Orthographic Terrain Views Using Data Derived from Digital Elevation Models

Ralph O. Dubayah and *Jeff Dozier*
Department of Geography, University of California, Santa Barbara, CA 93106

ABSTRACT: A fast algorithm for producing three-dimensional orthographic terrain views uses digital elevation data and co-registered imagery. These views are created using projective geometry and are designed for display on high-resolution raster graphics devices. The algorthm's effectiveness is achieved by (1) the implementation of two efficient grey-level interpolation routines that offer the user a choice between speed and smoothness, and (2) a unique visible surface determination procedure based on horizon angles derived from the elevation data set.

INTRODUCTION

THIS PAPER EXPLORES an algorithm for producing orthographic views of terrain using digital elevation data and co-registered imagery. The algorithm is uniquely characterized by its use of horizon angles, determined from the elevation data set, to determine surface visibility (Dozier *et al.*, 1981).

The terrain surface is defined by a grid of digital elevation values stored in raster format, which are transformed and projected onto a viewing plane. Associated with each of these points is a grey value, or perhaps multispectral values, from the co-registered input image, which can be a synthetically generated shaded relief image or a remotely sensed image as shown in Plates 1 and 2 (Horn and Bachman, 1978; Horn, 1981; Batson *et al.*, 1975). The end result is an arbitrary aspect view of the image such as that shown in Plate 3. These views are simple and fast to create, require virtually no user inputs other than elevation files, digital images, and viewing angles, and have a variety of applications for Earth scientists.

The initial motivation for this algorithm was the need to create orthographic terrain views for our image processing environment suitable for display on a high resolution output device. We wanted an algorithm that was easy to implement, easy to use, ran quickly, and produced smooth, visually pleasing images of terrain regardless of view orientation. Such features as windowing and zooming were not required because most image processing systems, our included, already have these fundamental capabilities. Likewise, it was not necessary for the program to generate synthetic reflectance images from elevation data, because this is best accomplished by a separate program.

GENERATING THREE-DIMENSIONAL IMAGES

RAY TRACING VERSUS FORWARD PROJECTIVE GEOMETRY

Given these requirements of simplicity, and smoothness, what methods can be used? Several programs have been written which deal specifically with creating orthographic and perspective views of terrain (Faintich, 1974; Bunker and Heartz, 1976; Woodham, 1976; Strat, 1978; Dungan, 1979; Scholz *et al.*, 1984). In addition to these, there has recently been an explosion in computer graphics research aimed at producing realistic scenes, regardless of the objects involved. The techniques used can be divided into two broad categories: those which use ray tracing and those which use forward projective geometry.

Ray tracing is a method in which rays are traced from the viewpoint, through the viewing screen and into the object being viewed. It has evolved to deal with increasingly complex computer graphics scenes involving multiple reflections and refractions (Rubin and Whited, 1980; Catmull and Smith, 1980; Glassner, 1984). Ray tracing essentially maps from the two-dimensional plane of the view screen to the three-dimensional object world. For non-parametrically defined surfaces, most ray tracing algorithms involve some kind of search procedure, because this mapping from two to three dimensions is not unique.

As applied to terrain viewing, ray tracing techniques are slow unless programmed in hardware. Topographic surfaces are generally not defined parametrically and, therefore, search is necessary. For example, one typical technique involves defining the ray through the particular view screen pixel in terms of object space coordinates and then searching along this line until an input pixel is found whose elevation is greater than or equal to the elevation of the line at that point (Dungan, 1979). This is a time-consuming procedure for large data sets.

PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING,
Vol. 52, No. 4, April 1986, pp. 509–518.

0099-1112/86/5204-509$02.25/0

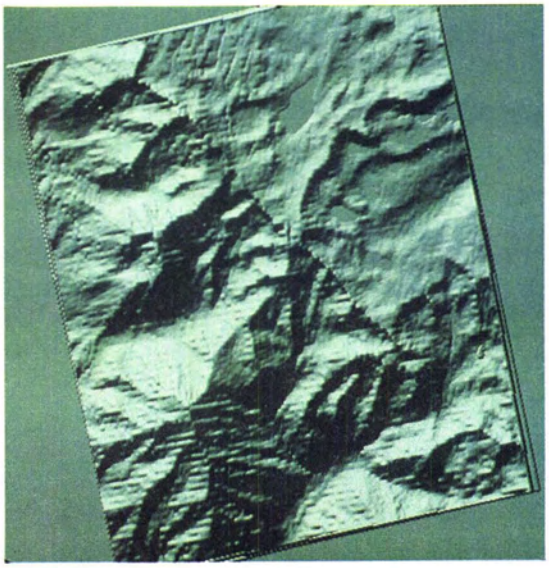

PLATE. 1. A synthetic shaded relief image of the Mt. Tom, California 7.5-min quadrangle, created from an elevation grid with 30-m resolution. The sun's position is 32° east of south and 25° above the horizon. The striping in the image results from noise in the USGS Digital Elevation Models. The elevation grid is registered to the TM images in Plates 2 and 3. The SOM projection is rotated slightly from north.

PLATE. 2. False color composite of the Mt. Tom area shown in Plate 1 using TM bands in green, red, and near-infrared wavelengths (bands 2, 3, and 4). The sun's altitude and azimuth are identical to those in Plate 1.

