MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

A. I. Memo 463

March 1978

SHADED PERSPECTIVE IMAGES OF TERRAIN

Thomas M. Strat

Abstract. In order to perform image analysis, one must have a thorough understanding of how images are formed. This memo presents an algorithm that produces shaded perspective images of terrain as a vehicle to understanding the fundamentals of image formation. The image is constructed using standard projection equations along with an efficient hidden-surface removal technique. The image intensity is calculated using the reflectance map, a convenient way of describing the surface reflection as a function of surface gradient. Aside from its use as a tool toward understanding image analysis, the algorithm has several applications of its own, including providing video input to a flight simulator.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

TABLE OF CONTENTS

INTRODUCTION

This memo describes an algorithm which produces a shaded perspective view of a digitally defined surface. Hidden surface elimination is done very cheaply by taking advantage of an algorithm demonstrated by Woodham [12]. Historically, attempts at producing projections automatically were hampered by a lack of a fast method for performing hidden surface elimination. The method used, to be referred to as a systematic profile expansion technique, performs the elimination with minimal overhead.

Calculation of intensities at image points is also done cheaply. By utilizing tools developed by Horn [5, 6], the intensity at a point is shown to be a function of the surface normal at that point. Thus, the intensity calculation is reduced to determination of the surface normal followed by calculation of the corresponding reflectance by making use of gradient-space concepts.

The input to SPIT (Shaded Perspective Images of Terrain) is based on a topographic map showing contours of elevation. For efficiency reasons, several contour maps have previously been converted to 2-dimensional arrays of elevations (Digital Terrain Models). The program works directly from these arrays.

The output is a 2-dimensional array of intensities which, when displayed appropriately, form the image. This image is a shaded picture of the terrain defined by the input Digital Terrain Model.

* * *

The following conventions have been adhered to throughout: The terms *picture* and *image* are synonymous and are used interchangeably to refer to the picture that is produced by the program. Similarly, the terms *terrain*, *landscape*, *object*, and *surface* all refer to the landform being drawn. Four different coordinate systems are used:

> *Object space* refers to the input array and is denoted by (x, y, z) where z is the elevation at the point (x, y). The object space axes are referred to as X, Y, Z.
>
> *Eye space* refers to a coordinate system where the origin is at the viewer and is denoted by (x', y', z'). X', Y', Z' are the eye space axes and the positive Y' axis is the direction of view. This is explained in more detail later.
>
> *Image space* refers to the picture that is produced and its coordinates are (i, j). Image space axes are referred to as I and J.
>
> *Gradient-space* is a separate concept and is denoted by (p, q) while the axes are denoted by P and Q.

Two types of projections are commonly used to draw images. *Orthographic Projection* is simpler to compute than perspective. In an orthographic drawing, an object would be drawn the same size regardless of its depth in the image, and a scene would appear as if viewed from a very large distance. *Perspective Projection* accounts for depth by showing distant objects smaller than near objects of the same size. The result is a natural looking image, similar to a photograph.

MOTIVATION

During the past decade, much attention in artificial intelligence has been devoted to determining shape from the shading information in images [7]. The reason for this has been the need for such a capability in vision systems. This memo deals with the inverse problem--that of drawing a shaded image based on information about an object's shape. Although the inverse problem is somewhat easier and has already been partially solved, there are still many challenging applications to be worked out. Furthermore, a thorough understanding of how objects reflect light is a prerequisite to analyzing images of objects.

When considering what kinds of objects to draw, three domains immediately come to mind.

A) Geometric objects: This domain has received much attention in recent years as a part of blocks world analysis. While efforts on geometric objects have been highly successful, there are yet some interesting problems to be worked out. How the objects should be represented and possible tesselations of object surfaces are examples.

B) Mathematical functions: Because of the difficulty humans have in visualizing 3-dimensional functions, this domain has an immediate application in drawing graphs. An algorithm for displaying such graphs as orthographic projections has recently been described by Woodham [13].

C) Landscapes: Until now, this domain has remained uninvestigated. The ease with which a human can understand a drawing as opposed to, say, a contour map is an attractive application of this domain. In fact, this is my primary motivation and the remainder of this memo deals with methods of producing shaded, perspective drawings of terrain.

HISTORY

The idea of using shading in images of terrain is not new. Cartographers have been producing shaded maps to show relief for centuries. Such hill-shading on maps provides for quick comprehension of the topography--an important feature when the interpretor's time is limited (as in aviation, for example).

Why do we see so few shaded maps? One reason is that present manual methods of production are expensive because they require skilled artists with good cartographic insight. These artists use air-brushes in a slow, inaccurate, and tedious operation, working from existing contour maps and sometimes aided by aerial photography.

Horn has investigated this process and has devised a scheme which performs the shading task automatically, by utilizing knowledge about how light is reflected from a surface. This results in a map which is produced automatically from contour maps and is comprehended much more easily than a contour map. Compare Figures 1 and 2.

While the shaded map is a vast improvement over contour maps, we can still do better. Consider a shaded, 3-dimensional drawing of terrain from an arbitrary angle. Such

a drawing which was produced by SPIT is shown in Figure 3. Although this "map" is no longer useful for computing distances, it appears to depict the topography in a manner which humans can ascertain quickly.

FIGURE 1: Contour Map of "Dents de Morcles"

