and macros that operate on all selected processors in parallel are identified by !! suffixes, e.g., (+!! A B). Reducing operations return a value from the currently selected set, e.g., (*min A). Relative addressing in the grid is also provided. The form (news!! A -1 0) returns a pvar that is equal to A shifted one position to the left.

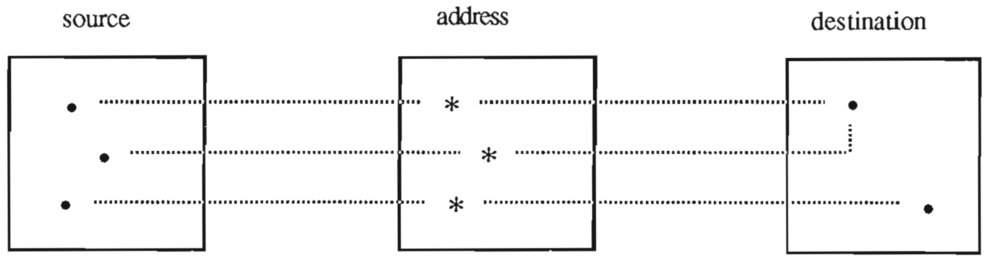
Scans[4] involve the application of a binary associative operator $\oplus$ (e.g., plus, and, or, min, max, or copy) to the set of elements [$a_0, a_1,..., a_{n-1},$] to produce the set [a_0, ($a_0 \oplus a_1$),... ($a_0 \oplus a_1$ ... a_{n-1})]. On the CM, scans can be performed on pvars along any dimension in either the forward or reverse direction. In addition, they can be conditioned on other pvars. Fig. 1 provides examples of scan operations. Implementations of shape-from-shading and scene rendering algorithms using scans are discussed in the next section.

processor number	0	1	2	3	4	5	6	7
value pvar	1	2	3	4	5	6	7	8
plus-scan (left to right)	1	3	6	10	15	21	28	36
segment pvar	nil	t	nil	t	nil	t	nil	t
segmented copy-scan (right to left)	2	2	4	4	6	6	8	8

Fig. 1 Examples of scan operations



a) Schematic of pref!! operation. The address pvar contains source addresses. Collisions due to multiple reads occur if a source element is written to more than one destination.



b) Schematic of *pset operation. The address pvar contains destination addresses. Collisions due to multiple writes occur if a destination element is read from more than one source.

Fig. 1 Interprocessor communication operations used for geometric transformations

Pref!! and *pset are interprocessor communication operations (Fig. 1). The form (pref!! A B) returns a pvar C that contains the value of A obtained from the processors addressed by B. The form (*pset A B C) copies the value of A and writes it as the value of B in each processor referenced by C. Collisions can occur when more than one output processor tries to read from the same input processor, or more than one input processor tries to write into the same output processor. The CM automatically handles collisions (several options are available to the user) at the expense of memory and/or speed. Pref!! and *pset are used to implement coordinate transform operations and are discussed further in the next section.

3. TERRAIN RECONSTRUCTION AND VISUALIZATION ALGORITHMS

Fig. 2 depicts the scene image formation and generation geometry. The slant and tilt angles of the sun are σ_S and τ_S where the slant is the angle between a ray and the z axis and the tilt is the angle between a ray and the x axis. The image i(x,y) is assumed to have been acquired under an orthographic imaging assumption (i.e., the altitude of the spacecraft is much larger than the range of elevations and the field of view is small). The synthetic image i'(x,y) is an oblique parallel projection of the original image mapped onto the derived elevation surface z(x,y). Its position is specified by slant and tilt angles σ_V and τ_V.