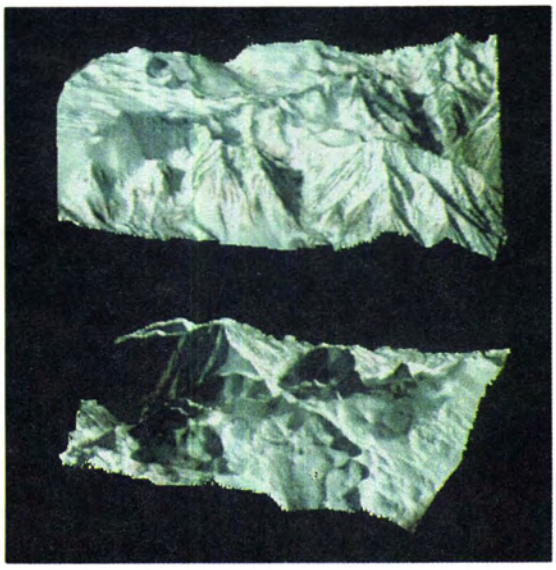

PLATE. 3. Oblique orthographic views of Plate 2. In the top picture, the view angle is 55° from zenith and 100° east of south. In the bottom picture, the zenith viewing angle is 60° and the azimuth 65° west of south.

The second class of procedures, labeled here as forward projective geometric techniques, map from three-dimensional object space to the two-dimensional view plane. The object is first reoriented based on geometric transformations and then projected onto the viewing plane. The major disadvantage of this technique is that grey levels must be interpolated in the viewing plane, as described later. Most commercially available computer graphics packages use this technique because it is easy to program, involves no time consuming calculations, and is fairly flexible (Foley and Van Dam, 1982). Most are oriented towards point-line representations of surfaces, and have such features as windowing, clipping, zooming, and perspective ability which are unnecessary for our application.

A few programs have been developed specifically for creating perspective images of terrain, mainly for military purposes (Bunker and Heartz, 1976; Strat, 1978; Dungan, 1979; Scholz *et al.*, 1984). Because they have been generalized to handle the perspective problem, they must use generalized hidden surface removal techniques, which involve either some type of sorting, depth buffering, or ordered profile expansion, none of which take advantage of the simpler viewing geometry of parallel projections.

In choosing among these projective geometric alternatives, we recognized two facts. First, we already use "horizon angles" within our image processing system for irradiance calculations in mountainous terrain. For each point in an elevation grid, the angle to the local horizon for one or more specified azimuths is determined. This information can be exploited to quickly and efficiently determine which input pixels are visible. Second, the smooth-

ness of the final output image is a function of the grey-level interpolation technique used, which in turn affects the processing speed. Because smooth interpolation functions are computationally more expensive, the algorithm offers the user a choice between speed and smoothness.

The Algorithm

Based on the above considerations, a forward projective geometric algorithm has been designed that is different from those cited above in that it uses a horizon-based visible surface test, and includes two interpolation functions: a very fast integerization procedure and a reasonably fast distance weighted routine utilizing lookup tables. The final results reflect the success of the techniques employed here.

The theory behind the algorithm naturally divides into four parts: (1) geometric transformations, (2) parallel projections, (3) visible surface determination, and (4) image reconstruction on raster devices. The material presented in each of these sections is important for giving the user an overall understanding of the image formation process in general, as well as outlining the specifics of the algorithm. Before proceeding, a short overview of the algorithm is presented.

Three co-registered input image files are required: an elevation image, an image to be viewed, and an image of the horizon angles in the viewing direction, which has been previously calculated from the elevation data. In addition, azimuthal and zenith view angles are specified. Optionally, the type of grey-level interpolation is also given: nearest-neighbor or distance-weighted resampling.

The three image files are sent to an orthographic transformation subroutine. Based on the horizon image, the subroutine only transofrms those points that are visible. The transformed x and y coordinates are normalized into the range $0 \rightarrow 1$, and then scaled to non-integer device coordinates. Finally, the grey levels associated with the non-integer device coordinates are interpolated to integer device locations and written to the output file. The final size of the output image is determined by the size of the input image and the viewing angles.

GEOMETRIC TRANFORMATIONS

Homogeneous Coordinates

Homogeneous coordinate representation is often used for projective geometry transformations because it allows translations, scalings, and rotations to be treated uniformly as matrix multiplications (Rogers and Adams, 1976). In a homogeneous coordinate representation an n-component position vector is represented by an $n + 1$ component vector. For example, the vector $[x\ y\ z]$is represented by homogeneous coordinates in 4-space as $[x\ y\ z\ 1]$. The transformation of this vector in 4-space is given by

$$[X\ Y\ Z\ H] = [x\ y\ z\ 1]\mathbf{T}_4$$

where $[X\ Y\ Z\ H]$ are the transformed homogeneous coordinates and $\mathbf{T}_4$ some 4×4 transformation matrix. The transformed regular coordinates are then

$$[x'\ y'\ z'\ 1] = \left[\frac{X}{H}\ \frac{Y}{H}\ \frac{Z}{H}\ \frac{H}{H}\right]$$

If $H = 1$, such as for affine transformations, then $x' = X$, $y' = Y$, and $z' = Z$.

