A NEW APPROACH TO RELIEF REPRESENTATION

J. Raul Ramirez, PhD
The Ohio State University Center for Mapping, Columbus, Ohio, USA

ABSTRACT
Relief representation is a difficult problem. The surface of the Earth cannot be defined by a physical or mathematical representation but can be approximated by digital elevation models (DEM), triangular irregular networks (TIN), and contour lines. Each representation carries approximation errors.

A DEM introduces the greatest errors because we assume that the surface of the Earth can be approximated to a regular grid of elevations. A TIN introduces less error than a DEM because we use an irregular network of triangles to represent the relief. Another representation is by contour lines. Contour lines describe the relief without approximation along each contour. Contour lines are the relief representation that introduces the least amount of error.

In this paper we will present a new approach to relief representation using linear features. The new encoding approach is an extension of the Freeman chain code, where the regular coding grid is replaced by a circular structure.

INTRODUCTION
Relief representation is a difficult problem. The Earth and/or its surface cannot be defined by a physical or by a mathematical representation. We use a figure such as the geoid, which has a physical representation or the ellipsoid that has a physical and a geometric representation to approximate the Earth. In computers we approximate the representation of the surface of the Earth by different digital elevation models (DEM). These digital models carry different degrees of approximation. Examples of them are grid digital elevation models (GDEM), triangular irregular networks (TIN), and contour lines.

GDEM and TIN are terrain representations developed exclusively for a computer environment. Contour representation is used in conventional hard-copy maps (for many years) and it is also used in computers. There is not an universal terminology to define the computer-based representation of the relief. For example, Burrough (1986) defines a DEM as “any digital representation of the continuous variation of the relief over space.” This definition includes the three representations mentioned here: GDEM, TIN, and contour lines. The Harvard Design School defines a DEM as “a type of raster GIS layer. Raster GIS represents the world as a regular arrangement of locations. In a DEM, each cell has a value corresponding to its elevation.” In the “Spatial Data Transfer Standard Mapping of the USGS” the USGS Mid-Continent Mapping Center Branch of Research, Technology and Applications defines a DEM as “Terrain elevations at regularly spaced horizontal intervals, i.e., an a grid of regularly spaced elevations.” In this paper we use the term digital elevation model (DEM) in the context of Burrough (1986).

Figure 1. Example of a GDEM

Stair-Step Representation

One elevation value per Grid.

Information lost: No idea about neighbors
Of the three digital models of the relief, a GDEM is the simplest one. In it, we assume that the surface of the Earth can be approximated to a regular grid of elevations resulting in a stair-step, discontinuous representation of the relief. This approach introduces the greatest amount of errors. Figure 1 illustrates this representation. Of course, the greater the size of the grid meshes, the greater the approximation of the surface representation.

A TIN represents a surface by a set of contiguous, non-overlapping triangles. An elevation value is assigned to each triangle vertex. This approach introduces less error than a DEM because we use an irregular network of triangles to represent the relief. An irregular network of triangles of different sizes fit better to the Earth than a regular grid. We assume that the area covered by each triangle can be modeled by a planar surface. This planar surface is, of course, the approximation in the triangular irregular network. Elevations inside each triangle can be interpolated from the information of the vertices. Figure 2 illustrates the idea of a TIN.

The third representation of the terrain relief is by contour lines. Contour lines are lines drawn on a map connecting points of equal elevation. If you walk along a contour line, you neither gain nor lose elevation. Contour lines describe the relief without approximation along each contour. In theory, if we could collect contour lines with contour intervals of 5 cm, we would have a relief representation with errors no greater than 5 cm. If we could collect contour lines with contour intervals of 1 cm, we would have a relief representation with errors no greater than 1 cm and so forth. This is the representation of the relief that introduces the least amount of error, especially when small contour intervals are used. Figure 3 is an example of contour lines with a contour interval of 4.

From the viewpoint of digital data storage, GDEMs are more efficient to store than TINs and contour lines. Contour lines are the least efficient to store. GDEM are stored with a header that defines the coordinates origin.
(for example, the upper right and lower left corners of the grid), the length of each grid mesh, type of measurement units, number of elevations per record, direction of data storage, followed by a set of records containing the elevation value of each grid mesh. The fact that planimetric coordinates of each grid mesh do not need to be stored makes this digital data storage very efficient. On the other hand, there is a major loss of information when relief information is stored in a DEM because we assume a regular terrain representation that in general does not reflect the true nature of the area represented.