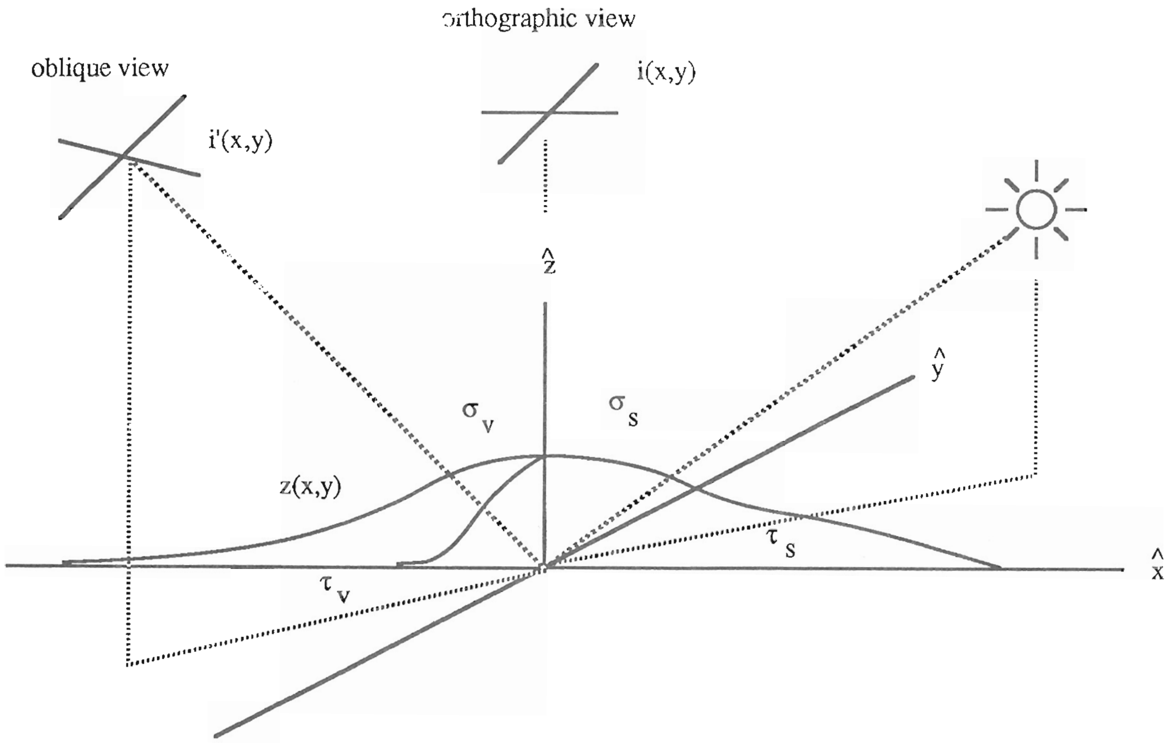


Fig. 2 Planetary imaging geometry. Sun and viewpoint are far enough away from the scene for parallel projection.

The basic processing sequence may be summarized as follows. First, the imagery is preprocessed to remove noise and improve contrast. Next is it rotated so that the sun is along the positive x-axis. The elevations are then computed from the rotated image. The image and its derived elevation map are rotated towards the viewer (assumed to lie along the negative y-axis). Finally the hidden surface removal and oblique parallel projection are performed. A series of successive views can then be generated as shown in the videotape presented at the conference. The following subsections describe key algorithms following the processing flow in a more or less logical order.

3.1 Image Preprocessing

For the Viking Orbiter imagery used in our system, image restoration and enhancement are required to remove "salt and pepper" noise due to random bit errors in transmission and to enhance the local contrast for display. Both can be implemented in terms of local means, which are easily implemented with scans.

Salt and pepper noise is removed by using a Laplacian operator (the difference between an image and its 3x3 local mean) to detect pixel outliers and replacing the value of outliers by their local mean. Similarly, contrast enhancement is performed by scaling the difference between an image and its local mean. Local means can be computed in two scans of an image, one for each dimension. In one dimension a plus scan, say left to right, is performed on the image i(m,n):

$$s(m,n) = \sum_{m'=0}^{m} i(m',n) \qquad (1)$$

and shifted versions are subtracted to produce the M by 1 local mean:

$$i_{Mx1}(m,n) = [\ s(m+M/2,n) - s(m-M/2,n)\ ]\ /\ M \qquad (2)$$

Repeating this in the other direction produces an M by N local mean when done.