The Generalized 4×4 Transformation Matrix

The matrix T_4 can be used to perform the linear geometric transformations of rotation, translation, local and overall scaling, reflection, shearing, and perspective transformations.

$$\mathbf{T}_4 = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_{11} \\ a_{21} & a_{22} & a_{23} & b_{21} \\ a_{31} & a_{32} & a_{33} & b_{31} \\ c_{11} & c_{12} & c_{13} & d_{11} \end{bmatrix}$$

$\mathbf{T}_4$ can be partitioned into four sections. The 3×3 submatrix containing the a_{ij} elements performs linear transformations such as scaling and rotation. The 1×3 submatrix containing the c_{ij} elements handles translation. The 3×1 matrix with elements b_{ij} is used for perspective transformation and the element d_{11} controls overall scaling.

$\mathbf{T}_4$ is best understood by decomposing it into a set of 4×4 transformation matrices which can then be multiplied to produce the general 4×4 matrix.

Three-Dimensional Rotations

Given a right-handed coordinate system, a positive rotation about an axis is defined such that when looking along the axis of rotation towards the origin, the direction of rotation of the other axes is counterclockwise. The transformation matrices for rotations of angles θ, ϕ, and ψ about the x, y, and z axes (see Figure 1) are given in homogeneous coordinates by

$$\mathbf{T}_{\text{rot}}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{\text{rot}}(\phi) = \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{\text{rot}}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

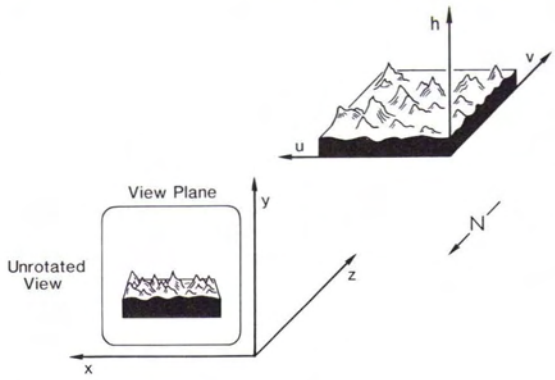

FIG. 1. A depiction of the image formed with no rotation or translation about the object space axes. Notice that the origin of the elevation grid is defined to be the northwest corner of the grid.

In general, only two rotations are performed for terrain viewing: by angle ϕ in the azimuthal direction about a vertical axis and by angle θ about the horizontal axis parallel to the projection plane (*i.e.*, in altitude). Rotation about the axis perpendicular to the projection plane tilts the terrain block.

SCALING

Global and local (axis specific) sccaling is controlled by the diagonal terms of the 4×4 transformation matrix. Given a point $[x\ y\ z\ 1]$ in homogeneous coordinates, its scaled homogeneous coordinates are obtained by multiplying by

$$\mathbf{T}_{\text{scale}} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{bmatrix}$$

where a, b, and c are the local scale factors and d the overall scale factor. Note that vertical exaggeration is possible by either increasing the local vertical scale factor or decreasing the local horizontal factors.

TRANSLATION

Any position vector can be arbitrarily positioned using a translation matrix with non-zero elements in the last row. For example, the point $[x\ y\ z]$ is translated to $[(x + \Delta x)\ (y + \Delta y)\ (z + \Delta z)]$ by multiplying the point, in homogeneous coordinates, by a translation matrix, $\mathbf{T}_{\text{trans}}$. That is,

$$T_{\text{trans}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix}$$

AFFINE TRANSFORMATIONS

If the last column of a transformation matrix is [0, 0, 0, 1], then that matrix is said to produce an affine transformation such that straight lines in the original image are straight in the transformation. The product of two affine tranformations is itself affine.

The rotation, scaling, and translation matrices can be mutliplied into one general 4×4 transformation matrix:

$$\mathbf{T}_4 = \mathbf{T}_{\text{rot}}(\theta)\mathbf{T}_{\text{rot}}(\phi)\mathbf{T}_{\text{rot}}(\psi)\mathbf{T}_{\text{trans}}\mathbf{T}_{\text{scale}}$$

Other transformation matrices may be multiplied to obtain a different $\mathbf{T}_4$, noting that matrix multiplication is not associative. For the terrain views generated by the algorithm presented here only scaling, rotation, and translation transformations are used. Once the transformed homogenous coordinates are obtained, they can be projected onto a viewing plane using one of the projections discussed later.

In theory, a point in object space can be transformed and projected to a location specified in device units in the viewing plane by multiplication with this one general transformation matrix and a projection matrix (Foley and Van Dam, 1982). In practice, this is possible only if the size of the output image is known *a priori*. The program presented here dynamically adjusts the size of the output image based on the rotation angles to assure that no clipping takes place and that there is no pixel undersampling (for subsequent grey-level interpolation). Because there is no windowing, the range of the output image coordinates are not known until after the input image is transformed. Thus, object space coordinates are not transformed and translated to their correct device space coordinates simultaneously. This method causes slight loss of efficiency, but the advantage is that it guarantees the user a smooth terrain view with no unwanted and unexpected windowing.

PARALLEL PROJECTIONS

The three-dimensional coordinates representing the image to be displayed and the elevation grid are transformed using an orthogonal transformation consisting of rotations, perhaps translations, and perhaps scaling. The transformed three-dimensional coordinates are then projected onto a viewing plane by an orthographic projection, through mutliplication with the projection matrix.

