A Vector-Based Slope and Aspect Generation Algorithm

Paul Ritter
Remote Sensing Research Program, Space Sciences Laboratory, University of California, Berkeley, CA 94720

ABSTRACT: Slope and aspect information, alone or in combination with other data, can be of considerable value in a geographic information system (GIS). Some examples include mapping for forest fuels management, determining suitability for development, estimating potential availability of solar energy, and determining erosion potential. This paper describes an algorithm for use in a raster-based GIS that generates slope and aspect values from digital elevation data.

INTRODUCTION

SLOPE AND ASPECT DATA can be combined with other information in a geographic information system (GIS) to assist in providing answers to a wide variety of questions. A forest manager may need to know what fuels management techniques can be used in a particular area; a planner may wish to check potential sites for solar housing development, or to check for possible erosion problems; or a researcher may require information on solar energy in order to model water loss through evapotranspiration. Unfortunately, slope and aspect data are not generally available in digital form.

Personnel at the Remote Sensing Research Program (RSRP) at the Space Sciences Laboratory, University of California at Berkeley, use a raster-based GIS, and have developed an algorithm for generating slope and aspect from available digital elevation data. Examples of these data include Digital Elevation Models (DEMs) available from the U.S. Geological Survey and the older Digital Terrain Tapes (DTTs) from the Defense Mapping Agency. Given the elevation for all cells (or pixels) in the raster data base, the algorithm produces a slope and aspect value for each pixel.

The algorithm uses certain aspects of vector mathematics. For those unfamiliar with vectors, they can perhaps best be visualized as an arrow, or as a line segment with a tail at one end and a point at the other end. In three-dimensional space vectors are described by a triplet of numbers, usually denoted $\mathbf{v} = (\Delta x, \Delta y, \Delta z)$, where the values of Δx, Δy, and Δz are the distances along the three-dimensional axes one has to travel to get from the vector's tail to its head.

This algorithm generates slope and aspect values for a pixel from its "normal vector," denoted **n**. Considering the pixel as a small rectangular plane, the normal vector is an arrow with its tail in the center of the pixel pointing at right angles to the pixel's plane. From the amount and direction of tilt of this vector, one can derive the slope and aspect of the pixel.

In describing the algorithm, a few assumptions have been made. It is assumed that the elevation data are entered into the database with true north-south aligned with the database's Y axis and east-west oriented with the X axis. It is also assumed that the pixels are square rather than rectangular, so that the distance between pixels is the same in the X and Y directions.

CALCULATING THE PIXEL NORMAL VECTOR

The normal vector, **n**, is calculated using the elevation values of the four immediately adjacent pixels. Consider the pixel P_0 in Figure 1. A vector with its tail at the center of pixel P_1 and its point at the center of P_3 is shown. This vector, denoted $\mathbf{n}_e$, gives an "east-west tilt" for pixel P_0. Vector $\mathbf{n}_e$ is the triplet $(2d,0,e_3-e_1)$, where d is the distance between pixel centers and e_3 and e_1 are the elevations of pixels P_3 and P_1, respectively.

Similarly, we can define a vector $\mathbf{n}_s = (0,2d,e_2-e_4)$ from pixel P_4 to pixel P_2 giving the "north-south tilt" of pixel P_0. If P_0 happens to be on the edge of the database, its elevation is used in place of the missing pixel.

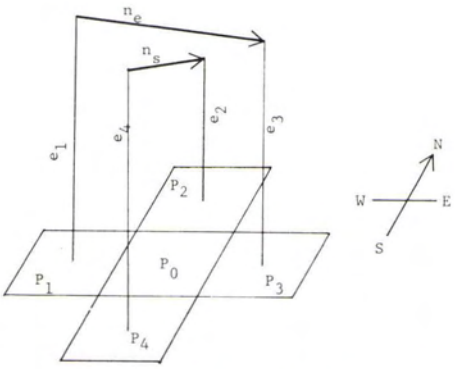

FIG. 1. The relationship between the pixel of interest, P_0 and the four neighboring pixels, shown in an oblique view. The boxes labeled P_0 through P_5 are the "sea-level" equivalent of the pixels. The lines labeled e_1 through e_4 give the magnitude of the pixel elevations, rising from the pixel centers. Also shown are the vectors $\mathbf{n}_e$ and $\mathbf{n}_s$.