3.2 Affine Transformation

The shape-from-shading and scene rendering algorithms described in the next two sections require that imagery be rotated in the x-y plane. Rotations are accomplished by the 2-d affine transformation:

$$\begin{aligned} x' &= ax + by + c \\ y' &= dx + ey + f \end{aligned} \qquad (3)$$

where (x,y) and (x',y') are input and output pixel coordinates. Catmull and Smith[7] describe a two-pass algorithm that involves accessing the data to be transformed in scan-line order, first in the x-direction and then in the y-direction. Since this requires twice as many memory references on the CM, a single pass algorithm was implemented directly using the pref!! operation described earlier. As a result, the transformation is "output driven" as shown in Fig. 3. The positions of all output pixels are found in the input, in effect solving for (x,y) in terms of (x',y'). Each output pixel value is bilinearly interpolated from the values of its four nearest neighbors in the input. This implementation of the affine transformation executes in an amount of time related to the number of collisions (the number of times output processors trying to read from the same input processor).

Operation	Reads/Processor (average)	Time (sec.)
translation	1	1.4
rotation (45°)	1	2.1
scale by 1/2	1	1.4
scale by 1	1	1.3
scale by 2	4	4
scale by 4	16	18

Table 1 Example timings for 2-d affine transformation on 512^2 image with 8192 processor CM-2

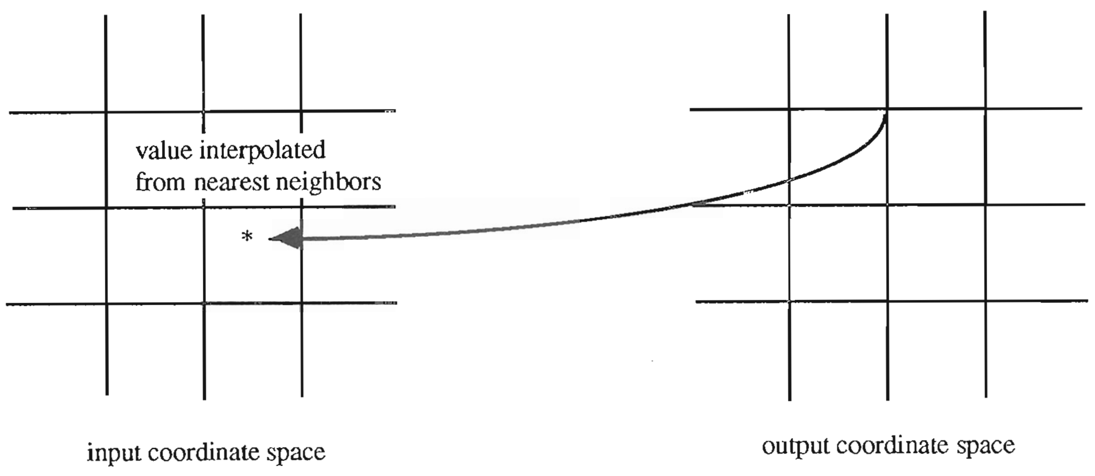


Fig. 3 Output driven geometric transformation used in 2-d affine warper

3.3 Shape-from-Shading

Shape-from-shading techniques involve the estimation of either the 2-d gradient field of a 3-d surface or the height of the surface itself from shading information in one or more image. Most approaches to single image shape-from-shading use either some form of numerical integration or constrained optimization technique. The latter involves iterative methods that attempt to produce smooth surface estimates that minimize the difference between the observed image and a synthetic image derived via the reflectance map. The iterative techniques are typically less sensitive to noise than the numerical integration techniques, but are very slow and are not discussed further in this paper. A particularly simple form of the numerical integration approach that involves summing brightness values along parallel strips in the image is described below.