If $[x\ y\ z\ 1]$ represents a transformed point in homogenous coordinates, the coordinates of the point when projected onto the $x = k$ plane are

$$[x\ y\ z\ 1]\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ k & 0 & 0 & 1 \end{bmatrix} = [k\ y\ z\ 1]$$

The x coordinate of each point is equal to k as expected. In practice, the viewing plane is often the zero plane of an axis.

Before an input image can be transformed, it is necessary to relate the real-world or object space

coordinates of the input image to the viewing space coordinates of the output image. Images with coordinates given in line and sample numbers often have their origin located at the top left hand corner. How one chooses the default orientation of the input terrain image is somewhat arbitrary. For right-handed view space coordinates it is customary to label the vertical axis of the view plane y, the horizontal axis x, and the axis perpendicular to the view plane z. The origin of the view space is then located at the bottom right corner of the view plane with y increasing upwards, x increasing to the left, and z increasing into the view plane (see Figure 1).

Given an object space coordinate system of $[u\ v\ h]$, the default orientation is such that u is parallel to x, v is parallel to z, and h is parallel to y. The viewing plane is the $[x\ y]$ plane. The default view with no rotation or translation of the object space coordinates is, therfore, the horizontal view shown in Figure 1. The object is being viewed from the north on the horizon. For terrain views the convention is to specify a viewing angle (0° at normal, 90° at horizon) and an azimuth (±180° with 0° being south and positive towards east). With this relation between object and viewing space, the axes of rotation are h and u, with angles of rotation ϕ and θ, respectively. Rotations about v will tilt the terrain view unnaturally.

A rotation about h followed by a rotation about the u axis is performed by using $\mathbf{T}_{rot}(\phi)$ and $\mathbf{T}_{rot}(\theta)$. That is,

$$[u\ h\ v\ 1]\begin{bmatrix}\cos\phi & 0 & -\sin\phi & 0\\ 0 & 1 & 0 & 0\\ \sin\phi & 0 & \cos\phi & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & -\sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$= [u\ h\ v\ 1]\begin{bmatrix}\cos\phi & \sin\phi\sin\phi & -\sin\phi\cos\phi & 0\\ 0 & \cos\phi & \sin\phi & 0\\ \sin\phi & -\cos\phi\sin\phi & \cos\phi\cos\phi & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

In algebraic form, the transformed regular coordinates are

$$x' = u\cos\phi + v\sin\phi$$
$$y' = u\sin\phi\sin\theta + h\cos\theta - v\cos\phi\sin\theta$$
$$z' = -u\sin\phi\cos\theta + h\sin\theta + v\cos\phi\cos\theta$$

Projection onto the viewing plane $z = 0$ is accomplished by multiplying these transformed coordinates by the appropriate projection matrix:

$$[x'\ y'\ z'\ 1]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} = [x'\ y'\ 0\ 1]$$

Notice that multiplication by the projection matrix is unnecessary for this parallel projection. Also, the transformed coordinate z' does not need to be calculated at all. Frequently, this value is saved because it provides depth information in the viewing space and can be used for hidden surface removal. The visible surface algorithm used here does not require this depth information because visible pixels are determined before the transformation process begins.

THE HORIZON TEST FOR VISIBLE PIXELS

A very simple test for pixel visibility based on the horizon angles exists for parallel projections of terrain. For every point in an elevation grid, the angle to the horizon H_ϕ in the viewing azimuth ϕ is calculated by a fast method developed by Dozier *et al* (1981). A pixel is visible from the viewing direction if the viewing angle θ (from zenith) is above the horizon angle H_ϕ (from horizontal), i.e., if $H_\phi < 90° - \theta$.

Speed and efficiency are the main advantages of the horizon test. Each pixel is tested only once and only visible pixels are transformed. The horizon angles are preprocessed and saved. In addition, because the horizon information is used for irradiance calculations, the image often already exists.

The horizon angles depend on vertical exaggeration, so need to be recomputed if the exaggeration is changed. For cases where this is a problem, i.e., where one wishes to experiment with many different exaggerations, it is possible to store the coordinates of the grid points that form the horizons so that no recomputation is necessary.

RASTER RECONSTRUCTION

Image processing environments generally use images whose grey levels or DN values are defined at integer coordinates. The projective transformation of an input image whose values are a function of integers $f(l,s)$ produces an output image whose values are a function of non-integer raster device coordinates $g(x, y)$. Thus, $f(l, s)$ may map between the pixels in $g(x, y)$ or, if we choose integer coordinates in $[x\ y]$, the inverse mapping usually leads to non-integer image coordinates in $[l\ s]$. Some form of grey-level interpolation, or resampling, is needed to obtain output values at integer positions.

This resampling of the scattered grey levels can lead to another problem: aliasing. Aliasing is a consideration any time an image has spatial frequencies higher than twice the display sampling frequency. It is impossible to correctly reconstruct an image whose grey levels vary faster than 0.5 cycles/pixel. Frequencies higher than this are aliased, or folded, over into the lower frequencies and produce such visual effects as staircase edges and Moiré patterns.

