Rapid Calculation of Terrain Parameters For Radiation Modeling From Digital Elevation Data

JEFF DOZIER, ASSOCIATE, IEEE, AND JAMES FREW, MEMBER, IEEE

Abstract—**Digital elevation models are now widely used to calculate terrain parameters to determine incoming solar and longwave radiation for use in surface climate models, interpretation of remote-sensing data, and parameters in hydrologic models. Because of the large number of points in an elevation grid, fast algorithms are useful to save computation time. We describe rapid methods for calculating slope and azimuth, solar illumination angle, horizons, and view factors for radiation from sky and terrain. Calculation time is reduced by fast algorithms and lookup tables.**

I. Use of Digital Elevation Models in Radiation Calculations

IN ALL but very gentle terrain, significant variation in the surface climate and in remote-sensing images results from local topographic effects. The major contributors to this variation are solar and longwave (thermal) irradiance, although there are also important topographic variations in wind speed and soil moisture. The topographic effects on solar irradiance are mainly variation in illumination angle and shadowing from local horizons. In the thermal part of the electromagnetic spectrum, the emission from surrounding slopes usually causes valley bottoms to receive more thermal irradiance than unobstructed areas. Problems in calculating radiation over mountainous areas have been addressed by many papers in the last two decades [1], [2].

Most radiation calculations over terrain are made with the aid of digital elevation grids, whereby elevation data are represented by a matrix. In the United States, these are available as Digital Elevation Models (DEM's) from the U.S. Geological Survey [3]. The 1:250 000 quadrangles for the entire U.S. are available at a grid resolution of 63.5 m (0.01 in. at map scale), and the 1:24 000 quadrangles are available at 30-m resolution. Several commercial firms sell software which will derive digital elevation models from digital stereophotography from aircraft or satellite.

The rationale for addressing the problem of rapidly calculating the necessary terrain parameters stems from the large size of the DEM's used in some current applications. The earlier investigations cited above used either coarse grid spacings or small areas, with the resulting grids containing only 10^2 to 10^3 points. Even in these applications [1], [4], computer time for generating terrain parameters accounted for a significant part of the total computer resources used. In our current application—investigation of the snow surface radiation balance in the southern Sierra Nevada—we often use terrain grids of 10^5 to 10^6 points.

It is possible to reduce the computation time required to calculate some terrain parameters through the use of better algorithms, lookup tables to replace calculations, and integer instead of floating-point arithmetic. In this paper, we describe rapid methods for calculating: (i) Slope and azimuth; (ii) illumination angle; (iii) horizons; and (iv) view factors for radiation from sky and terrain. This list includes all necessary variables used in radiation models. The drainage basin algorithm used to prepare some of the figures is described in detail elsewhere [5]. We also hope that the techniques described can be extended or generalized to other calculations with DEM's [6]–[8].

II. Data Representation

We avoid the use of floating-point numbers in storing large data sets, such as satellite images, digital elevation models, and any parameters derived from them, even though the values represented are real rather than integer quantities. The reasons for this are to reduce the size of the data set and make it portable between machines with different internal floating-point representation.

The common strategy adopted here is data representation by "piecewise linear quantization." $Q_L(x, N)$ is the N-bit linear quantization of a real number x, whereby an unsigned N-bit integer in the range $[0, 2^N - 1]$ is mapped to the range $[x_{\min}, x_{\max}]$. Within the range $[0, 2^N - 1]$ there are break-points, $b = 0, \cdots, B - 1$, where $B \geq 2$, with a value x_b corresponding to each b. Between break-

Manuscript received October 8, 1989. This work was supported by NASA.

J. Dozier is with the Center for Remote Sensing and Environmental Optics, University of California, Santa Barbara, CA 93106. He is also affiliated with the Universities Space Research Association, NASA/Goddard Space Flight Center, Greenbelt, MD 20771.

J. Frew is with the Center for Remote Sensing and Environmental Optics, University of California, Santa Barbara, CA 93106.