The cross product of any two vectors is a third vector that is orthogonal (at right angles) to both of the original vectors. Because vectors $\mathbf{n}_s$ and $\mathbf{n}_e$ are orthogonal to each other and parallel to the plane of P_0, the cross product $\mathbf{n}_s \times \mathbf{n}_e$ will produce a vector that is orthogonal to the plane of P_0. This is the normal vector **n**.

Using the definition of the cross product (actually there are two definitions; this discussion uses the one that will produce vector **n** pointing out toward space rather than into the earth), the formula is

$$\begin{aligned}\mathbf{n} = \mathbf{n}_e \times \mathbf{n}_s &= (2d,0,e_3-e_1) \times (0,2d,e_2-e_1) \\ &= (-2d(e_3-e_1), -2d(e_2-e_4), 4d^2)\end{aligned}$$

This result can be simplified by dividing each term by $2d$ and moving the minus inside the parentheses, producing

$$\mathbf{n} = (e_1-e_3, e_4-e_2, 2d) \quad (1)$$

Note that this algorithm is applicable even if the database has rectangular pixels such that the distance between pixel centers is different in the east-west direction than in the north-south direction. In this situation the vectors are described with $\mathbf{n}_e$ having a $2d_x$ term and $\mathbf{n}_s$ a $2d_y$ term, and the cross product cannot be reduced by dividing through by the $2d$ term.

CALCULATION OF THE SLOPE

The slope of any pixel is the angle of the pixel's plane with respect to the horizontal plane. This algorithm produces slope

PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING,
Vol. 53, No. 8, August 1987, pp. 1109–1111.

0099-1112/87/5308–1109$02.25/0

as a percent of rise over run. A slope of 100 percent, for example, represents a 45 degree slope because rise and run are equal. Consider a pixel's normal vector $\mathbf{n} = (n_x, n_y, n_z)$, as shown in Figure 2. The slope of this vector is the ratio of its rise over run. The rise of the vector is the z component n_z, while the run is shown as the dotted line in Figure 2. By the Pythagorian Theorem, the length of this line is $\sqrt{n_x^2 + n_y^2}$, so the slope of the vector is

$$\text{slope}(\mathbf{n}) = \frac{n_z}{\sqrt{n_x^2 + n_y^2}}.$$

Because the vector is at right angles to the pixel's plane, the rise over run of the vector is the same as the run over rise of the pixel. By inverting the normal vector's slope formula, one can get the rise over run of the pixel's plane: i.e.,

$$\text{slope(pixel)} = \frac{\sqrt{n_x^2 + n_y^2}}{n_z}.$$

Substituting the values from Equation 1 for n_x, n_y, and n_z, and multiplying by 100 to convert to percent, results in the formula

$$\text{Slope of } P_0 \text{ in percent} = 100\frac{\sqrt{(e_1 - e_3)^2 + (e_4 - e_2)^2}}{2d}.$$

ASPECT COMPUTATION

The pixel's aspect is determined from the projection of its normal vector into the horizontal plane. This projection is shown by the dotted line in Figure 2. Figure 3 shows the same line as viewed from directly above the horizontal plane. The angle α between the projection and the X axis can be obtained from the trigonometric function arctangent:

$$\alpha = \arctan\left(\frac{n_y}{n_x}\right)$$

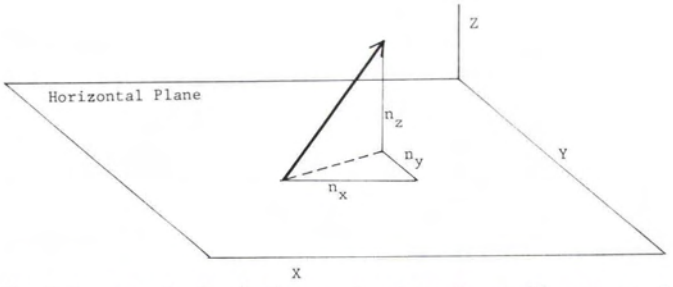

FIG. 2. An example of a pixel's normal vector **n** shown with respect to the horizontal plane. The vector is pointing up, to the right, and away from the viewer. This is indicated by the tracing of its three components, n_x, n^y, and n_z. The dotted line shows the vector's projection onto the horizontal plane.

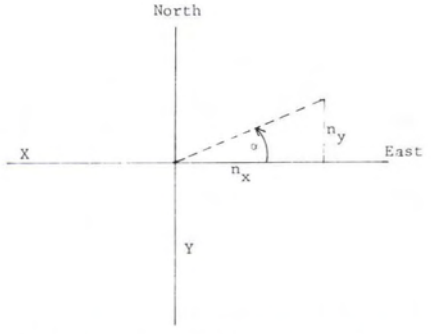

FIG. 3. A view of the normal vector **n** projected into the horizontal plane.