GREY-LEVEL INTERPOLATION

The two major techniques used for grey-level interpolation in image processing are "pixel carryover" and "pixel filling" (Castleman, 1979). The

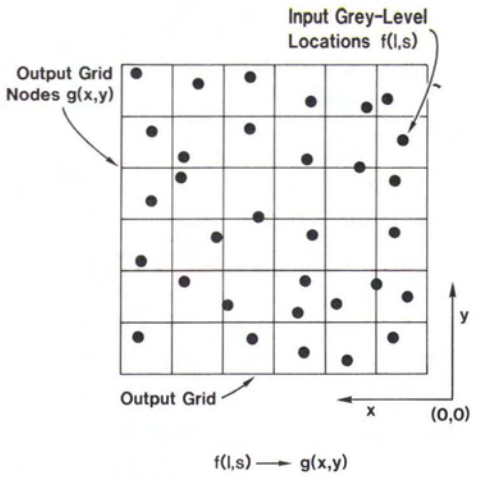

FIG. 2. The pixel carryover technique. The input grey levels must be interpolated to output integer grid locations.

method chosen depends on the desired direction of mapping between $f(l,s)$ and $g(x,y)$. If this direction is from $f(l,s) \rightarrow g(x,y)$, the input values are said to "carry over" to the output image plane. If the grey levels then map between output pixel locations, some type of resampling is performed as shown in Figure 2.

With pixel filling, the inverse transformation is performed; that is, the output image integer grid locations are mapped onto the input image, from $g(x,y) \rightarrow f(l,s)$. If the output grid location falls between input pixel locations, a value is interpolated.

The pixel filling method is preferred for several reasons. First, each output pixel is addressed only once. Secondly, it guarantees that each output pixel will have a reflectance value. This is not true for pixel carryover. If magnification or rotation is involved, some output pixels may have no input pixels mapping to them.

However, pixel filling requires that a unique inverse transformation exist from $g(x,y) \rightarrow f(l,s)$. For mappings from one two-dimensional space to another, such as for rotations, this transformation can usually be obtained. Image processings systems, for example VICAR (Castleman, 1979), use the pixel filling method for these types of transformations.

The projection problem is another matter. It involves going from the two-dimensional space fo the viewing plane to three-dimensional object space. Unfortunately, it is impossible to uniquely specify this inversion for non-parametrically defined elevation models with an orthographic projection. In the absence of such parameterization, it is necessary to use time consuming search procedures. A quick look at the forward equations shows why inversion is not possible. The forward equations, derived earlier, are

$$x' = u\cos\phi + v\sin\phi$$

$$y' = u\sin\phi\sin\theta + h\cos\theta - v\cos\phi\sin\theta$$

$$z' = u\sin\phi\sin\theta + h\sin\theta + v\cos\phi\cos\theta$$

In the inverse transformation, only x' and y', θ and ϕ are known; h and z' are not. If, however, h is defined as function of $[u\ v]$, then inversion using the first two equations may be possible.

For the above reasons, a pixel carryover technique is implemented here. The problem is to now interpolate the scattered grey values to integer output grid nodes. Two methods are considered: a nearest neighbor technique and a distance-weighted method. Both techniques are implemented with two goals in mind: visual quality and execution speed.

NEAREST-NEIGHBOR RESAMPLING

Assume we have the appropriately scaled non-integer coordinates of a transformed image. How can grey levels be assigned to integer grid locations? A fast method is to convert the coordinates to integers by either truncation or rounding.

This method has the advantage of being extremely fast, but has two minor problems: multiple pixel assignments and pixel dropouts. Multiple pixel assignments occur when more than one input value maps into the same output grid cell. Temporal priority is one possible solution. The grid cell gets the grey level of the last pixel to map into it. However, because only visible pixels have been transformed, some important reflectance information may be lost.

The solution implemented here averages all the values together. They are accumulated and then divided by the number of pixel contributors. This is somewhat wasteful of storage because a buffer must be created which holds the number of contributions received by each grid cell. The benefit of this averaging is that it blurs information along the "line of sight" much the way humans do, producing visually realistic results.

Pixel dropouts occur when no input pixel maps close to an output grid cell. The resulting image contains black pixel-sized gaps at these locations. The extent of pixel dropout is a function of the transformations involved and the amount of relief in the original image. Pixel dropout is not a serious problem and can be ignored, if desired. One way of combating it is through the use of more complicated resampling routines. A simpler answer, and one which loses no computational efficiency, is to shrink the output image by an "appropriate" amount. This technique, used here, is discussed in more detail in the **Pixel Dropout** section.

DISTANCE-WEIGHTED RESAMPLING

One of the problems with a simple nearest-neighbor method is its tendency to produce somewhat "blocky" images. Smoother appearing images can be created by using a higher order interpolation scheme. A complex interpolation

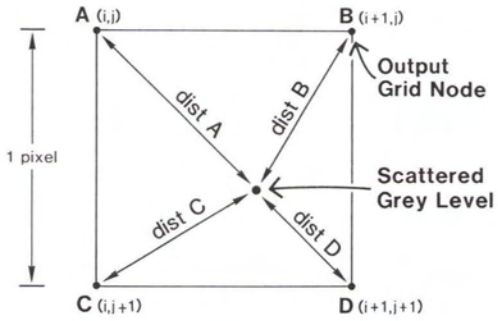

FIG. 3. Each input grey level contributes a fraction of its value to the surrounding grid nodes A, B, C, D based on its respective distances from them.