IEEE Log Number 9035826.

points, the mapping between Q_L and x is linear:

$$Q_L(x, N) = \left[Q_b + (Q_{b+1} - Q_b) \frac{x - x_b}{x_{b+1} - x_b} + 0.5 \right], \quad \text{for } x_b \le x < x_{b+1}. \tag{1}$$

Its inverse is

$$x = (x_{b+1} - x_b) \frac{Q_L(x, N)}{Q_{b+1} - Q_b} + x_b, \quad \text{for } Q_b \le Q_L(x, N) < Q_{b+1}. \tag{2}$$

The "floor" notation in (1) means that we take the largest integer not greater than the quantity within. We need store only the quantized integer values, the break-points, and the floating-point values corresponding to each break-point. Normally, $x_{b+1} > x_b$ for all b, but the definitions given above do not require this and we sometimes reverse the quantization. The relationship between b and x_b must be monotonic, however.

III. Boundary Conditions for Radiation Models

The radiative transfer equation [9] for plane-parallel atmospheres is usually expressed as

$$\cos\theta \frac{dL(\tau, \theta, \phi)}{d\tau} = -L(\tau, \theta, \phi) + J(\tau, \theta, \phi) \tag{3}$$

where L is the radiance at optical depth τ in direction θ, ϕ; ϕ is the azimuth; and θ is the angle from zenith. J is the source function; it results from the scattering of both direct and diffuse radiation or, at thermal wavelengths, emission.

The solution of (3) requires that we specify upper and lower boundary conditions. At the top of the atmosphere the boundary condition is simple, $L\downarrow(0) = 0$. At the bottom of the atmosphere over rugged terrain, however, the boundary condition is more complicated. In the solar spectrum, irradiance has three sources: (i) Direct irradiance from the sun; (ii) diffuse irradiance from the sky, where a portion of the overlying hemisphere is obscured by terrain; and (iii) direct and diffuse irradiance, on nearby terrain, that is reflected toward the point whose boundary condition we want to specify. In the thermal portion of the spectrum the solar contribution is absent, the diffuse irradiance results from atmospheric emissions, and the contribution from the surrounding terrrain is from emission instead of reflection. For remote-sensing interpretation, the upwelling radiation normal to a slope S must be normalized to a horizontal plane by multiplying by cos S.

The direct irradiance on a slope is

$$E_S = \cos\theta_S E_0 e^{-\tau_0/\cos\theta_0} \tag{4}$$

where E_0 is the exoatmospheric solar irradiance perpendicular to the sun's rays; τ_0 is the optical depth of the atmosphere; and θ_0 is the solar illumination angle on a horizontal surface. Further, θ_S is the solar illumination angle on the slope.

The effect of earth and atmosphere curvature on the path length is less than 1% for $\theta_0 \le 72°$; for larger solar zenith angles, Kasten's [10] empirical equations for the optical path length can be used.

Diffuse irradiance from the sky, either scattered or emitted, is

$$E_d = \pi V_d \overline{L_d} \tag{5}$$

As $\overline{L_d}$ is the mean downward radiance on an unobstructed horizontal surface, $\pi\overline{L_d}$ is the diffuse irradiance. The sky-view factor V_d is the ratio of the diffuse sky irradiance at a point to that on an unobstructed horizontal surface; i.e., $0 < V_d \le 1$. It accounts for the slope and orientation of the point, the portion of the overlying hemisphere visible to the point, and the anisotropy of the diffuse irradiance. We define $\eta_d(\theta, \phi)$ as an anisotropy factor such that $\eta_d(\theta, \phi)\overline{L_d} = L(\theta, \phi)$. Therefore, η_d is normalized such that its hemispheric integral projected onto a horizontal surface is π; i.e.,

$$\int_0^{2\pi} \int_0^{\pi/2} \eta_d(\theta, \phi) \sin\theta \cos\theta \, d\theta \, d\phi = \pi. \tag{6}$$

V_d on slope S with azimuth A is found by projecting each element of η_d onto the slope and integrating over the unobstructed hemisphere (i.e., from the zenith downward to the local horizon) through angle H_ϕ, for each direction ϕ. For an unobstructed horizontal surface $H_\phi = \pi/2$ (Fig. 1). The horizon can result either from "self-shadowing" by the slope itself or from adjacent ridges. In the isotropic case, where $\eta_d = 1$, the inner integral in (7a) below, for a given azimuth, can be evaluated analytically, leading to the approximation in (7b):

$$V_d = \frac{1}{\pi} \int_0^{2\pi} \int_0^{H_\phi} \eta_d(\theta, \phi) \sin\theta \left[\cos\theta \cos S + \sin\theta \sin S \cos(\phi - A)\right] d\theta \, d\phi \tag{7a}$$

$$\approx \frac{1}{2\pi} \int_0^{2\pi} \left[\cos S \sin^2 H_\phi + \sin S \cos(\phi - A) \times (H_\phi - \sin H_\phi \cos H_\phi)\right] d\phi. \tag{7b}$$