However, the arctangent function produces values relative to the X axis, while what is wanted is the angle relative to due north, the Y axis. Figure 4 makes this more clear. Most computer implementations of the trigonometric functions return values in radians, so in order to convert the value returned by the arctangent function to the desired degrees azimuth it must be converted to degrees, then adjusted relative to 90 degrees if n_x is positive, or 270 degrees if n_x is negative. For example, if the angle α in Figure 3 is 30 degrees, the arctangent function will return the value $\pi/6$ radians. The appropriate correction must be applied to convert the value $\pi/6$ to 60 degrees, the angle relative to North.

Using 57.296 as the radian-to-degree conversion factor, the formula for aspect in degrees azimuth is

$$\text{aspect} = 90 - 57.296\left(\arctan\left(\frac{n_y}{n_x}\right)\right) \quad \text{if } n_x > 0$$

$$\text{aspect} = 270 - 57.296\left(\arctan\left(\frac{n_y}{n_x}\right)\right) \quad \text{if } n_x < 0$$

To avoid dividing by zero, vectors with $n_x = 0$ must be treated as a special case. If the x-component of the vector is zero, its projection must fall along the Y axis; the aspect is either 360 (due north) or 180 (due south), depending on the sign of n_y. (The value 0 is reserved for flat pixels, which have no defined aspect.) Substituting the values from Equation 1 for n_x and n_y produces the formulas for aspect when n_x is non-zero: i.e.,

$$\text{aspect} = 90 - 57.296\left(\arctan\left(\frac{e_4 - e_2}{e_1 - e_3}\right)\right) \quad \text{if } (e_1 - e_3) > 0$$

$$\text{aspect} = 270 - 57.296\left(\arctan\left(\frac{e_4 - e_2}{e_1 - e_3}\right)\right) \quad \text{if } (e_1 - e_3) < 0$$

COMPARING OTHER SLOPE AND ASPECT ALGORITHMS

The Earth Laboratory Applications Software (ELAS) is a public-domain image processing software package that is in use at a number of locations in the U.S. (Earth Resources Laboratory, 1986). The ELAS module TOP6 produces slope, aspect, and slope length using a 3-by-3-window neighborhood approach. Using the difference in elevations and the distance between pixel centers, the slope between the object pixel and each of its eight neighbors is computed. These eight slope values are examined and the largest is selected as the slope of the object pixel. To

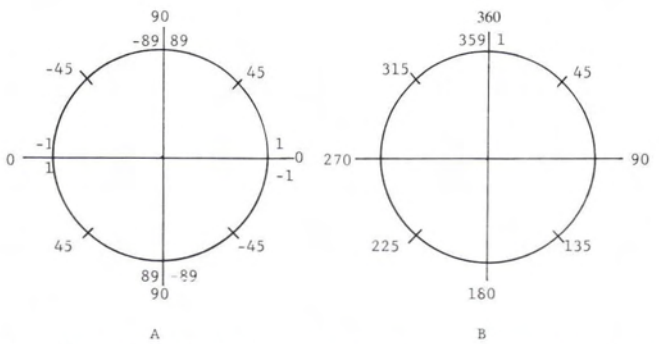

FIG. 4. This figure illustrates the difference between degrees azimuth and the values returned by the arctangent function. Figure 4A gives the values returned by the arctangent function (in degrees), while their equivalent in degrees azimuth are shown in Figure 4B.

get the aspect of the object pixel, its elevation and the elevation of the pixel producing the largest slope are compared. The aspect is defined as the direction from the higher elevation to the lower elevation.

The TOP6 module examines all eight neighboring pixels, but only two pixels are actually used to determine slope and aspect, namely the object pixel and one other. The elevations of other neighboring pixels are ignored. For aspect, only flat or one of the eight multiples of 45 degrees can be obtained.

A method for determining slope, aspect, and surface curvature using a two-dimensional Fourier transform is described by Papo and Gelman (1984). Briefly, their approach represents the elevation surface as a polynomial function; the slope magnitude and slope direction (aspect) can be derived from the first derivative of this function while the second derivative can give surface curvature. In the application, this technique is achieved by converting the elevation grid from the space domain to the spatial-frequency domain by use of a two-dimensional fast Fourier transform (FFT). Slope, aspect, and curvature can then be derived by applying appropriate factors to the spatial-frequency data, then using an inverse FFT to return to the space domain.