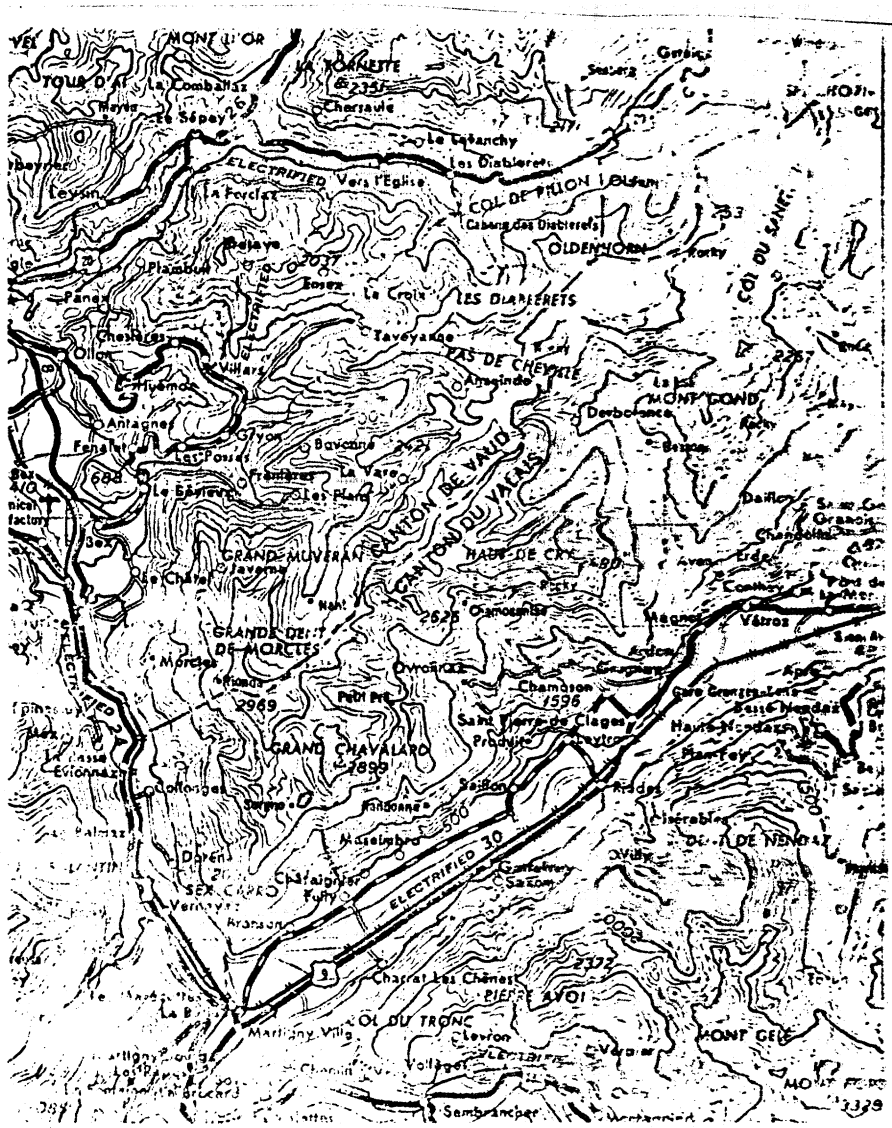


FIGURE 2: Shaded Map of "Dents de Morcles"

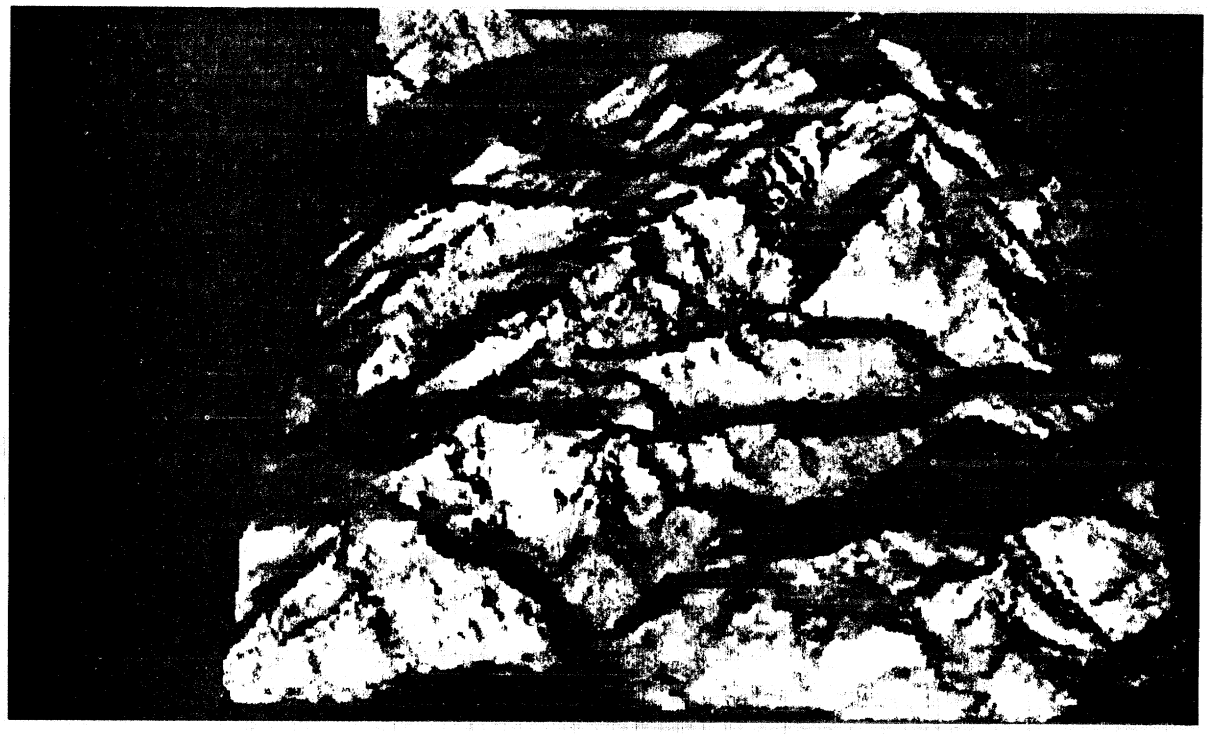


FIGURE 3: Shaded Perspective Image of "Dents de Morcles"

APPLICATIONS

Since we can't use this picture to calculate distances, what good is it? The following is a partial list of potential applications:

A) Visual input to flight simulators. In a flight simulator an image is projected onto the windshield of the 'aircraft' which simulates what a pilot would see in flight. This image is currently produced in one of two ways:

1) As a computer generated line drawing of the airport, towns, lakes, mountains, etc.

2) By a television camera which is being moved appropriately over a large terrain model.

A shaded, computer-generated drawing of terrain might prove to be a way of producing high quality images for a flight simulator.

B) As a supplement to maps in a military operation. When setting up a defensive position or planning an offensive operation, it is vital to have a firm grasp of the topography of the area. As pointed out before, contour maps are difficult to comprehend in a short amount of time. Thus a unit commander normally makes a reconnaissance flight over enemy terrain to learn the topography. A shaded drawing of the terrain would have two advantages:

1) Allow for quick comprehension of the topography--a must in a combat situation.

2) Decrease the need for a risky reconnaissance mission.

C) Astronaut training aids. Flights to the moon and other terrestial bodies are expensive and the time spent there is scarce. Astronauts could be trained with computer-generated pictures so that, upon arrival, they will instantly recognize landing zones and other locations.

D) As an aid to architects. An architect considering the planning of a town or the positioning of buildings needs to fully understand the topography of the area. Using SPIT, he could see the landform from any angle and resolution he desires.

THE ALGORITHM

The algorithm that has been developed is based on that developed by Woodham for displaying mathematical surfaces. The theory behind the algorithm divides naturally into two fundamental parts. One is the transformation from object space to image space while the determination of the appropriate intensity value at each image point is the other. The two calculations are independent and are described in the following two sections.

THE TRANSFORMATION FROM OBJECT SPACE TO IMAGE SPACE

In order to calculate the image point (i, j) which corresponds to a particular surface point (x, y, z) we can think first of transforming the point (x, y, z) from its object space representation into an eye space representation (x', y', z'). The eye coordinate system has its origin fixed at the viewpoint and its positive y' axis pointed in the direction of view. Then we can generate two dimensional display coordinates (i, j) by projecting the eye coordinate point (x', y', z') onto the plane of the display, as shown in figure 4.