The reflectance map R specifies the brightness of a point as a function of its gradients, i.e.,

$$i(x,y) = \alpha\, R(p,q) \tag{4}$$

where $p = \partial z/\partial x$ and $q = \partial z/\partial y$ and α depends on the albedo, the solar radiance, and the gain of the imaging system. Generating a shaded rendition of a 3-d surface in computer graphics involves simply computing brightness values from gradients. Shape-from-shading attempts to determine gradients, and ultimately elevations, from image brightness values. Horn notes in an early paper[3] that a linear approximation is adequate for most reflectance functions when the scene is obliquely lit. For example, Pentland[8] shows that for overhead viewing, if the gradients are small $|p|, |q| < 1$ and the zenith angle of the light source is greater than about 30°, the reflectance map for a Lambertian surface can be approximated by

$$R(p,q) = (p \cos\tau_s \sin\sigma_s + q \sin\tau_s \sin\sigma_s + \cos\sigma_s). \tag{5}$$

If the image is rotated so that the light source lies along the x-axis, Eq. 5 can be further simplified as $R(p,q) = p \sin\sigma_s + \cos\sigma_s$, and the elevations can be computed by simply integrating brightness values along parallel strips in the x-direction:

$$z(x') = z_0 + \int_0^{x'} \frac{\partial z}{\partial x}\, dx. \tag{6}$$

It can be shown that the elevation map can be determined, up to a scale factor and an offset by,

$$z(m,n)= z(0,n) + (1/\alpha \sin\sigma) \sum_{m'=1}^{m} [\, i(m',n) - \alpha \cos\sigma \,] \tag{7}$$

where the indices, m and n, are discrete and correspond to the x- and y-axes. The offsets z(0,n) for n = 0,1,... and the scale factor α are unknown. If the slopes in the direction of the light source can be assumed to be distributed about zero and the albedo assumed constant over the field of view, then $\alpha \cos\sigma$ can be approximated by the mean brightness using arguments similar to those used by Pentland[9]. Insofar as the boundary condition is concerned, one can assume that it is flat or can solve for the values that minimize the mean squared error between rows. These values turn out to be the average elevations of the strips. Using the latter boundary condition is equivalent to assuming that the average elevation along each row is the same.

The implementation of the strip integration method by scans can be summarized below. After rotating the image by $-\tau_s$, the average brightness is subtracted row by row by performing a plus scan left to right, dividing the result in the rightmost processor by the width of the image, performing a copy scan right to left, and subtracting this result from the image. The actual integration is performed right to left using a plus scan. Row offsets are adjusted by subtracting the average elevation from each row which requires two more scans. Smoothing between rows may be performed using another scan in the y-direction to further reduce striping.

3.4 Scene Rendering

Once the elevations have been computed, synthetic views can be created by projecting the original image (which may be enhanced to improve contrast) onto the elevation surface and reprojecting it to another viewpoint. In this section, it is shown that oblique views can be generated in constant time using scans.

Let i(m,n) and z(m,n) be the image and corresponding elevation surface rotated so the viewer is below (i.e., along the negative y-axis). If the viewer is at the horizon and is far enough away for a parallel projection to hold, only those portions of the scene whose elevation

$$z(x,y) > \max_{y'< y} \{ z(x,y') \} \tag{8}$$

will be visible. This is a trivial hidden surface computation that can be performed with a max scan bottom to top on the elevations, followed by a comparison of this result with the original elevations. The more general situation of a viewer located at an angle ϕ_v with respect to the zenith can be reduced to the above case by rotating the elevation surface:

$$z' = y \cos\phi_v + z \sin\phi_v \,. \tag{9}$$

As the viewing angle moves towards the zenith, more of the scene will be visible. The visible pixel values are then mapped via parallel projection onto the viewing plane by the transformation

$$\begin{aligned} y' &= y \cos\phi_v + z \sin\phi_v \\ x' &= x. \end{aligned} \tag{10}$$

This transformation is "input driven" and is implemented using *pset as shown in Fig. 4. Since the hidden surface algorithm removes those pixels which would otherwise write over the visible pixels in the output plane, no collisions in writing occur. However, not every output processor will be set and so the output image must be interpolated along the y-direction. This is done with two final scans in the y-direction, top to bottom and bottom to top.