Contribution from the surrounding terrain is

$$E_t = \pi C_t \overline{L_t}. \tag{8}$$

The average irradiance reflected or emitted from the surrounding terrain is $\pi\overline{L_t}$. The terrain configuration factor C_t includes both the anisotropy of the radiation and the geometric effects between that point and each point in the surrounding terrain with which it is mutually visible. The contribution of each of these terrain elements to the configuration factor could be computed [11], but this is a formidable computational problem. Instead, we use the following approximation:

$$C_t = \frac{1}{\pi} \int_0^{2\pi} \int_{H_\phi}^{\psi_\phi} \eta_v(\theta, \phi) \sin\theta \left[\cos\theta \cos S + \sin\theta \sin S \cos(\phi - A)\right] d\theta \, d\phi \tag{9a}$$

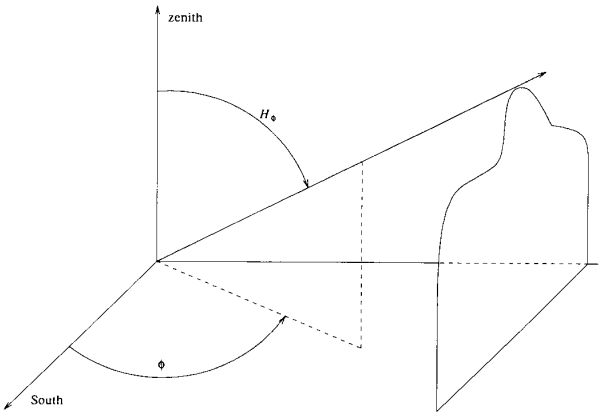


Fig. 1. Horizon angle H_ϕ for direction ϕ.

$$\approx \frac{1 + \cos S}{2} - V_d. \qquad (9b)$$

The value of η_v accounts for the anisotropy of the reflected or emitted radiance from the surrounding terrain, including geometric effects. The limits of integration for the inner integral are from the horizon downward to where a ray is parallel to the slope:

$$\psi_\phi = \arctan\left[\frac{-1}{\tan S \cos(\phi - A)}\right]. \qquad (10)$$

In the upslope direction $\cos(\phi - A)$ is negative, so $\psi_\phi < \pi/2$. In the downslope direction $\cos(\phi - A)$ is positive, so $\psi_\phi > \pi/2$. Across the slope, $\psi_\phi = \pi/2$.

Rigorous calculation of C_t is difficult because it is necessary to consider every terrain facet visible from a point to calculate η_v. In contrast to the sky radiation, the isotropic assumption is unrealistic because considerable anisotropy results from geometric effects, even if the surrounding terrain is a Lambertian reflector or a blackbody emitter. We therefore note that V_d for an infinitely long slope is $(1 + \cos S)/2$, which leads to the approximation in (9b).

IV. Algorithms for Rapid Calculation

The terrain grid is oriented as shown in Fig. 2. Spacing between grid points is Δh in both the x and y directions, although the algorithms presented here could be modified for rectangular spacing, if desired. Elevations z are stored as N-bit linear quantizations $Q_L(z, N)$ in the range $[z_{\min}, z_{\max}]$. The grid is oriented with the rows from west to east and the columns from north to south, so that x increases southward and y increases eastward. Fig. 3 is an image of such a grid for the southern Sierra Nevada.

Implementations of the algorithms are presented in the C programming language [12]. Some of the more esoteric features of C, such as the use of register variables and some arcane but speedy pointer operations, have been omitted for simplicity. The function POW2(n) returns 2^n for positive integer values of n and is usually implemented as a macro:

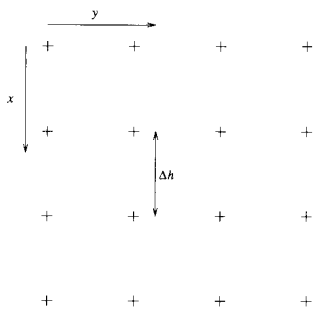


Fig. 2. Orientation of digital elevation model, showing direction of x and y coordinates and spacing Δh.