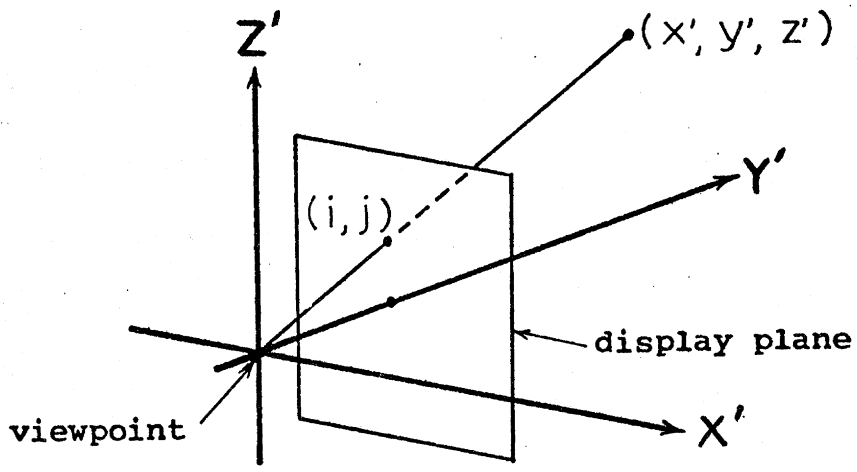


FIGURE 4

As can be seen, a straight line connects the viewpoint, the image point, and the surface point. In fact, this is true for every surface point. Defining the constant, f, to be the separation of the image plane from the viewpoint we have the following relationships

$$i/f = x'/y' \qquad j/f = z'/y'$$

which define the standard perspective projection.

If we assume the viewpoint is far away from the surface, we can approximate y' by k, a constant for all surface points (x', y', z'). Then defining y_o as k/f we have

$$i = x'/y_o \qquad j = z'/y_o$$

which is the standard orthographic projection.

The following equation calculates x', y' and z'.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = M \begin{bmatrix} x-xcen \\ y-ycen \\ z \end{bmatrix} - \begin{bmatrix} 0 \\ R \\ 0 \end{bmatrix}$$

This equation summarizes the shifts and rotations necessary to convert from surface space to eye space. M is the appropriate 3x3 rotation matrix precomputed from the position of the eye. (xcen, ycen) is the center point of the land area to be drawn. R is the distance between the viewer and the center point. The direction of view is y'.

In order to avoid displaying surface points occluded from view by other portions of the surface, information about the depth of each point in the image must be made available. One method would be to associate with each image point (i, j) the depth value y' of the corresponding eye coordinate point (x', y', z'). This approach has several drawbacks.

1) Depth values must be explicitly calculated.
2) Depth values require an additional 2-dimensional storage array.

The need to explicitly calculate depth is eliminated by processing points in the XY plane in a systematic order working away from the viewer. In order for the hidden-line elimination to be correct, one must guarantee that points that map to the same column in the image are visited monotonically away from the viewer, as shown in Figure 5. This process, known as profile expansion, guarantees that a point in the foreground is processed before a point in the background.

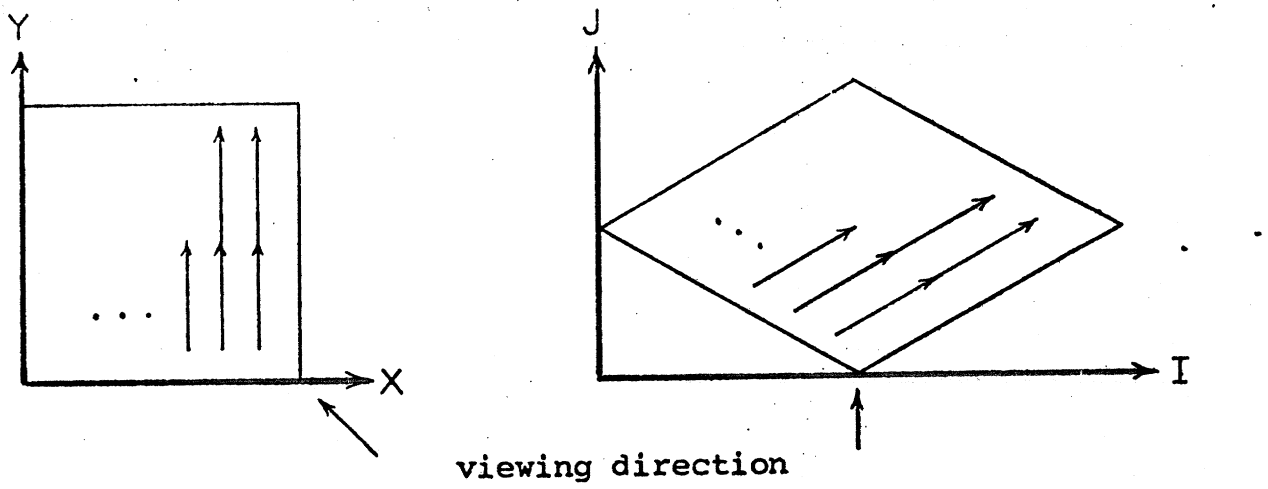


FIGURE 5

Then we can adopt the following rule for hidden surface elimination:

> An image point (i, j) is displayed if and only if its row coordinate j is greater than the largest row coordinate of every other point so far displayed in column i.

Depth information is implicitly represented as a one-dimensional array referred to as the current horizon, holding the largest row coordinate so far displayed in each column. The decision of whether or not to display a particular image point is based on a comparison of its j coordinate and the corresponding value in this array.

GREY LEVEL DETERMINATION

In the preceding section we showed how to determine where to put a point in object space in image space using a perspective projection. Now we will investigate what to put there. Our final goal is to produce a shaded drawing. This drawing can be thought of as an array of dots each of which has the appropriate intensity (called grey-level).

Horn has shown that the amount of light reflected in a certain direction is a function of the properties of the surface at the point of reflection and the geometry of the direction to the light source, the viewing direction, and the surface orientation [6]. In particular, the surface orientation plays a major role. The surface orientation is specified by the vector which is perpendicular to the plane tangent to a point as shown in figure 6. This vector is called the local normal or surface normal.