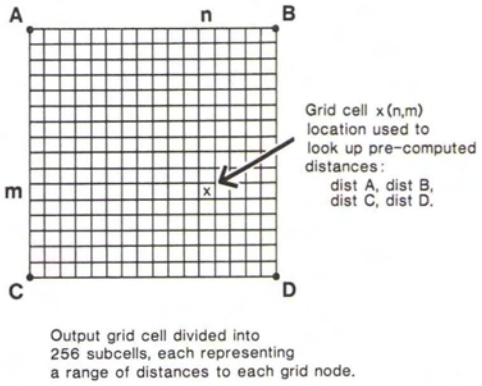

FIG. 4. Grid cell used to create lookup table for distance weighted resampling.

function is unnecessary for this application. One viable resampling option is the distance-weighted algorithm.

In general, each transformed input pixel maps between four output grid lcoations. An input pixel then contributes a fraction of its grey level to each of the four surrounding grid points, based on the its distance from each point (Figure 3). As each scattered point is examined, these four distances are calculated. The value is then divided by each of the four distances in turn, resulting in four fractional grey levels. These grey levels are then accumulated at each grid node. The inverse distances are also summed in a distance buffer. On output, the accumulated fractional grey levels are divided by the sum of the n inverse distances. For an output pixel at coordinate $[l\ s]$ we have

$$\text{output value} = \frac{\sum_{i=1}^{n} \frac{DN_i}{\sqrt{x_i - l)^2 + (y_i - s)^2}}}{\sum_{i=1}^{n} \frac{1}{\sqrt{(x_i - l)^2 + (y_i - s)^2}}}$$

Clearly this procedure is computationally slow.

Bilinear interpolation may appear to be more efficient (Castelman, 1979); however, it suffers from the same lack of ordering, in terms of input points, that limits the application of true nearest-neighbor schemes. Without some type of sorting and distance checking, the scattered points that surround a grid node are unknown. Bilinear interpolation is, thus, a slow process with pixel carryover techniques. Indeed, both it and distance-weighting are computationally expensive enough, when compared with the nearest-neighbor scheme, that the increased smoothness of the final product is not worth the added expense. Fortunately, there is a better solution.

Lookup tables can significantly increase the speed of distance-weighted interpolation. First, the cell defined by the four output grid locations surrounding a scattered point is divided into subcells of arbitrary fineness, say 16×16. Each subcell has four distances associated with it corresponding to the distances to each grid node. There are then only 16×16 possible distances between a subcell and a grid node (Figure 4). Four lookup tables of size 16×16 containing inverse distances are created and stored. The coordinates of a scattered point are converted to table indices and the distance to the grid nodes obtained immediately. The computational time savings are large and the procedure straightforward to implement.

Distance-weighted resampling tends to blur the image in the process of smoothing. Straight lines, edges, etc., tend to look less jagged because of this. This method also reduces pixel dropout because each grid point essentially has four grid cells from which it can acquire reflectance information. Even using this interpolation method, however, freedom from pixel dropouts cannot, in theory, be guaranteed. A brief look at the factors affecting pixel dropout will show why this is so and suggest a solution.

PIXEL DROPOUTS

Consider thet input elevation grid shown in Figure 5a; a towering mountain, one pixel wide, on a flat plane. An oblique view looking south from the horizon would produce the output shown in Figure 5b. Clearly, the interpolation methods presented here cannot deal with a situation like this. Neither have the capability to fill in grey values between the peak

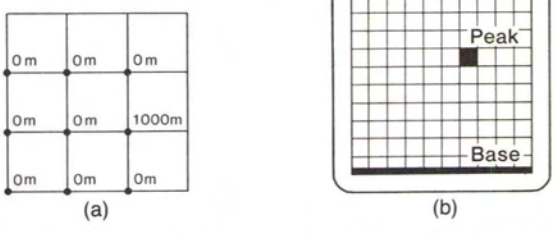

FIG. 5. Extreme vertical exaggeration leads to pixel dropouts in the image plane (a) Input elevation grid. (b) Output image.

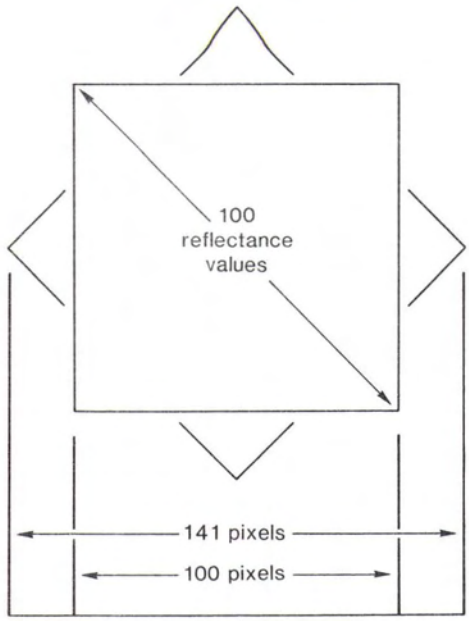

FIG. 6. Rotated and unrotated grids. The unrotated grid has only 100 reflectance values along the diagonal.