Fig. 3. Digital elevation image around the Emerald Lake basin in the southern Sierra Nevada, made by stereophotogrammetry from low altitude aerial photographs. Grid spacing is 5 m. The image shown covers 1.45 × 2.40 km.

```
#define POW2(n) (1 << (n)).
```

The constant PI is assumed to be the value of π, to the precision of the host machine.

A. Slope and Azimuth

The sine of the slope angle S with the range [0, 1] is represented by an M-bit linearly quantized value $Q_L(\sin S, M)$ with two break-points. We use the sine instead of the cosine because the greatest precision is then for the lowest slopes. The azimuth (or aspect) A is represented by $Q_L(A, M)$ in the range $[-\pi, \pi(1 - 2^{1-M})]$, also with two break-points. $A = 0$ is toward the south, and positive azimuths are toward the east; note that $-\pi$ and π are the same azimuth.

The fundamental equations are given below. The signs of the numerator and denominator allow A to be uniquely specified over $[-\pi, \pi]$:

$$\tan S \equiv |\nabla_z| = \left[(\partial z/\partial x)^2 + (\partial z/\partial y)^2\right]^{1/2} \qquad (11)$$

$$\tan A = \frac{-\partial z/\partial y}{-\partial z/\partial x} \qquad (12)$$

where $\partial z/\partial x$ and $\partial z/\partial y$ are calculated by finite differences. At point i, j:

$$\frac{\partial z}{\partial x} = \frac{z_{i+1,j} - z_{i-1,j}}{2\Delta h} \tag{13a}$$

$$\frac{\partial z}{\partial y} = \frac{z_{i,j+1} - z_{i,j-1}}{2\Delta h}. \tag{13b}$$

The elevations are stored in the terrain grid as N-bit linearly quantized integers, where N is usually greater than M.

B. Illumination Angle

The most important variable controlling the incident radiation on a slope in mountainous terrain is the local solar illumination angle. If the sun is not hidden by a local horizon (a problem addressed in the next section), the local illumination angle θ_S on a slope S with azimuth A is given by:

$$\cos\theta_S = \cos\theta_0 \cos S + \sin\theta_0 \sin S \cos(\phi_0 - A) \tag{14}$$

where θ_0 is the illumination angle on a horizontal surface and ϕ_0 is the azimuth of illumination.

As in the previous section, the quantities sin S and A are stored as M-bit linear quantizations. We achieve speed in the calculation of $Q_L(\cos\theta_S, M)$ by the following steps.

1) We ignore variations in latitude and longitude over the terrain grid, or at least over portions of it. Therefore, θ_0 and ϕ_0 are constants.

2) Because S and A are stored as linear quantizations, there are only 2^M possible values for cos S, sin S, and cos $(\phi_0 - A)$. We, therefore, build lookup tables to avoid computing trigonometric functions at each point Table I.

3) Moreover, although there are 2^{2M} possible S, A pairs, not all combinations occur in a typical terrain grid. Therefore the algorithm can keep track of which S, A pairs have already been encountered (Table II). Fig. 4 shows a shaded-relief image of the elevation data in Fig. 2, using the slopes and azimuths in Fig. 3, a solar zenith angle of 60°, and a solar azimuth of 42° east of south. These correspond to the solar position at the time of the Landsat overpass in mid-February.

C. Horizons

The cosine of the angle from the zenith to the horizon in a given azimuth ϕ is represented by an M-bit linear quantization in the range [0, 1]. The algorithm described in this section was previously published [13], but we have implemented many changes to make it clearer and faster.

By rotating a grid in direction ϕ, we reduce the horizon problem to its one-dimensional equivalent. Along each row of the rotated grid (Fig. 5) we want to calculate the horizon angle in the forward (or backward) direction for each point; then we rotate the solution back to the original orientation. We [13] showed that this can be solved in "order N" iterations, where N is the number of points in the profile and the computation time is linearly related to N. All other methods for solving this problem [14], [1] are apparently order N^2, so computing times for larger