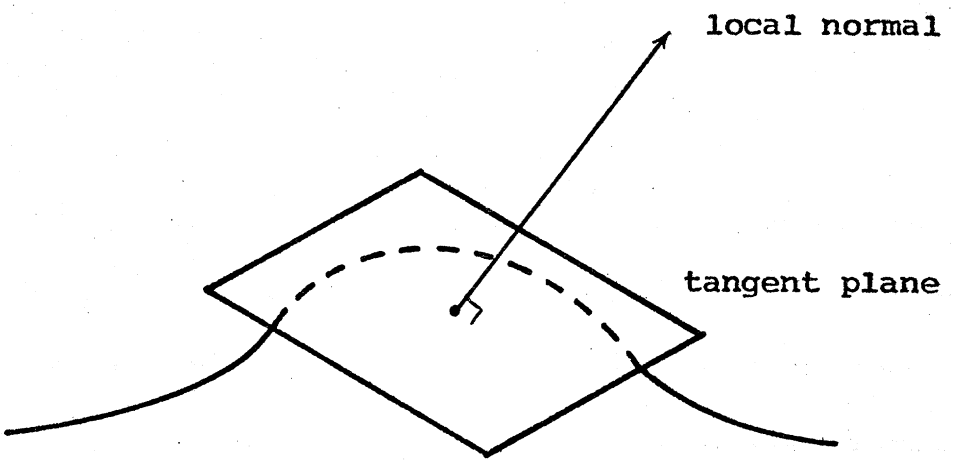


FIGURE 6

If the equation of the surface is given as $z=f(x, y)$ then the equation of the tangent plane at the point (x_1, y_1, z_1) can be written as

$$x(\partial z/\partial x) + y(\partial z/\partial y) - z = x_1(\partial z/\partial x) + y_1(\partial z/\partial y) - z_1.$$

Thus the local normal is given by the vector

$$((\partial z/\partial x), (\partial z/\partial y), -1).$$

For convenience, the first partial derivatives are abbreviated as p and q where $p=\partial z/\partial x$ and $q=\partial z/\partial y$. It is now clear that surface orientation has two degrees of freedom.

This leads us into the concept of gradient-space. The mapping from surface orientation to gradient-space is straightforward. If we construct a normal (p, q, -1) at a point on a surface, it maps into the point (p, q) in gradient-space. Thus, gradient-space is actually a two dimensional plot of the slope of a surface. The origin in gradient-space corresponds to a horizontal plane. Moving along the positive P axis tilts the plane clockwise around the Y axis in object space. Similarly, tilting around the X axis away from the viewer in object space corresponds to increasing q in gradient-space. We'll get back to this later.

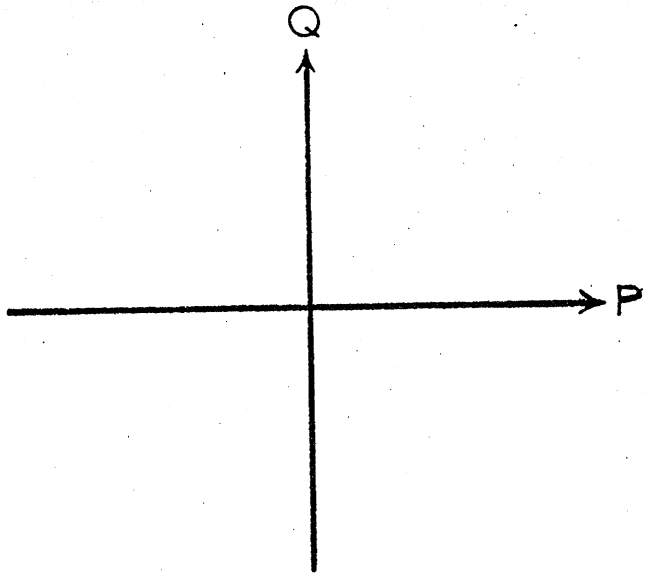


FIGURE 7: Gradient Space

* * *

Now we must examine the source-viewer-surface geometry. Consider the angles i, e, and g as defined in figure 8. These stand for incident, emittant, and phase angles respectively.

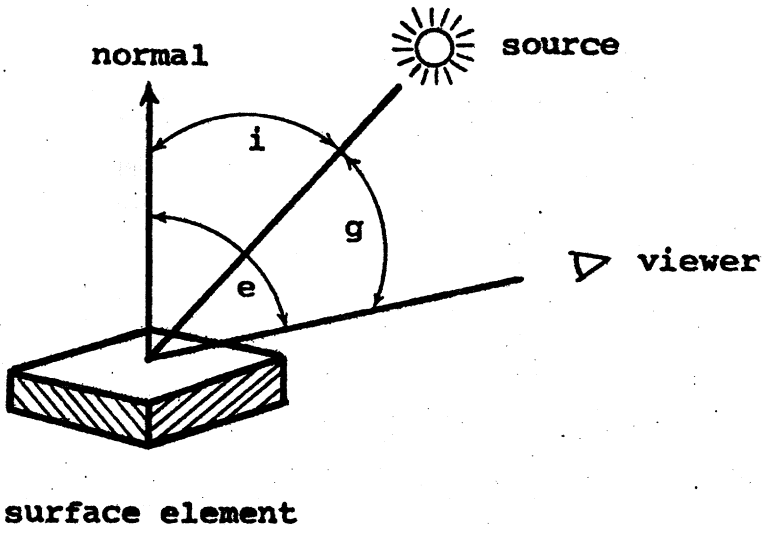


FIGURE 8

The geometry of reflection is governed by these three angles. The reflectance function is a measure of how much of the light incident on a surface element is reflected in a particular direction, and is written as $\phi(i, e, g)$. Let's investigate ϕ further by considering some examples:

Consider a perfect lambertian surface. This perfect diffuser has the property that it looks equally bright from all directions and that the amount of light reflected depends only on the cosine of the incident angle [6]. First we will consider the special case of a single light source near the viewer. The incident angle equals the emmittant angle and is simply the angle between the surface normal and the view-vector. If we plot reflectance as a function of p and q in gradient space, we get a central maximum of one at the origin

and a circularly symmetric function that monotonically falls to zero as one goes to infinity in gradient space. See figure 9.

The mathematics of many other reflectance functions have been worked out, mostly due to Horn. It is not the purpose of this paper to explain these derivations, so I will simply present without justification some that I have found appropriate.