This method offers considerable information, but at a cost. The use of the FFT not only provides slope and aspect information that considers contributions from all the surrounding pixels, but can also provide, by examination of the spectral density matrix, insights into the inherent characteristics of the topography. It is, however, considerably more compute-intensive than either ELAS's method or the vector-based algorithm.

The Land Analysis System software (U.S. Geological Survey, 1984), a public domain software package in use at NASA's Goddard Space Flight Center, includes a module called SLAP that has four methods of computing slope and aspect from elevation data. All four methods utilize a 3 by 3 matrix of elevation pixels with the object pixel in the center. Three of these methods, called "maximum drop," "maximum," and "minimum," are variations of the ELAS procedure described above. Each method computes slope using the object pixel and one of its eight neighbors, and produces aspect as one of eight possible directions. The three methods differ only in the way the neighbor pixel is chosen.

The fourth, called the "mean" method, uses a more sophisticated approach similar to the vector-based method. Two different slopes are calculated; the "vertical" slope takes the average elevation of the top three pixels of the 3 by 3 window, subtracts the average elevation of the bottom three pixels, then divides by the distance between the two rows. In a similar way, a "horizontal" slope is computed using the right three pixels' average minus the left three pixels' average. The percent slope of the object pixel is then $100\sqrt{vslope^2 + hslope^2}$.

The aspect computation in SLAP takes the arc tangent function of the vertical slope over the horizontal slope. This value is converted from radians to degrees, then adjusted to the correct quadrant depending on the signs of the slopes.

COMPUTER IMPLEMENTATION

The Pascal code for the core of the algorithm is given in Figure 5. A calling program would be written to read the elevation data, pass the values to this procedure, then scale the returned slope and aspect values appropriately and output them. Fortran 77 code would be similar, although in Fortran the use of the atan2 function instead of atan could save one conditional test in the calculation of aspect.

```
procedure slpasp( e1,e2,e3,e4,d : integer; var slp,asp : real);

{ input values e1 thru e4 are the elevation values, d is the}
{ distance between two pixels. The procedure returns percent}
{ slope in 'slp', aspect in degrees azimuth in 'asp', with  }
{ 0.0 for flat, 360.0 = due north. slpasp is called once for}
{ each pixel in the database.}

const r2d = 57.296;       {radians-to-degrees conversion }

var nx,ny : integer;  {temporary variables}

begin
     nx := e1-e3; ny := e4-e2;

     {produce the slope in percent }
     slp := 100*sqrt(nx*nx + ny*ny) / (2*d);

     {Compute aspect.}
     if slp = 0.0 then asp := 0.0     {If slope is zero aspect is zero.}
     else if nx = 0 then begin        {don't divide by 0. If nx = 0... }
          if ny < 0 then asp := 180.0   {then aspect is either due S... }
          else asp := 360.0                {or due North, depending on ny.  }
          end
     else begin
          if nx > 0 then              {check if east or west facing...}
               asp := 90 - r2d * arctan(ny/nx)     {and adjust accordingly}
          else asp := 270 - r2d * arctan(ny/nx)
     end
end;
```

FIG. 5. The Pascal code for the algorithm.

SUMMARY

The vector-based algorithm described here is a good compromise between accuracy and computational complexity. A small amount of precision is lost by ignoring the contributions from the diagonally adjacent pixels when calculating the slope and aspect, but it is more accurate than algorithms that produce slope and aspect from only two pixels. Slope and aspect values that are produced by this algorithm are more precise than required for most uses. In actual practice, the slope variable is usually "sliced" into five percent increments, while the 360 degrees of aspect are converted into 48 categories at best. Under these conditions, the advantages of a relatively simple and easily implemented algorithm outweigh the loss of accuracy.

ACKNOWLEDGMENT

This algorithm, in a slightly different form, was first described to the author in the mid-1970s by Francis Harvey, then with the RSRP.

REFERENCES

Earth Resources Laboratory, 1986. *Earth Resources Laboratory Applications Software, Vol. II. User Reference*, Jan. 1986, pp. A48–A49. NASA National Space Technology Laboratories Earth Resources Laboratory Report No. 183, NSTL City, Mississippi.

Papo, H.B., and E. Gelman, 1984. Digital Terrain Models for Slopes and Curvatures, *Photogrammetric Engineering and Remote Sensing*, Vol. 50, pp. 695–701.

U.S. Geological Survey, 1984. *Land Analysis System User's Guide*, USGS EROS Data Center, Sioux Falls, South Dakota.

(Received 15 February 1986; revised and accepted 24 April 1987)