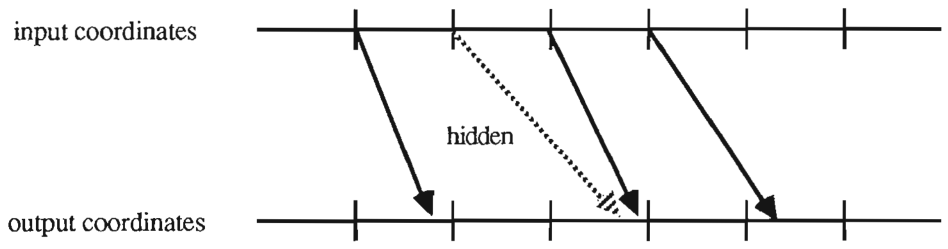


Fig. 4 Input driven geometric transformation used in parallel projection algorithm

4. EXAMPLE

Some sample results produced by the system are presented in Fig. 5. The original image (a) is a 512 by 512 region extracted from a raw Viking Orbiter frame. The three oblique views, (b) through (d) were generated in less than a minute from this image. A view of the elevation surface derived from the single image shape-from-shading algorithm is depicted in (e). An oblique view from a similar position is shown below in (f).

5. SUMMARY

An interactive system, implemented on a CM-2 Connection Machine, for computing terrain elevation maps and synthetic views of planetary scenes from a single panchromatic image was described. Plans are to integrate a more general planetary terrain reconstruction algorithm[10] that combines shape-from-stereo and shape-from-shading on the Connection Machine system. These systems will provide the planetary community with effective tools to support geological studies, mission planning (e.g., for the Mars Rover), and terrain mapping applications.

REFERENCES

1. K. Hussey, Mars: The Movie, Digital Image Animation Laboratory, JLP, Pasadena, CA. 1988.

2. W. D. Hillis, The Connection Machine, MIT Press, Cambridge, MA, 1985.

3. B. K. P. Horn, "Understanding Image Intensities," Artificial Intelligence, Vol. 8, pp 201-231, 1977.

4. G. Blelloch and J.J. Little, "Parallel Solutions to Geometric Problems on the Scan Model of Computation," Technical Report, AIM-952, MIT AI Laboratory, Cambridge, Ma, 1987.

5. L. W. Tucker and G. G. Robertson, "Architecture and Applications of the Connection Machine," Computer, Vol. 21, No. 8, August, 1988.

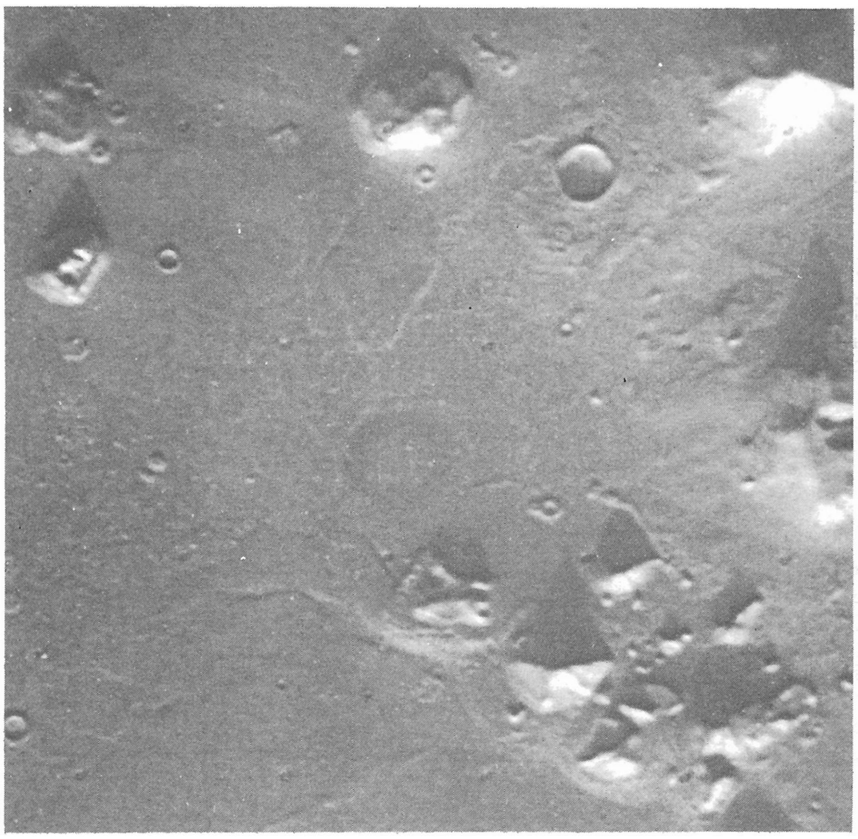
6. *Lisp Reference Manual (Version 5.0), Thinking Machines Corp., Cambridge, MA, Sept. 1988.