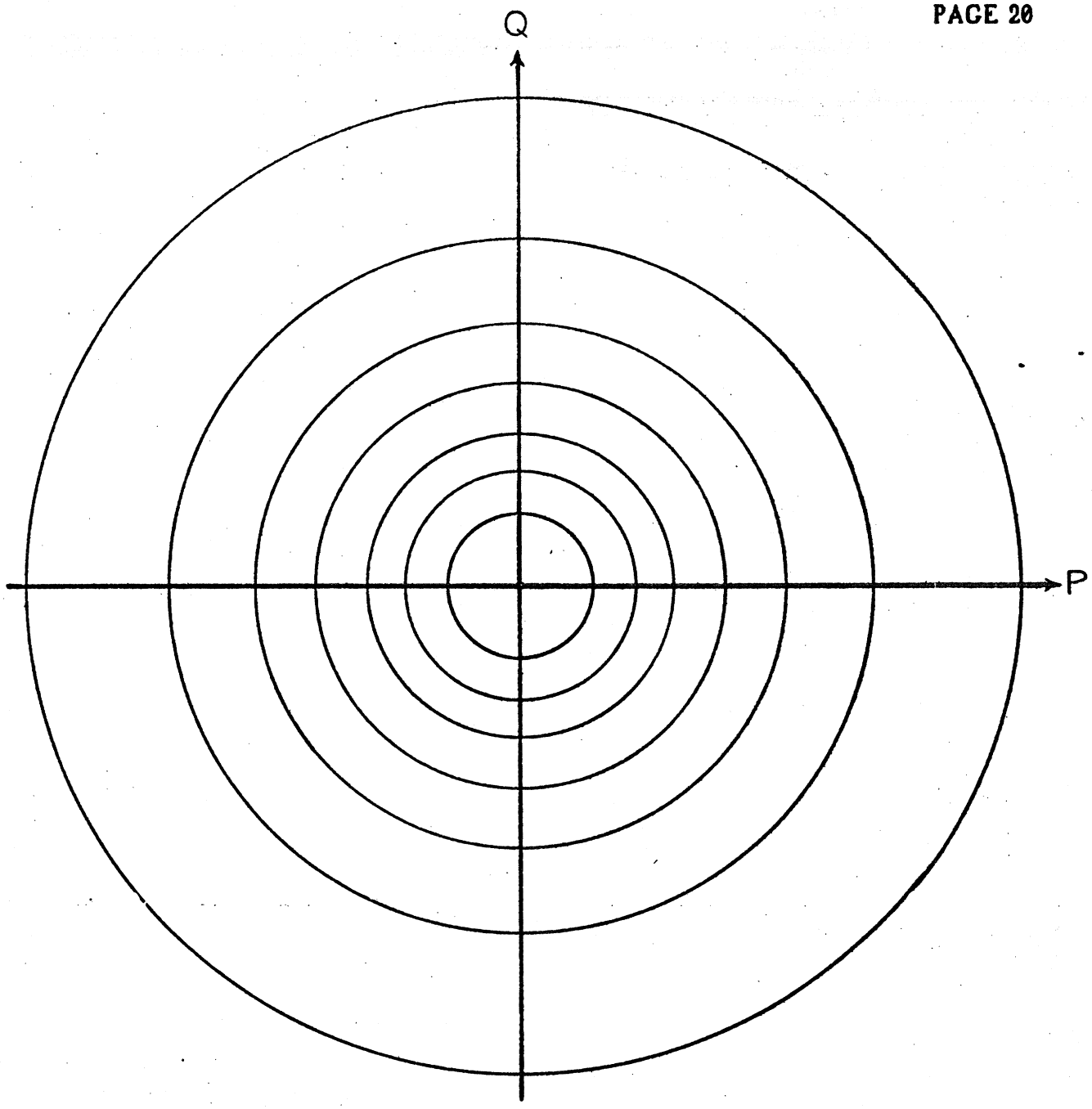


FIGURE 9: Contours of constant I = cos(i). Contour intervals are 0.1 wide. This is the gradient space image for objects with lambertian surfaces when there is a single light source near the viewer.

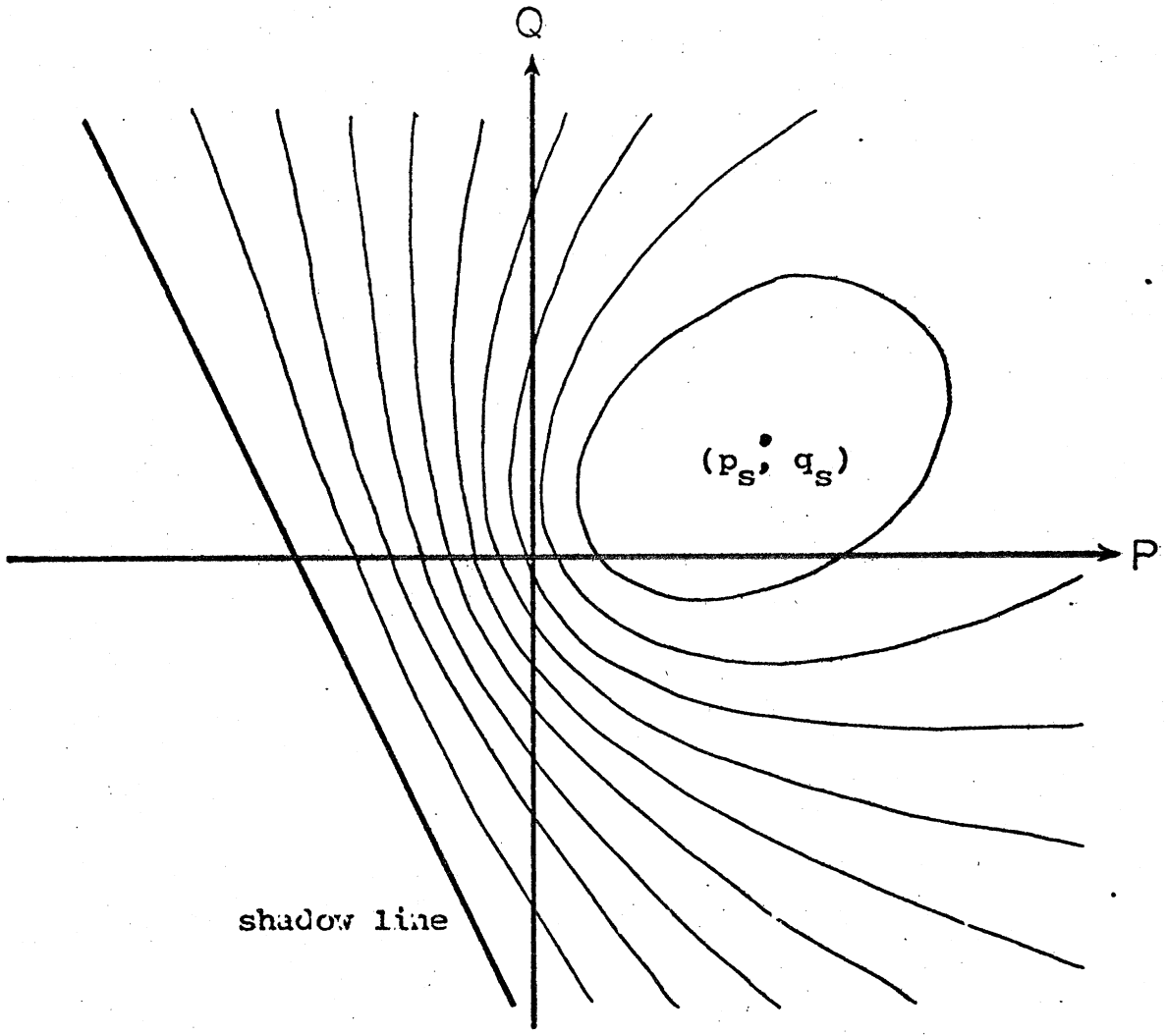


FIGURE 10: Contours of constant I = cos(i). Contour intervals are 0.1 wide. The direction to the source is $(p_s, q_s) = (.7, .3)$. This is the gradient space image for objects with lambertian surfaces when the light source is not near the viewer.