TABLE I
FUNCTION 1: TRIGONOMETRIC FUNCTION LOOKUP TABLES

Function 1. Trigonometric Function Lookup Tables

```
extern double cos(), sqrt();

void
trigtbl(nSlopeBits, nAzmBits, solarAzm, sinSlope, cosSlope,
        cosAzmDiff)

    int       nSlopeBits, nAzmBits;
    float     solarAzm, *sinSlope, *cosSlope, *cosAzmDiff;
{
    int       slope, azm;
    float     slopeMax, azmCoef;

 /*
  * Compute sines and cosines for each possible slope.
  * Start with maximum slope and work backwards.
  */
    slope = POW2(nSlopeBits) - 1;
    slopeMax = (float) slope;

    do {
        float     temp = (float) slope / slopeMax;

        sinSlope[slope] = temp;
        cosSlope[slope] = sqrt((1 - temp) * (1 + temp));
    } while (--slope >= 0);
 /*
  * Compute cosine(solar azimuth - azimuth) for each possible
  * azimuth.  Start with maximum azimuth and work backwards.
  */
    azm = POW2(nAzmBits) - 1;
    azmCoef = 2.0 * PI / (float) azm;
    solarAzm += PI;

    do {
        cosAzmDiff[azm] =
            cos(solarAzm - azmCoef * (float) azm);
    } while (--azm >= 0);
}
```

TABLE II
FUNCTION 2: VECTOR OF COSINES OF LOCAL ILLUMINATION ANGLE

Function 2. Vector of Cosines of Local Illumination Angle

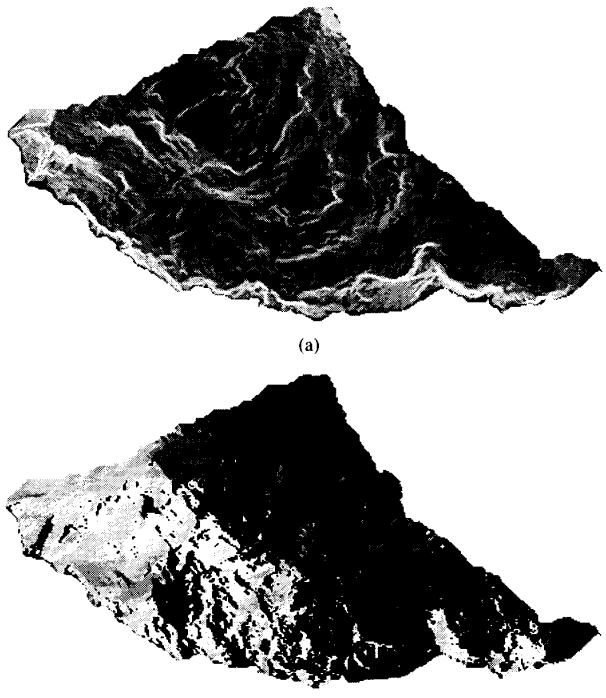
```
unsigned
cosine(slope, azm, cosSolarZ, sinSolarZ, sinSlope,
       cosSlope, cosAzmDiff, already, savedCosZ, nCosZBits)

    unsigned  slope, azm, **savedCosZ;
    float     cosSolarZ, sinSolarZ, *sinSlope, *cosSlope,
              *cosAzmDiff;
    int       **already, nCosZBits;
{
    float     cosZ;
 /*
  * If we've already encountered this (slope,azimuth) combination
  * then use precomputed cosine.
  */
    if (!already[slope][azm]) {
        cosZ = cosSolarZ * cosSlope[slope] +
            sinSolarZ * sinSlope[slope] * cosAzmDiff[azm];

        if (cosZ <= 0)
            savedCosZ[slope][azm] = 0;
        else
            savedCosZ[slope][azm] =
                cosZ * (POW2(nCosZBits) - 1) + 0.5;

        already[slope][azm] = 1;
    }

    return (savedCosZ[slope][azm]);
}
```



(a)

(b)

Fig. 4. Slope (a) and azimuth (b) images of the elevation grid in Fig. 3, masked for the Emerald Lake drainage basin.

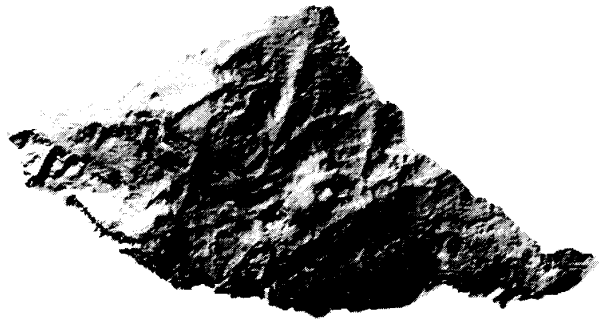


Fig. 5. Shaded relief image of the Emerald Lake drainage basin, at a solar zenith angle of 60° and a solar azimuth of 42° east of south, corresponding to the solar position at the time of a mid-February Landsat overpass.

elevation grids become enormous. The fast solution is achieved by casting the problem in a somewhat ill-posed way. Instead of directly finding the horizon angle, we find the point that forms the horizon.