7. E. Catmull and A. R. Smith," 3-D Transformations of Images in Scanline Order," Computer Graphics, Vol. 14, No. 3, July, 1980, pp 279-286.

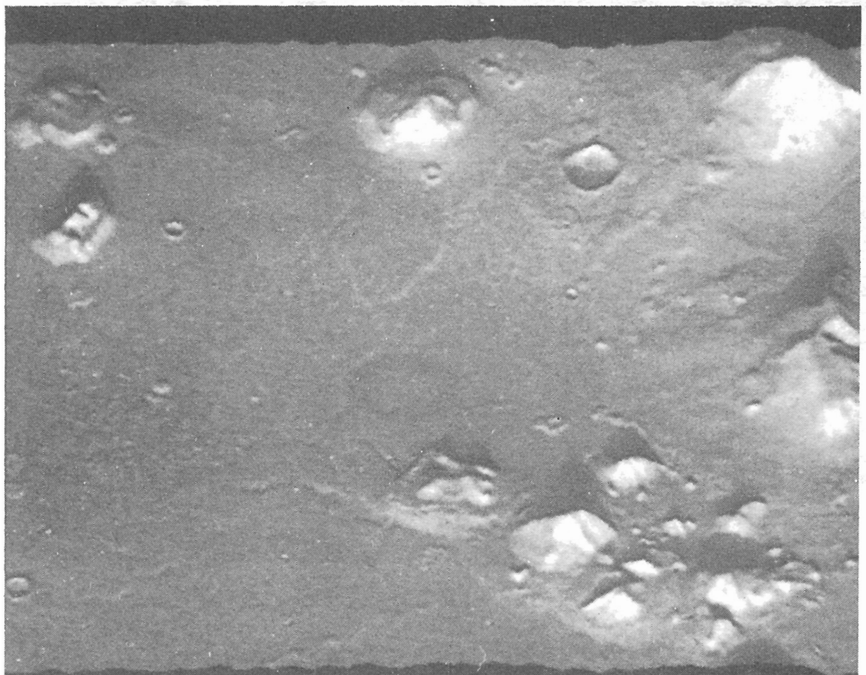
8. A. Pentland, "The Transform Method for Shape from Shading," Technical Report 106, MIT Media Laboratory, Cambridge, MA, July, 1988.

9. A. Pentland, "Local Shading Analysis," IEEE Trans. PAMI, Vol. 6, No. 6, pp 661-674.

10. K. Hartt and M. Carlotto, "A Method for Shape-From-Shading using Multiple Images Acquired Under Different Viewing and Lighting Conditions," Proc. IEEE Comp. Vision and Patt. Recog. Conf., pp 53-60, June, 1988.

