Conversion of Cartesian Coordinates from and to Generalized Balanced Ternary Addresses

Jan W. van Roessel
TGS Technology, Inc., EROS Data Center, Sioux Falls, SD 57198

ABSTRACT: Hexagonal grids have several advantages over square grids, such as a greater angular resolution and unambiguous connectivity. The Generalized Balanced Ternary (GBT) system is a spatial addressing method for hexagonal grids in which the hexagons are arranged in hierarchical aggregates, and which accommodates vector operations in GBT space. Efficient algorithms for converting Cartesian coordinates from and to GBT addresses are based on the dual representation of the hexagonal tessellation. The GBT-to-Cartesian algorithm is an order of magnitude faster than the Cartesian-to-GBT algorithm, the latter requiring interpolation and GBT addition for each digit of the generated GBT address.

INTRODUCTION

MOST GRIDDED OR RASTER REPRESENTATIONS used in remote sensing and GIS applications use rectangular pixels or grid cells. There are alternatives, however, as a two-dimensional plane can also be tessellated (covered with tiles) with triangles or hexagons. We will be concerned with hexagons. Hexagonal arrays have been used in pattern recognition for many years (Crettez, 1980). Applications in remote sensing have not been very abundant, but some do exist (Gibson and Lucas, 1982). Hexagonal patches are employed in the Gestalt orthophotomapping process (Crawley, 1975). Hexagons have been used to construct elevation models, and to form slope or aspect polygons (polyhexes) from adjacent hexagons with identical slope or aspect (van Roessel *et al.*, 1978). One firm has developed a spatial database management system based on hexagons (Gibson and Lucas), and has built a GIS around this system. Hexagonal grid cells have attracted proponents of their use in defense simulations, while others in the defense establishment have tried to exorcise "hexes" (H. M. Sassenfeld, written commun., 1979).

The advantages and disadvantages of hexagonal versus rectangular systems have been expounded in several instances. Crettez (1980) mentions that a hexagonal tessellation minimizes the gap sizes between circular points, and removes the connectivity ambiguity present in the rectangular tessellation (four-way and eight-way connectedness). He goes on to say that it does not present any preferential directions, offers a greater angular resolution, is more alike to the mosaic of retinal receptive fields at the fovea level, and suits the representation of optical images which have a circular symmetry like the visual field. Samet (1984) points out that hexagons limit basic resolution because, unlike rectangles, hexagons cannot be further subdivided into smaller hexagons. Furthermore, Mandelbrot (1982) states that ". . . it is a widespread source of irritation that hexagons put together do not quite make up a bigger hexagon." Instead hexagons can be grouped into aggregates (Gibson and Lucas, n.d.), also called heptuplets (Crettez), rosettes (Samet), or Gosper islands (Mandelbrot). First-level aggregates (seven hexagons) can be grouped into second-level aggregates, and so on. A hexagonal tessellation and organization into aggregates is shown in Figure 1.

To use hexagons, an efficient way to convert from the rectangular to the hexagonal representation must be available. Two major steps are involved in this process: coordinate conversions and resampling. We will address the first problem, and then present an efficient way to convert Cartesian coordinates to GBT (Generalized Balanced Ternary) addresses and vice-versa.

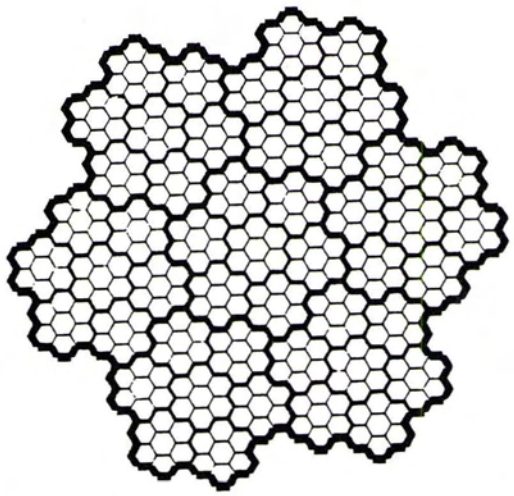

FIG. 1. Hexagonal tessellation and aggregate organization.

GENERALIZED BALANCED TERNARY

The GBT system was developed by Gibson and Lucas (1982), who describe it as a hierarchical addressing system for Euclidean space that has a useful algebraic structure. It consists of a hierarchy of cells. At each level, cells are constructed from the previous level according to a rule of aggregation. In the second dimension, the cell is a hexagon and the aggregates are those of Figure 1. GBT is a generalization of the balanced ternary (BT) system, a system praised by Knuth (1969) as ". . . the prettiest numbering system of all . . . ." Instead of bits, BT uses "trits" $(-1,0,1)$, symmetric about the origin. Knuth states that this system may become important some day when the "flip-flop" will be replaced by a "flip-flap-flop." Although Lucas and Gibson generalized BT to higher dimensions, we will only be concerned here with the second dimension.