This method has an additional advantage: While the angles to the horizons will change with the vertical exaggeration of the terrain grid, the coordinates of the points forming the horizons do not. When computing orthographic views of satellite images registered to elevation grids, the same horizon algorithm can be used to decide *a priori* which points in the image are visible [15]. By storing the points that form the horizons, one can easily generate different vertical exaggerations of the same grid.

TABLE III
FUNCTION 3: HORIZON IN FORWARD DIRECTION FOR EQUI-SPACED ELEVATION VECTOR

Function 3. Horizon in Forward Direction for Equi-Spaced Elevation Vector

```
#define SLOPE(i, j, z)  (                                   \
            ((z[j]) <= (z[i])) ? 0 :                        \
            ((z[j]) - (z[i])) / (float) ((j) - (i)) \
        )

horlf(elev, horzPt, nHorz)
    int      *horzPt, nHorz;
    float    *elev;
{
    int       i, j, currHorz;
    float     slopeItoHorzJ, slopeItoJ;

 /* End point is its own horizon. */

    horzPt[nHorz - 1] = nHorz - 1;

 /* Loop from next-to-end backward to beginning. */

    for (i = nHorz - 2; i >= 0; --i) {
 /*
  * Start with adjacent point in forward direction; loop
  * until slope from i to j is greater than or equal to
  * the slope from j to its horizon.
  */
        currHorz = i + 1;

        do {
            j = currHorz;
            currHorz = horzPt[j];
            slopeItoJ = SLOPE(i, j, elev);
            slopeItoHorzJ = SLOPE(i, currHorz, elev);
        } while (slopeItoJ < slopeItoHorzJ);

        if (slopeItoJ > slopeItoHorzJ)
            horzPt[i] = j;
        else if (slopeItoJ == 0)
            horzPt[i] = i;
        else
            horzPt[i] = currHorz;
    }
}
```

For a grid row, define an elevation function z on the points $j = 0, 1, \cdots, N-1$. Since the points are evenly spaced, the abscissa is specified by $j\Delta h$. The horizon function h in the forward direction satisfies, for all $0 \le i < N$:

1) $i \le h_i$, hence $h_{N-1} = N - 1$;

2) if $z_i \ge z_j$ for all $i < j < N$, then $h_i = i$; i.e., if the elevation is greater than or equal to any other point in the forward direction, we say it is its own horizon; and

3) if k is the largest value greater than i and less than N such that $z_k > z_i$ and

$$z_j \le z_i + (z_k - z_i)\left|\frac{j-i}{k-i}\right|$$

for all $i < j < N$, and $j \ne k$, then $h_i = k$; that is, if two points in the forward direction are equally suitable as horizon candidates, the farthest is chosen.

A simple algorithm for determining the horizon functions for each point would be to compute the slope from each i to each $j > i$ and choose the maxima. Unfortunately, the number of comparisons in such an algorithm is on the order of the number of points in the elevation profile squared.

We achieve a much faster speed by comparing the slope from i to j with the slope from j to h_j. If the slope from i

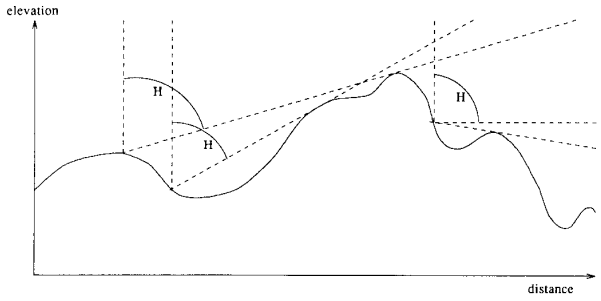


Fig. 6. A single horizon profile showing the horizon angle H in the forward direction for three points.