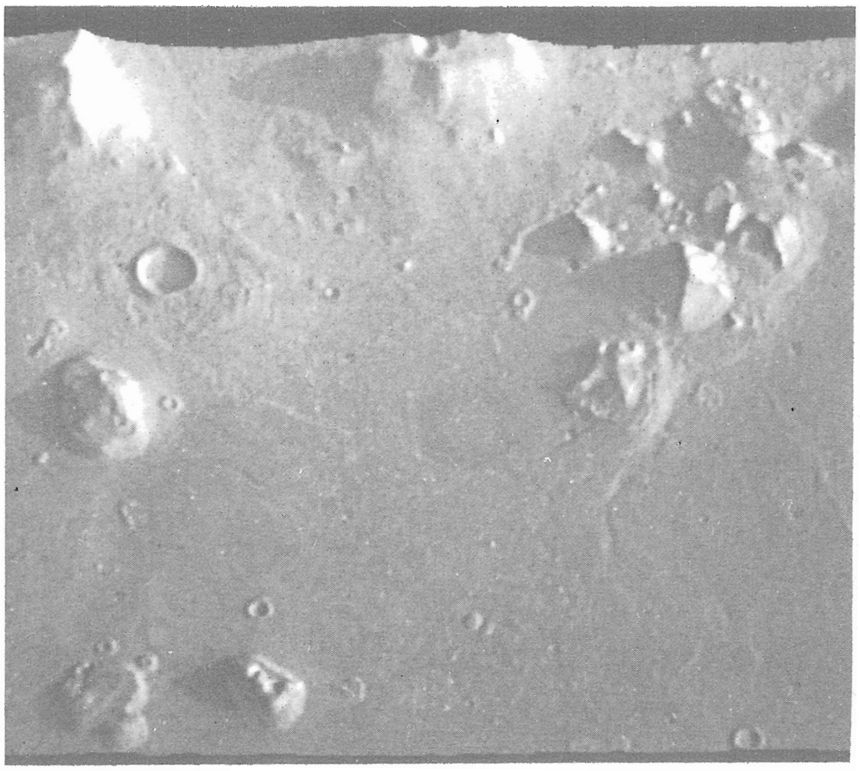


(a)

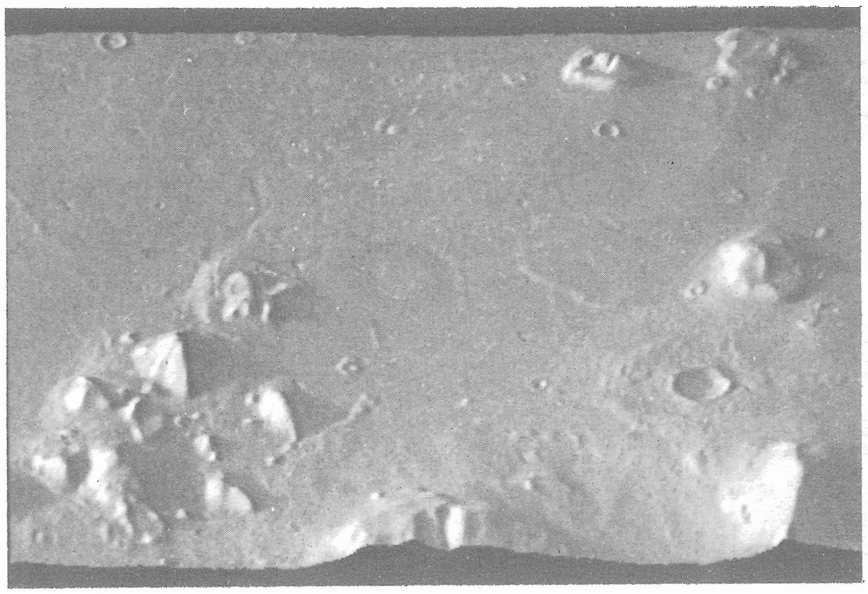


(b)

Fig. 5 Results obtained using Viking Orbiter imagery of Mars.