In a GDEM we assume that a regular grid can approximate the representation of the relief of the surface of the Earth. The basic unit in this representation is the regular grid mesh. Each grid mesh carries an elevation value for the whole mesh, which is the explicit information carried by the basic unit. We know the geospatial location of each grid mesh, but if we occupy a particular grid mesh, there is not any type of information about the neighbors. If, for example, all eight surrounding grid meshes have the same elevation as the one we occupy, we only can determine this by examining all eight neighbors. This is similar to the problem of pixel information in digital images. This kind of limited information is a major source of concern when we need to perform an analysis of relief data because the process will force us to access more grids that we may want.

TINs can be stored in two different forms: triangle-by-triangle, and points and their neighbors. In the triangle-by-triangle approach, each computer record contains: an identification number for each triangle, the absolute coordinates (X, Y, Z) for the three vertices, and the identification numbers of the three neighboring triangles. A way to make this storage a more efficient approach is based on the fact that a vertex participates in average in six triangles. In each record a vertex identification (instead of the absolute coordinate values) is stored and there is a separate file containing vertices identification and the corresponding absolute coordinate values. The points and their neighbors approach stores for each vertex an identification number, the absolute X, Y, and Z coordinates, and the pointers (reference) to the neighboring vertices in clockwise or counter-clockwise order.

TINs carry more explicit and implicit information than GDEMs. Each triangle carries explicitly its location, the elevation of its vertices, and the identification to adjacent triangles (or vertices). They also carry implicitly the approximate elevations of all points inside the triangle.

Contour lines are stored as an ordered sequence of points. In general, each computer record stores a contour identification, the elevation value, and a set of planimetric coordinates (X, Y) in a given absolute coordinate system. A long contour may have literally thousands of points and it may be stored in many computer records.

Contour lines carry more explicit and implicit information than GDEMs and TINS. Each contour line shows explicitly an elevation and all the points along the contour line that have the same elevation on the ground. Contour lines describe a continuous path along each contour trajectory and define closed planes for the same elevation. These planes are always parallel to each other. Contour lines as a whole carry implicit elevations for those areas of the surface not traveled by contours. The elevation for any point in those areas can be computed from the contour elevations.

In this paper we will present several new ideas about relief representation using linear features. We will present two basic ideas: (1) a new encoding approach of relief using long lines, and (2) a cost-efficient data organization method for the retrieval and analysis of relief information.

ENCODING OF RELIEF USING LONG LINES

Ground objects represented by linear features are of great interest in mapping and GIS. Examples of linear features are roads, waterways, and terrain contours. Conventionally, they are encoded in computer files as a set of ordered absolute planar coordinate values (X, Y) or as latitude and longitude (ϕ, λ).

These computer representations for planimetric features, such as roads and waterways, are fine if the user is interested only in their graphic representation. In the case of contours, this alone gives a poor visual representation of the relief, if the user intends to look at the surface from a different viewpoint that just from above. But, all these computer representations are problematic, if we are interested in performing analyses for portions of those linear features. For example, if we need to answer the question, what linear features pass through an area of interest, these representations are inefficient. Figure 4 shows an example of this inefficiency.

To answer the question – Which lines go through the red box? – We will need to test all the points of the six lines against the location of the red box. The reason is that there is no contour information related to specific areas on the surface and we will need to check all the points to find it. In the case of TINs a similar process needs to be followed and may require testing all the triangles to answer the question of what triangles are in the red
box. In the case of GDEM, it will be a lot easier to find what grids are in that box just because of the raster nature of the GDEM representation. Therefore, finding what contours (or triangles) are in a specific area is time consuming and costly.

We would like to review traditional methods of encoding linear features in computer files as a first step to presenting our approach.

**Encoding of Lines: Freeman Code**

The Freeman chain code (Freeman 1974) was developed in the 1970’s to represent drawing lines in computers (quantization of lines). It was considered an improvement over square-box and grid-intersect quantization, which were the methods used at that time. Let us start by understanding these two quantization methods and their relation to the Freeman chain code. Figure 5 illustrates the idea behind the square-box and grid-intersect.