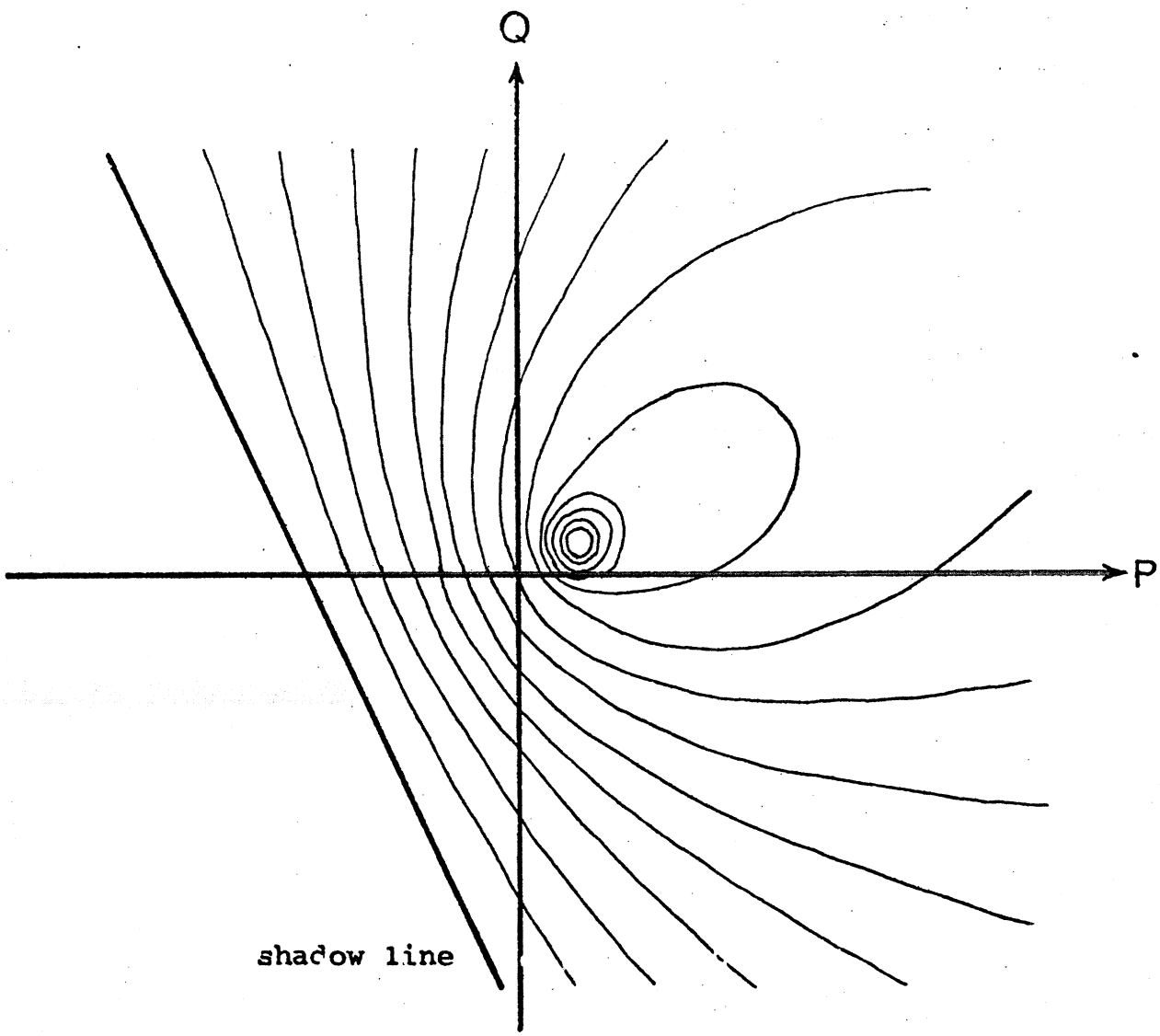


FIGURE 11: This is the gradient space image for a surface with both a matte and a specular component of reflectivity illuminated by a single point source not near the viewer.