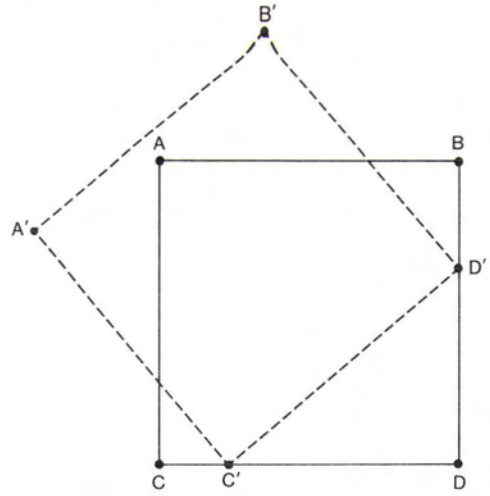

FIG. 7. Simple integerization does not provide a grey value for output node A, resulting in a pixel dropout due to resampling.

and base. Normal elevation data, even under reasonable vertical exaggeration, is never this rough. Thus, although it cannot be guaranteed, very few pixel dropouts should occur solely because of terrain roughness.

A second factor affecting pixel dropout is rotation in azimuth. Suppose a square image of dimensions 100×100 is rotated 45°, as shown in Figure 6. The horizontal and vertical dimensions of the rotated image are now each $100\sqrt{2}$; therefore, 141 grid locations must be filled in along the widest horizontal row, but the original image contains only 100 values along the diagonal corresponding to this row. The additional values must be obtained by interpolation.

The distance-weighting algorithm works satisfactorily in this case, and no pixel dropouts occur. The nearest-neighbor technique does lose pixels from this transformation. There are three ways to deal with this problem: ignore it, use more sophisticated and computationally more expensive interpolation schemes, or shrink the image until each grid location has a reflectance value. We usually use the last alternative because it involves no more calculation and virtually assures an image free from pixel dropouts under normal terrain viewing situations. Further, the exact size of the final image is often unimportant. Most users prefer a slightly smaller image to one that is blemished by pixel dropouts, no matter how minor. Most image processing systems have zoom features to magnify the image after such compression.

It is important to note that the amount of shrinkage should vary with the azimuthal rotation angle ϕ. Very little shrinkage should occur for small rotation angles, shrinkage is maximized at 45° rotation, and no shrinkage should occur for 90° rotations. For rotations, it is possible to derive a compression formula which guarantees each output pixel will have a reflectance value associated with it which meets these constraints.

Consider a situation where a grid has been rotated and superimposed on an unrotated output grid as shown in Figure 7. In this worst case, using integerization resampling, it is impossible for the top left grid node to ever obtain a reflectance value. From geometric considerations, shrinking the dimensions of the rotated grid cell by $\sqrt{2} \mid \cos (45° - \phi) \mid$, where ϕ is the rotational angle, will fix the problem. If ϕ is greater than 90°, it should be reduced by 90°.

The shrinkage factor is implemented in the device subroutine which scales the normalized transformed coordinates to scattered device space coordinates. Because the final output image size is based on the range of the device coordinates and the shrinkage factor, and not on *a priori* windowing information, this scaling takes place outside the general 4×4 transformation matrix. The row and column sizes of the output image are determined by dividing the ranges of the line and sample coordinates by the shrinkage factor.

The normalized coordinates are then multiplied by the row and column sizes, yielding scattered coordinates correctly scaled to the device space and to the appropriate output image size.

ALIASING

The effects of aliasing due to the above resampling processes are not objectionable for terrain viewing.

The main artifcat is the staircase appearance of edges which are not perpendicular to the view plane axes. For images from satellites, such linear features as highways and airport runways appear staircased owing to aliasing. For synthetic shaded relief images, the effect tends to be limited to the borders of the scene.

Aliasing occurs whenever an image is sampled at a rate less than twice the maximum spatial frequency in the image. This critical sampling rate is known as the Nyquist frequency (Jerri, 1979). If we sample at a rate lower than the Nyquist rate, some of the high frequency information will be aliased, or folded over, into the lower frequencies of the reconstructed image. With images defined by pixel units, the smallest display interval or period is one pixel. The maximum diplay sampling rate is thus one sample/pixel. Therefore, the highest frequency we can display without aliasing is simply half of this or 0.5 cycles/pixel.

The images viewed are discrete functions of line and sample spacing and do not contain any frequencies higher than 0.5 cycles/pixel. However, the axonometric projection transforms images such that they will usually contain higher frequencies. For example, a black pixel and a white pixel may be separated by a distance of one pixel in the original image but after a transformation may map to non-integer output positions that are close together, perhaps only a tenth of a pixel apart. This will then produce a frequency beyond the displayable limit.

The visual effects of aliasing, and alternatives for combating these effects, have been the subject of much discussion (Barros and Fuchs, 1979; Crow, 1977; Crow, 1981; Foley *et al.*, 1979). One method is to simply blur the output image. Another is to increase the resolution of the output device. Both of these suffer from obvious drawbacks and are not generally employed. A much more common anti-aliasing technique is filtering (Blinn, 1978; Moik, 1980; Rosenfeld and Kak, 1982). Filters are used to remove frequencies greater than half the Nyquist rate before sampling. The Fourier, Bartlett, and Wiener filters are some of the more common ones used for low-pass filtering. Visually, these filters tend to blur the image. This is expected because edges contain much high frequency information.