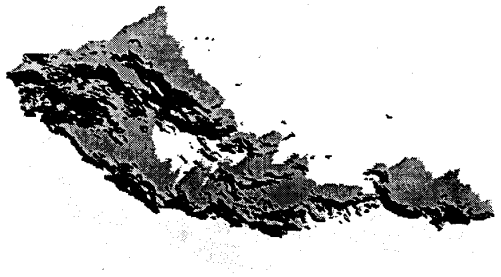


Fig. 7. Shadowed portions of the Emerald Lake drainage basin, for the same solar orientation as Fig. 5. The black areas are "self-shadowed"; i.e., the orientation of the slope is such that it is hidden from the sun. The grey areas are shadowed by local horizons.

to j is greater than from j to j's horizon, then all points forward of j are not visible from i and, therefore, j is i's horizon. Alternatively, if the slope from j to its horizon is greater than the slope from i to j, then all points between j and j's horizon need not be checked, because point j's horizon is visible from i. The number of comparisons in the fast algorithm is only linearly related to the number of points in the elevation profile.

Table III shows the one-dimensional horizon algorithm. Once the coordinates of the horizon points are found, the angles to the horizon are easily calculated.

The one-dimensional algorithm assumes that the elevation profile is aligned with one of the axes of the terrain grid. An arbitrarily aligned profile may be obtained by rotating the terrain grid to applying the one-dimensional algorithm. Actually, a full rotation is not necessary: It is sufficient to shear the grid by the appropriate angle, then adjust the value of Δh.

Fig. 6 shows those points that are shadowed for the same solar position as in Fig. 4.

D. Sky-View Factor

The sky-view factor V_d (7b) accounts for the portion of the overlying hemisphere visible to a grid point. To calculate it, one needs to know the slope S and azimuth A of the point, plus the horizon angle H_ϕ in a discrete set of directions ϕ. Usually 16 directions around the circle are enough. Because there are only 2^M possible values for S, A, or H_ϕ, the values of the trigonometric functions in (7b) can be stored in tables. Table IV shows the algorithm. Fig. 7 shows view factors for the same terrain grid as the previous figures.

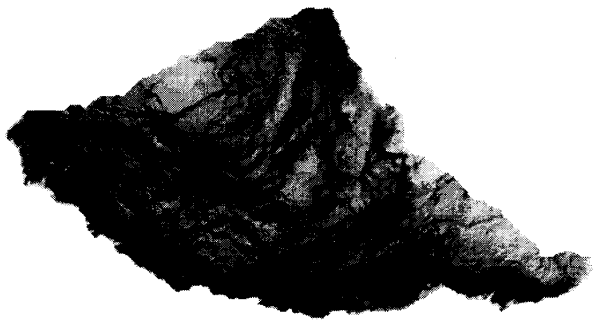


Fig. 8. View factors for sky radiation. The bright areas, on ridges, see a large portion of the 2π sky hemisphere; the dark areas have much of the overlying hemisphere obscured by adjacent terrain.

TABLE IV
FUNCTION 4: SKY VIEW FACTOR V_d

Function 4. Sky View Factor V_d

```
unsigned
viewd(slope, azm, horz, nHorz, sinSlope, cosSlope,
      cosAzmDiff, horzAngle, sinHorz, cosHorz, nViewBits)

    unsigned  slope, azm, *horz;
    int       nHorz, nViewBits;
    float     *sinSlope, *cosSlope, *cosAzmDiff, *horzAngle,
              *sinHorz, *cosHorz;
{
    int       direction;
    float     kCosSlope, kSinSlopeCosAzmDiff, view;

 /* These values are constant w.r.t. horizon direction. */

    kCosSlope = cosSlope[slope];
    kSinSlopeCosAzmDiff = sinSlope[slope] * cosAzmDiff[azm];

 /* Sum over all horizon directions. */

    view = 0;

    for (direction = nHorz; --direction >= 0;) {
        unsigned  kHorz;
        float     sinKHorz;

        kHorz = horz[direction];
        sinKHorz = sinHorz[kHorz];

        view += kCosSlope * sinKHorz * sinKHorz +
            kSinSlopeCosAzmDiff * (horzAngle[kHorz] -
                                   sinKHorz * cosHorz[kHorz]);
    }

 /* Convert result to linear quantization. */

    return (view * (POW2(nViewBits) - 1) / nHorz + 0.5);
}
```

V. CONCLUSION

The fast algorithms presented here save considerable time in calculating terrain parameters for solar and longwave radiation, for purposes of surface climate modeling or interpretation of remote-sensing images.