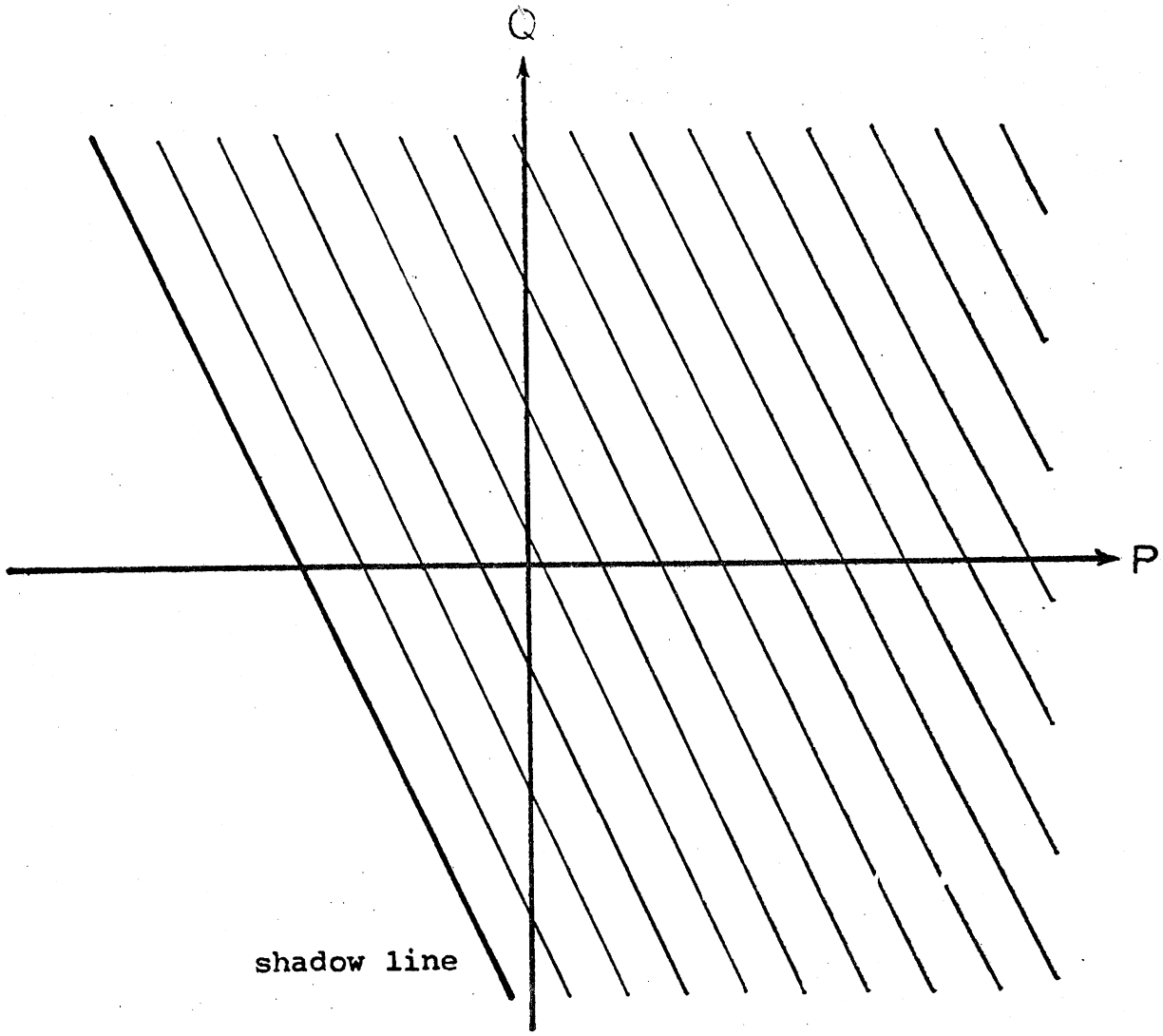


FIGURE 12: Contour intervals are 0.2 units wide. This is the reflectance function for the material in the maria of the moon when viewed from the earth.

Getting back to the problem of producing shaded drawings we now have the tools to select an appropriate image intensity value at each point in the image. After selecting a particular reflectance function which is believed to accurately model the surface characteristics, the surface normal must be determined at the point in question. The fraction of light reflected at that point is simply the value of the reflectance function which corresponds to the surface normal at that point. This intensity is then displayed at the image point calculated by the projection described in the preceding section and we have our shaded drawing.

Almost. . .

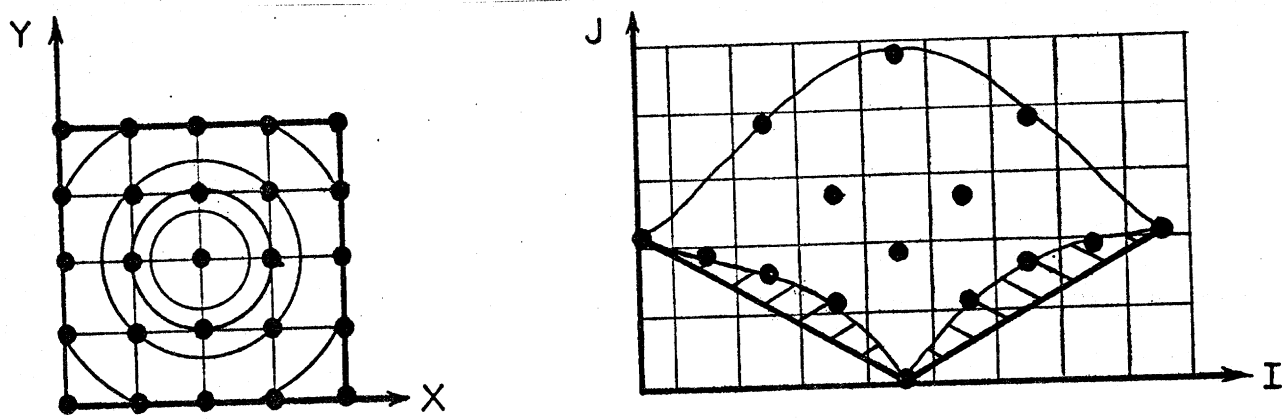
INTERPOLATION

In order to produce a shaded drawing, an intensity must be calculated for every point in the image. This far, however, we have simply mapped each point on the surface into an image point. This certainly does not guarantee that we have filled in all image points. In fact, it most certainly will not be the case. Figure 13 illustrates this point.

We might try circumventing this problem by sequentially exploring points in image space rather than in object space. This would certainly eliminate the need for interpolation. It does not work because each point in image space corresponds to the infinite number of points in object space whose locus is the straight line starting at the viewer and passing through the plane of display at the image point, as shown previously in Figure 4. We would then have the difficult task of finding the first surface point which

falls on this line.

Note that each point in object space corresponds to a unique point in image space, eliminating the need for further computation. It is for this reason that I chose to sequentially explore points in object space and must now worry about interpolating between missed image points.



OBJECT SPACE
CONTOUR MAP

IMAGE SPACE

FIGURE 13

What is to be done? Three solutions are offered in order of decreasing complexity. The second is the one actually implemented.

1) Simply expand profiles at such a high resolution that no image point will be missed. At best, this solution would be so computationally wasteful as to defeat the simplicity of the technique.

2) Find intermediate image points by interpolating in the XY plane whenever a point in the image plane is missed. This solution has minimal overhead and will compute the correct intensity values provided that there be no more than one local extremum of elevation between the two points in object space. This is fairly simple and works quite well.

3) Interpolate between known intensity values in image space. This method will compute incorrect intensity values because the points between which one interpolates typically cross real depth discontinuities.

Because of the size disparity due to depth in a perspective image, it is sensible to sample at a fine resolution when projecting points near the viewer and at a coarser resolution when working far from the viewer. A simple heuristic is used to account for this. When selecting points in the XY plane to be processed, a guess is made as to which point (x, y) will project into the adjacent pixel. If the guess is good, we need not interpolate for missed image points nor waste time projecting two surface points into the same pixel.

The following equations predict the point by assuming perfectly level terrain.

$$x_{n+1} = x_n \pm y'_n/F$$

$$y_{n+1} = y_n \pm y'_n/F$$

Since the terrain is not always level, we still need to interpolate occassionally.

HOW TO USE SPIT

This section is a brief user's manual to the program. The algorithm allows many degrees of freedom in specifying the image. Presently, all of the following are provided for.

1) Selecting the terrain to be drawn.
2) Selecting the reflectance function.
3) Moving the light source with both degrees of freedom.
4) Changing the dimensions of the image to be produced.
5) Shifting the terrain in X and Y.
6) Specifying the terrain sampling resolution.
7) Zooming in on a point on the ground and moving back out.
8) Stretching in X or Y (worthless).
9) Stretching in Z (interesting).
10) Changing the view angle with all three degrees of freedom.