Perhaps the most effective method is area anti-aliasing (Leler, 1980). Here, pixels on the view plane are regarded not as points but as regions. The value of each pixel is the average, by area, of each object within its boundary. Unfortunately, this method is computationally difficult to implement, especially for non-polygonally defined surfaces.

In practice, it is doubtful whether anti-aliasing procedures are necessary in the present application, because the artifacts present in terrain views are not serious.

Prefiltering is expensive and area anti-aliasing even more so. Interpolations reduce the effects of aliasing because they act essentially as low-pass filters. For example, distance-weighted interpolation tends to smooth much the way a low pass prefilter would. Thus, if a smoother, less blocky, anti-aliased image is required, distance weighted interpolation should be adequate.

CONCLUSION

An algorithm has been created to produce orthographic views of terrain using digital elevation data, co-registered images, and horizon information computed from the elevation data. The algorithm is unique in its use of horizon angles to determine surface visibility and its implementation of two efficient grey-level interpolation routines which offer the user a choice between speed and smoothness. One of the problems associated with interpolation using pixel carryover techniques—pixel dropouts—has been reduced by shrinking the output image by an appropriate amount.

In use, the program has proved satisfactory for most situations. Depending on site-specific hardware, its somewhat greedy memory allocations could limit the size of the input images, but this is not usually a problem if memory is de-allocated during execution. Extreme vertical exaggeration may cause pixel dropouts, but these can be made much less noticeable by using standard smoothing operations.

ACKNOWLEDGMENTS

Our work was supported by NASA grants NAS5-27463 and NAS5-28770.

REFERENCES

Batson, R. M., K. Edwards, E. M. Eliason, 1975. Computer-Generated Shaded-Relief Images: *Journal of Research, U.S. Geological Survey*, vol. 3, no. 4, pp. 401-408.

Blinn, J., 1978. *Computer Display of Curved Surfaces:* Ph.D. Thesis, Department of Computer Science, University of Utah, Salt Lake City.

Bunker, W., and R. Heartz, 1976. *Perspective Display Simulation of Terrain*: AFHRL-TR-76-39, Air Force Human Resources Laboratory.

Castleman, K. R., 1979. *Digital Image Processing*: Prentice-Hall, Englewood Cliffs, New Jersey.

Catmull, E., and A. R. Smith, 1980. 3-D Transformation of Images in Scanline Order: *Computer Graphics*, vol. 14, no. 3, pp. 279–285.

Dozier, J., J. Bruno, and P. Downey, 1981. A Faster Solution to the Horizon Problem: *Computers and Geosciences*, vol. 7, no. 2, pp. 145–151.

Dungan, W., Jr., 1979. A Terrain and Cloud Computer Image Generation Model: *Computer Graphics*, vol. 13, no. 2, pp. 143–150.

Faintich, M.B., 1974. *Digital Scene and Image Generation*: Technical Report TR-3147, Naval Weapons Laboratory.

Foley, J.D., and A. Van Dam, 1982. *Interactive Computer Graphics*: Addison-Wesley, Reading, Mass.

Glassner, A. S., 1984. Space Subdivision for Fast Ray Tracing: *IEEE Computer Graphics and Applications*, vol. 4, no. 1, pp. 15-22.

Horn, B. K. P., 1981. Hill Shading and the Reflectance Map: *Proceedings of the IEEE*, vol. 69, no. 1, pp. 14–47.

Horn, B. K. P., and B. L. Bachman, 1978. Using Synthetic Images to Register Real Images with Surface Models: *Communications of the ACM*, vol. 21, no. 11, pp. 914–924.

Jerri, A. J., 1979. The Shannon Sampling Theorem—Its Various Extensions and Applications: *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1565–1596.

Leler, W. J., 1980. Human Vision, Anti-Aliasing, and the Cheap 4000 Line Display: *Computer Graphics*, vol. 14, no. 3, pp. 308–313.

Moik, J. G., 1980. *Digial Processing of Remotely Sensed Images*: NASA SP-431, National Aeronautics and Space Administration, Washington, D.C.

Rogers, D. F., and J. A. Adams, 1976. *Mathematical Elements for Computer Graphics*: McGraw-Hill Book Company, New York.

Rosenfeld, A., and A. C. Kak, 1982. *Digital Picture Processing*: 1, Academic Press, New York.

Rubin, S. M., and T. Whited, 1980. A 3-Dimensional Representation for Fast Rendering of Complex Scenes: *Computer Graphics*, vol. 14, no. 3, pp. 110–116.

Scholz, D. K., S. W. Doescher, and J. W. Feuquay, 1984. The Use of DEM Datat for Generating Shaded Relief, Stereo and Perspective Views of Satellite Acquired Data: Machine Processing of Remotely Sensed Data Symposium, Purdue University, West Lafayette, In. (Paper presented but not published in Proceedings)

Strat, T. M., 1978. *Shaded Perspective Images of Terrain*: AI Memo 463, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA. NTIS AD-A055070/7GI 78-18 8B PCA03/MFA01

Woodham, R. J., 1976. *Two Simple Algorithms for Displaying Orthographic Projections of Surfaces*: Working Paper 126, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Mass.

(Received 23 March 1985; accepted 5 August 1985; revised 22 October 1985)