REFERENCES

[1] J. Dozier and S. I. Outcalt, "An approach toward energy balance simulation over rugged terrain," *Geograph. Anal.*, vol. 11, pp. 65–85, 1979.

[2] G. A. Olyphant, "Longwave radiation in mountainous areas and its influence on the energy balance of alpine snowfields," *Water Resources Res.*, vol. 22, no. 1, pp. 62–66, 1986.

[3] A. A. Elassal and V. M. Caruso, *Digital Elevation Models*. Alexandria, VA: U.S. Geological Survey, 1983, Circular 895-B.

[4] E. C. Anderson, "A numerical model for the estimation of solar radiation on rugged terrain," Ph.D. thesis, Ohio State Univ., Columbus, OH, 1985.

[5] D. Marks, J. Dozier, and J. Frew, "Automated basin delineation from digital elevation data," *Geo-Processing*, vol. 2, pp. 299–311, 1984.

[6] Y. B. Zecharias and W. Brutsaert, "Ground surface slope as a basin scale parameter," *Water Resources Res.*, vol. 21, no. 12, pp. 1895–1902, 1985.

[7] L. E. Band, "Topographic partition of watersheds with digital elevation models," *Water Resources Res.*, vol. 22, no. 1, pp. 15–24, 1986.

[8] E. M. O'Loughlin, "Prediction of surface saturation zones in natural catchments by topographic analysis," *Water Resources Res.*, vol. 22, no. 5, pp. 794–804, 1986.

[9] S. Chandrasekhar, *Radiative Transfer*. New York: Dover, 1960.

[10] F. Kasten, "A new table and approximation formula for the relative optical air mass," *Archiv für Meteorologie, Geophysik und Bioklimatologie*, vol. B-14, pp. 206–233, 1966.

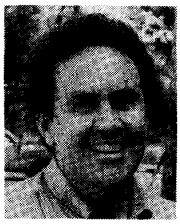
[11] R. Siegel and J. R. Howell, *Thermal Radiation Heat Transfer*, 2nd ed. New York: McGraw-Hill, 1981.

[12] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[13] J. Dozier, J. Bruno, and P. Downey, "A faster solution to the horizon problem," *Computers and Geosciences*, vol. 7, no. 2, pp. 145–151, 1981.

[14] L. D. Williams, R. G. Barry, and J. T. Andrews, "Application of computed global radiation for areas of high relief," *J. Appl. Meteorol.*, vol. 11, pp. 526–533, 1972.

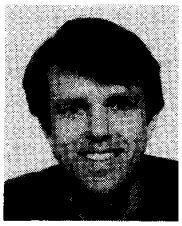
[15] R. Dubayah and J. Dozier, "Orthographic terrain views using data derived from digital elevation models," *Photogrammetr. Eng. Remote Sensing*, vol. 52, no. 4, pp. 509–518, 1986.



Jeff Dozier (A'84) received the B.A. degree in geography from California State University, Hayward, in 1968, the M.Sc. degree in 1969 and Ph.D. degree in 1973 from the University of Michigan, Ann Arbor.

He has taught at the University of California, Santa Barbara, since 1974, and from 1987–1990 was a Senior Member of the Technical Staff of the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, where he was the Project Scientist for HIRIS—the High-Resolution Imaging Spectrometer. He is now the Project Scientist for NASA's Earth Observing System at the Goddard Space Flight Center, Greenbelt, MD. He is or has been the Principal Investigator on several NASA grants in remote sensing, including two projects to evaluate and analyze Landsat Thematic Mapper data, the First ISLSCP Field Experiment, the upcoming Shuttle Imaging Radar-C, and the Earth Observing System. He was also a Principal Investigator on the California Air Resources Board's Integrated Watershed Project. He has published over 80 scientific papers and 1 book, and he is an Associate Editor of *Water Resources Research*. His research interests are in snow hydrology, Earth system science, radiative transfer in snow, remote sensing and data systems, image processing, and terrain analysis.

*



James Frew (S'84–M'86) received the B.A., M.A., and Ph.D. degrees in geography at the University of California, Santa Barbara.

He manages the computing activities at the Center for Remote Sensing and Environmental Optics at the University of California, Santa Barbara, where he has taught courses in the Geography and Computer Science Departments. His research interests are in image-processing algorithms and in computing environments for geographic data analysis. He is the Principal Architect of the *Image Processing Workbench*, a UNIX-based portable image-processing software system.