In two dimensions the integers 0 to 6 are used to index a cluster of hexagons (see Figure 2), where the hexagons labeled 1 to 6 are placed around the central 0 hexagon, such that the distance between the integers which are a power of 2 (1,2,4) is uniform and maximal (Gibson and Lucas, 1982). The purpose for this arrangement, which is different from the one employed by Crettez, is to create an efficient method for vector addition and multiplication using GBT addresses. These GBT functions correspond to regular vector operations in Euclidean space.

PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING,
Vol. 54, No. 11, November 1988, pp. 1565-1570.

0099-1112/88/5411-1565$02.25/0

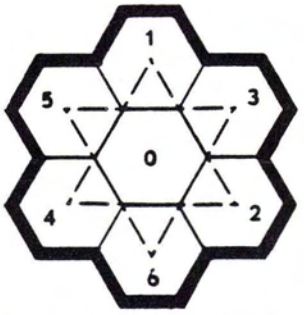

FIG. 2. GBT hexagon indexing.

TABLE 1. ADDITION TABLE

	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	12	3	34	5	16	0
2	2	3	24	25	6	0	61
3	3	34	25	36	0	1	2
4	4	5	6	0	41	52	43
5	5	16	0	1	52	53	4
6	6	0	61	2	43	4	65

Each GBT address g consists of a series of sequential digits g_i, each with a range from 0 to 6, where $i = 1, \ldots, n$, and n is the most significant digit. g_1 represents the first aggregate level, g_2 the second, and so on.

We will only be concerned with GBT addition for purposes of coordinate conversion. Addition of GBT addresses works much the same as ordinary integer addition. The numbers are added digit by digit, starting with the least-significant position. The digit by digit result is determined by an addition table (Table 1) (Lucas and Gibson).

In Table 1, 4 + 6 = 43 for example, which means that a 0-4 vector added to a 0-6 vector yields a vector terminating in the third hexagon of the fourth first-level aggregate. A difference from ordinary addition is that one may accumulate multiple carries. For instance in the addition 44 + 56 = 563 of the following example, we first add 4 + 6 = 43 according to the table. The 3 of 43 is the least significant digit of the answer, while the 4 carries to the next column. Adding 4 + 4 gives 41, with the 4 from 41 yielding the first carry of the left-most column. The 1 from 41 added to 5 from 56 yields 16, of which the 6 is the second digit of the answer and the 1 becomes the second carry in the left-most column. Finally, adding 4 and 1 produces 5 which is the most significant digit of the result. That is,

$$\begin{array}{r} {\scriptstyle 4} \\ {\scriptstyle 1\ 4} \\ \hline 4\ 4 \\ 5\ 6 \\ \hline 5\ 6\ 3 \end{array}$$

One advantage of GBT addressing of two-dimensional space is that the addresses can be arranged in the form of a tree. Because the GBT address is a single number, the extensive body of knowledge for one-dimensional trees can be applied to two-dimensions. Linear quad tree addressing may have a similar advantage (see Samet, 1984).

If the tree is organized in levels corresponding to the position of the GBT digit, then the two-dimensional hierarchy is the one of the hexagonal aggregates. Whereas triangular and square tessellations can be organized into quad trees, hexagons and hexagonal aggregates are most naturally arranged as sept trees, in which each node has seven children. Ahuja (1983) presents an analysis comparing the complexities of operations in a square and triangular quad tree, which appear to be the same. This type of analysis can be adapted to compare quad trees and sept trees. Given that one wants to access a randomly chosen neighbor N of a leaf node L in a tree with $d + 1$ levels, $K = 2k$ operations are required where k is the number of level changes involved in the access ($k = 1$ to reach a sibling). If p is the probability that N is a sibling of L, and P_{2k} is the probability that $K = 2k$, then $P_{2k} = (1 - p)^{k-1}p$ for $k < d$ and $P_{2d} = (1 - p)^{d-1}$ for $k = d$. The expected number of operations is then $E(K) = \sum_{k=1}^{d-1} 2kP_{2k} + 2dP_{2d}$. Ahuja points out that $p = 1/2$ for square and triangular quad trees. For sept trees, the probability that a randomly chosen neighbor of a hexagon or aggregate is a sibling is 1 for the center element, and 1/2 for the others. Thus, $p = (1/7)1 + 6/7(1/2) = 4/7$. For $d = 2$, $E(K)$ evaluates to 2.86 and 3.00 for sept and quad trees, respectively (one level change is always required), and for $d = 3$ the results are 3.22 and 3.50. For $d = 8$ the expected number of operations is 3.50 and 3.98. Therefore, for this type of analysis, it appears that sept trees may have an operations complexity advantage over quad trees.