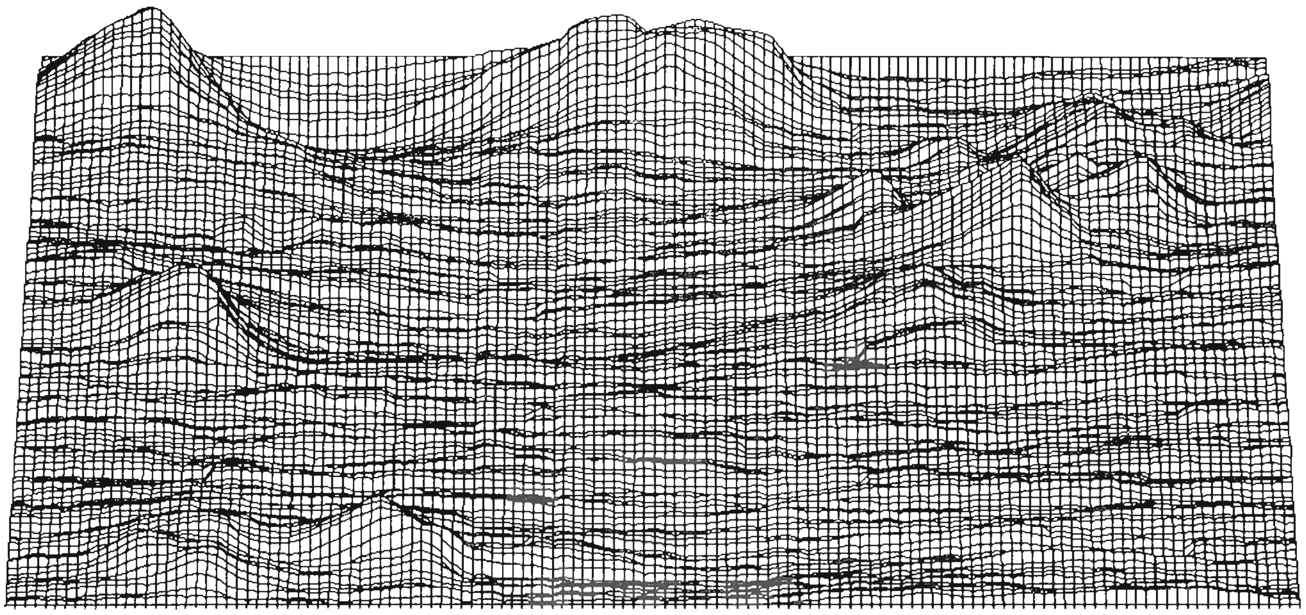


(c)

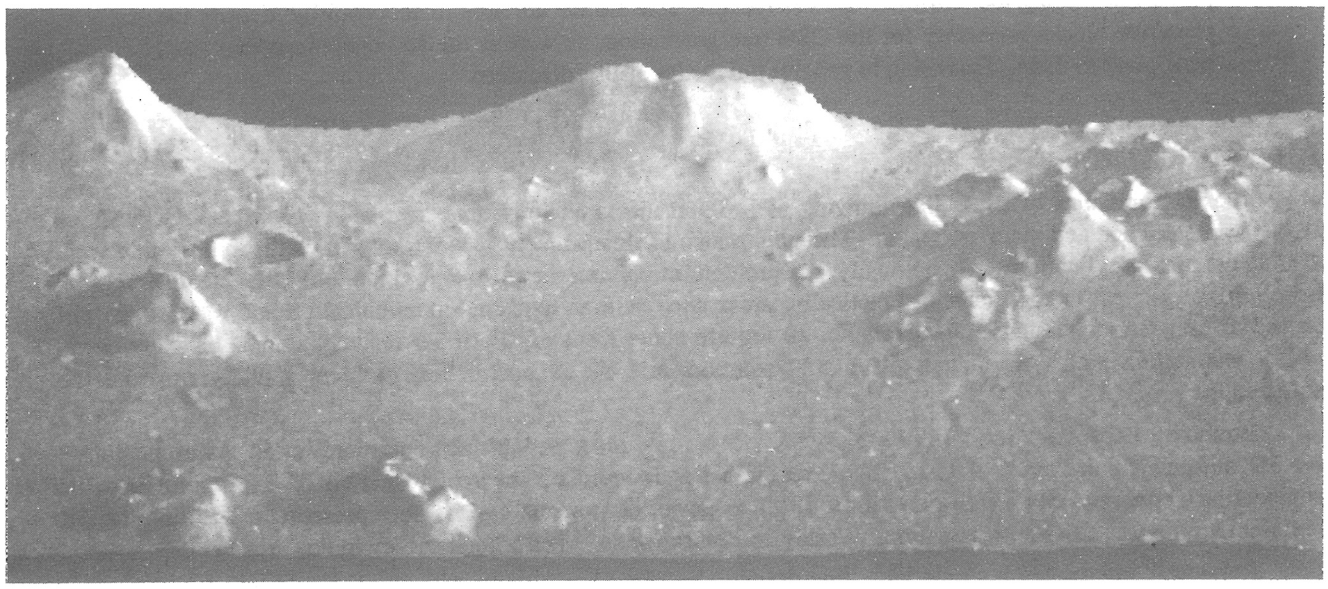


(d)

Fig. 5 Results obtained using Viking Orbiter imagery of Mars.



(e)



(f)

Fig. 5 Results obtained using Viking Orbiter imagery of Mars.