Figure 5-a shows a line to be encoded in a computer. Figure 5-b shows the result using the square-box quantization. The original line is replaced by the line (0, 1) (0, 2) (1, 2) (2, 2) (2, 3) (3, 3) (3, 2) (4, 2) (4, 3). Figure 5-c shows the result of the grid-intersect quantization. The original line is replaced by the line (0, 1) (1,2) (2, 3) (3, 2) (4, 3).

Figure 6 illustrates the idea of the Freeman Chain Code. Figure 6-a shows the result of using the grid-intersect quantization approach. Figure 6-b shows the chain-coding schema. This schema is based on the fact that “each node in sequence coincides with one of the eight grid nodes that surround the previous data node” (Freeman, 1974). As indicated by Freeman (1974) “if we label these eight neighboring grid nodes from 1 to 8 in a grid.
counterclockwise sense starting from the positive x-axis we can represent the line structure simply by the sequence of octal digits.” Figure 6-c shows the line and the corresponding octal digits that describe the line. This line is fully described by the Freeman Chain Code expression: 2 2 8 2.

**Encoding of Lines: Extending Freeman Code to the Vector Domain**

The Freeman Chain Code has proven to be very helpful in the computer encoding of linear features. However, this code was derived about 30 years ago when computer capabilities were limited by today’s standards. Perhaps a fresh look of the chain code would result in new ways to encode computer lines. A possibility is to examine the chain-coding schema in the vector domain. To do so, we could replace the regular coding grid used in the Freeman chain code by a circular structure. For example, let us consider a circle divided into 45° arcs. The result is a circle divided into eight segments (eight directions from the center of the circle). This structure shown in Figure 7 could be used to express the trajectory of the line AB in a similar fashion as the Freeman chain-coding schema. Using the circular schema, the line can be fully expressed by the coding 2 2 8 2.

![Figure 7. Extended Freeman Code-First Approximation](image)

The above approach reproduces the traditional Freeman Chain encoding with all its advantages and limitations. A major limitation is the approximate representation of the original line (see Figure 5-a). A better approximation to this line, line abedfgh, is shown in Figure 8-b (Figure 8-a is the original line shown in Figure 5-a). This is the kind of representation we are storing today (in vector format) encoded in computer files as a set of ordered absolute planar coordinate values (X, Y) or as latitude and longitude (φ, λ), as mentioned earlier. The question we would like to consider next is: Can we use something similar to the Freeman Chain code to store this better approximation in a computer file?

![Figure 8. A Better Line Approximation](image)

Let us study the Freeman chain-coding schema in greater detail to try to answer the above question. The Freeman chain-coding schema assumes a grid space where regular meshes exist. Under this assumption, 45° direction angles always express the location of the next node from the previous one. The distance of any line segment is always equal to d or d√2, where d is the width of a mesh. Freeman chain-coding schema is very efficient from the viewpoint of data storage compared with the square-box and grid-intersect methods. The encoding of a line using Freeman Chain Code can be done in 50% of the space needed by either of the other two methods.

Of course, without removing Freeman’s assumptions it is not possible to store a better approximation of a line because very few vector lines are given by line segments whose relative directions are all multiple of 45° and with all segments of equal length. Let us start by replacing the first constraint (45° direction angles) with a less restricted one.
Let us introduce a larger number of direction angles. In the vector domain we may improve the quality of the description of a line by increasing the number of direction angles, as shown in Figure 9 (12, 24, and 72 direction angles respectively). Of course many other sets of direction angles can be derived. As an example, we will work only with these three.

Let us see what kind of improvement we can get using these finer direction angles. To keep the second Freeman’s assumption, let us divide the line in equal length segments by dividing the length of the line by the number of line segments in the original line approximation. Figure 10-a shows the original line approximation that has seven line segments of different lengths. Figure 10-b shows the resulting line approximation with equal length segments.