The algorithm currently resides in compiled form on AI ITS @ MIT as

DSK: TMS; SPIT FASL

The compiled version of SPIT is loaded into the LISP environment by typing

(FASLOAD SPIT FASL DSK TMS)

This in turn loads several other files which are necessary for display purposes.

Next one must load the Digital Terrain Model to be used as input. This is accomplished by typing

(LOAD-DTM DTM xxx DSK WH)

where xxx is selected from the set

{BOW, DENT, DIABLO, GULF, TEHA}

This function also sets the name of the terrain, the dimensions, the scale and several other parameters.

Before attempting to construct the picture, many parameters must be initialized to specify the exact picture desired.

(DIMENSION)

accomplishes this by asking the user for each one in turn. XLO, YLO, XHI, YHI are the four corners of the land area to be drawn. EYE-AZIMUTH, EYE-ELEVATION, and R specify the viewer location in spherical coordinates. F is the constant of proportionality of the perspective projection. SUN-AZIMUTH and SUN-ELEVATION define the direction to the light source in spherical coordinates. The source is assumed to be an infinite distance from the surface. The viewer location can alternatively be specified in Cartesian coordinates by typing

(EYE).

Currently, only two reflectance functions are available. These are set by typing either

(LAMBERT) or (GLOSSY).

The default is Lambertian.

At this point the picture can be computed by calling the main function

(SHADE).

When SHADE is through, it prints out the values of the parameters that were used. To

prepare this image for the photowriter, use

(DUMP-IMAGE filespecs).

This results in the creation of a file ready to be photowritten by PW.

ALIASING

The problem of aliasing (under-sampling the data) is a fundamental concern of all automatic shading algorithms. One must be extra-cautious when using SPIT because the data (the surface) is sampled twice--once when the contour map is digitized and again when the DTM is sampled.

Crow [4] points out that characteristic aliasing problems occur (1) along edges and silhouettes of objects in the form of "jaggies", (2) in very small objects, and (3) in areas of complicated detail. For the purpose of images created by SPIT, the third type of aliasing can be ignored because the DTM cannot contain areas of complicated detail. Aliasing due to very small objects can occur only if they appear in the DTM but get "missed" in the construction of the image. This problem can be avoided by constructing the image at a fine enough resolution so that no points in the DTM are missed. While this is not computationally unfeasible (most DTM's are at a coarse resolution as it is), there are times when it is inconvenient. SPIT handles this problem adequately by interpolating within the DTM in a way that averages surrounding points--thus no point in the DTM will ever be missed entirely and no aliasing problems of the second type can occur.

On the other hand, jaggies are a problem as evidenced along ridges and horizons in

some of the images in the appendix. SPIT has no provision for minimizing jaggies and further study is required to eliminate them. An algorithm which attenuates jaggies successfully is given by Crow [4].

SUGGESTIONS FOR FURTHER WORK

In order to provide realistic visual input to a flight simulator, images must be produced at a rate of 30 frames per second. Current implementations in compiled LISP provide for perspective pictures in 60 seconds and orthographic pictures in 10 seconds. Certainly this must be improved upon if the system is to be feasible for this purpose. Writing the program in assembly language could result in speedups of a factor of ten--still not quite good enough. The key improvement can come from the parallelism inherent in the algorithm. Notice that each surface point can be projected and its intensity calculated independently of all other points. It has been shown [10] that exploiting this paralellism by using a Data Flow Computer can theoretically result in computation times on the order of hundredths of a second per frame.

I'm not convinced that the lambertian reflectance function yields the best looking pictures. Further research is necessary in this area to determine better ones. Possibly one that makes only the tops of hills very bright or one that simulates several light sources might be better.

Mutual illumination is a topic that is not well understood, especially for complicated surfaces like landscapes. The effects of light reflected from adjacent hills and

light that has been scattered by the atmosphere can cause differences between real images and synthetic ones. For better realism, more work needs to be done here.

Humans expect to find shadows in pictures. Images currently produced by SPIT contain only self-shadows. The addition of cast shadows to the algorithm is currently being studied by Woodham and the author.

For the purpose of flight simulators, it might be desirable to include such objects as runways, trees and buildings in the image. Such additions to the image are possible using techniques developed by James Blinn and Martin Newell at the University of Utah and promise to yield much more realistic-looking pictures [1]

REFERENCES

1. Blinn, James F., and Newell, Martin E., "Texture and Reflection in Computer Generated Images," Communications of the ACM, Vol. 19, No. 10, October 1976.

2. Bui Tuong Phong, "Illumination for Computer Generated Pictures," CACM, Vol. 18, No. 6, June 1975.

3. Catmull, E., "A Subdivision Algorithm for Computer Display of Curved Surfaces," UTEC - CSc - 74 - 133, University of Utah, December 1974.

4. Crow, Franklin C., "The Aliasing Problem in Computer-Generated Shaded Images," Communications of the ACM, Vol. 20, No. 11, November 1977.

5. Eastman, Jeffrey F., "An Efficient Scan Conversion and Hidden Surface Removal Algorithm," Computers & Graphics, Vol 1, pp. 215-220, Pergammon Press, 1975.

6. Horn, Berthold K. P., "Automatic Hill Shading Using the Reflectance Map," unpublished, 1976.

7. Horn, Berthold K. P., "Understanding Image Intensities," Artificial Intelligence, North-Holland Publishing Co., 1977.

8. Horn, Berthold K. P., "Obtaining Shape from Shading Information," in The Psychology of Machine Vision, ed. by P. H. Winston, McGraw-Hill, 1975, pp. 115-155.

9. Murai, S., Ohbayashi, S., and Tateishi, R., "Three Dimensional Representation of Landscape," Journal of Institute of Industrial Science, Vol. 27, No. 5, University of Tokyo.

10. Staudhammer, John and Ogden, Deborah J., "Computer Graphics for Half-tone Three-Dimensional Object Images," Computers & Graphics, Vol. 1, pp. 109-114, Pergammon Press, 1975.

11. Strat, Thomas M., "Application of Data Flow Computer Architecture to the Shaded Image Problem," unpublished, 1977.

12. Sutherland, I. E., Sproull, R. F., and Schumacker, R. A., "A Characterization of Ten Hidden Surface Algorithms," Computing Surveys, Vol. 6, No. 1, March 1974.

13. Weiss, R. A., "BE VISION - A Package of IBM 7090 FORTRAN Programs to Draw Orthographic Views of Combinations of Plane and Quadric Surfaces," Journal of ACM, Vol. 13, No. 2, April 1966, pp. 194-204.

14. Woodham, Robert J., "Two Simple Algorithms for Displaying Orthographic Projections of Surfaces," MIT Working Paper 126, August 1976.

APPENDIX

The following pages contain samples of images produced by SPIT. They were selected to depict the shortcomings as well as the various features of SPIT. All were produced using the Optronics International, Inc. Photowriter with WH;PWMAPS CTP4 as the mapping function.