We can express Line 10-b using one of the three sets of direction angles of Figure 9. Figure 9-a will give us the coarser representation and Figure 9-c the finer. Using Figure 9-a the expression of the line will be $3 2 2 1 1 2 1 3$; using Figure 9-b the expression of the line will be $5 4 2 2 3 2 4 4$; and using Figure 9-c the expression of the line will be $1 4 8 8 3 6 7 1 1 1$. If we use these encoding sequences to draw the line we will get the lines shown in Figure 11.

A graphical comparison of these lines with the original one shows an improvement in the visualization of the line with the use of finer direction angles, as shown in Figure 12.
It is possible to use even finer direction angles (1°, 30°, and so forth) and in such a case, we can get more precise line encoding. The major advantage of this type of encoding is its efficiency from the data storage viewpoint. Under the assumption that an equal line segment replaces the original line, the absolute coordinates of the origin, the length of the segment, the size of the direction angle, and a chain code of integer numbers describing the sequential direction angles can encode a line. This kind of encoding is at least 50% more efficient that encoding the same line by absolute coordinate values. The major disadvantage of this approach is the fact that the original line needs to be replaced by an equal-length line segment that introduces approximation errors. These errors increase in lines whose direction changes with steep angles and have short and long distances. For these lines, the larger the number of these angles, the larger the possibility of greater approximation errors.

A possible way to eliminate the restriction of our line encoding method is to remove the second assumption in the Freeman’s chain code schema: having the line description based on a regular grid. This can be accomplished by introducing a new parameter, DISTANCE. DISTANCE is the length of each line segment. This approach will eliminate most of the approximation in encoding a line. A long line can be encoded to almost any degree of accuracy by a chain code of the type \( d_1 \alpha_1, d_2 \alpha_2, d_3 \alpha_3, d_4 \alpha_4, \ldots \), where \( d_i \) is the corresponding length segment and \( \alpha_i \) is the corresponding direction angle. Figure 14 illustrates this idea.

In this figure, the line \( AG \) is composed of the line segments \( Ab, bc, cd, de, ef, \) and \( fg \). Segment \( Ab \) is described by the distance \( d_{ab} \) and the direction angle \( \alpha_{ab} \). The segment \( bc \) is described by the distance \( d_{bc} \) and the direction
angle $\alpha_{bc}$ and so forth. Let us consider the line given in Figure 10-a and its representation by distances and direction angles.

![Figure 15. Line Representation by Distances and Direction Angles](image)

Figure 15-a shows the original line from Figure 10-a; Figures 15-b and 15-c show the line encoded with distances and 30° direction angles. The resulting encoded line is shown alone in Figure 15-b and overlaid on the original line in Figure 15-c. Figures 15-d and 15-e show the line encoded with distances and 15° direction angles. The resulting encoded line is shown alone in Figure 15-d and overlaid on the original line in Figure 15-e. Figures 15-f and 15-g show the line encoded with distances and 5° direction angles. The resulting line is shown alone in Figure 15-f and overlaid on the original line in Figure 15-g.

As indicated earlier, it is possible to use finer direction angles (1°, 30°, and so forth). In such a case we can get more precise line encoding. A major advantage of this type of encoding is its efficiency from the data storage viewpoint. The absolute coordinates of the origin, the length of each segment, the size of the direction angle, and a chain code of integer numbers describing the sequential direction angles can encode a line. Depending on how we encode the line segment lengths and with what accuracy will decide the efficiency of this encoding method from the data storage viewpoint. In all cases this approach will be more efficient than the one based on absolute coordinate values.

CONCLUSIONS

We have presented a new line-encoding scheme that uses an auxiliary grid mesh to provide for cost-efficient storage and retrieval of relief information. This approach should provide a higher quality elevation data set than DEMs and TINs alone, because it allows storing the most precise relief description possible. Our approach should allow for an easy image registration and client applications (targeting, route planning, visibility) because data retrieval is very cost-efficient and fast. We also presented the idea of improved relief visualization by combining our line-encoding schema with a TIN data structure. This should result in a more precise representation and visualization of the terrain surface than current methods.

We believe these ideas can improve the storage, retrieval, analysis, and query of elevation information, and therefore, should be explored further by continuing the development of the theoretical framework. We propose to implement this approach for several test cases, including different types of relief.

REFERENCES