Another advantage of a single spatial address is that a GBT image (an array of pixel values corresponding to GBT addresses) can be stored as a one-dimensional array, eliminating the need for two-dimensional indexing.

THE DUAL REPRESENTATION

Instead of the hexagonal tessellation shown in Figure 1, in which all the hexagonal corner points are connected into hexagons, one can connect adjacent center points of the first-level aggregates, adjacent centers of the second level aggregates, and so on.

Not surprisingly, as can be seen in Figure 3, the connected center points also form hexagons, somewhat alleviating the irritation described by Mandelbrot, because now smaller hexagons do seem to form larger hexagons, but the different size hexagons appear to be rotated with respect to those of the next larger and smaller aggregates according to a constant angle. Also, we do not get a complete tessellation of the plane, because there are gaps between the hexagons at each aggregate level. We will call the hexagons in the dual representation the dual hexagons, and refer to them as H_i, where i represents the aggregate level.

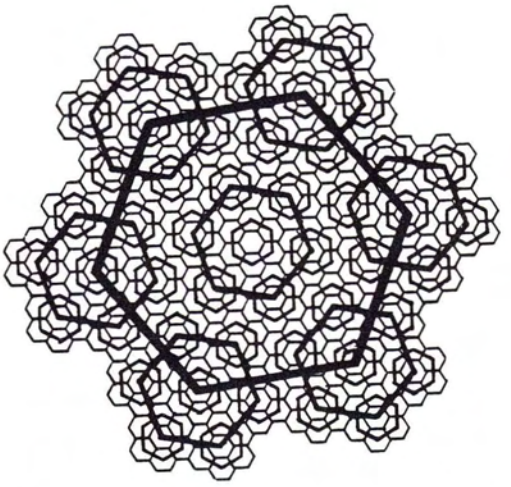

FIG. 3. The hexagonal tessellation and its dual representation.

An enlargement of a first-level aggregate and its dual hexagon is shown in Figure 4.

If r is the dimension of a side of one of the six equilateral triangles that make up a basic hexagon (the hexagon radius), then it can be seen in Figure 4 that the 2 by 7 matrix with the Cartesian coordinates of the 7 single-digit GBT addresses, in a coordinate system centered on and aligned with H_1, is as follows:

$$H = r \begin{pmatrix} 0 & 0 & 1.5 & 1.5 & -1.5 & -1.5 & 0 \\ 0 & \sqrt{3} & -\sqrt{3}/2 & \sqrt{3}/2 & -\sqrt{3}/2 & \sqrt{3}/2 & -\sqrt{3} \end{pmatrix} \quad (1)$$

and that R_1, the radius of H_1, is equal to $r\sqrt{3}$.

It can be seen from Figure 5 that R_2, the radius of H_2, equals

$$R_2 = \sqrt{(4.5r)^2 + (r\sqrt{3}/2)^2} = r\sqrt{21}. \quad (2)$$

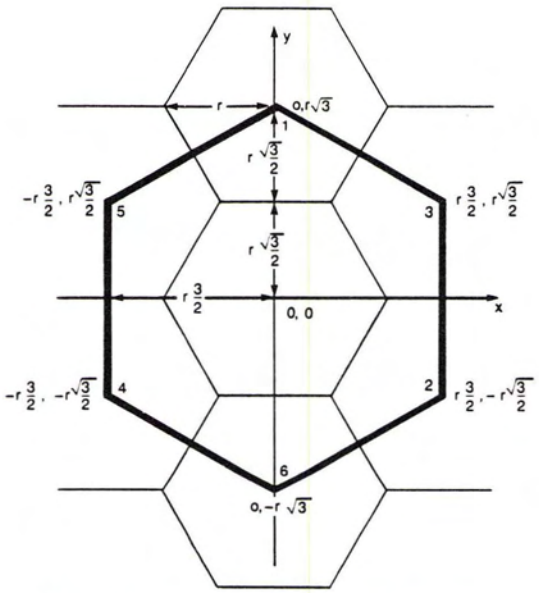

FIG. 4. First-level aggregate and dual hexagon.

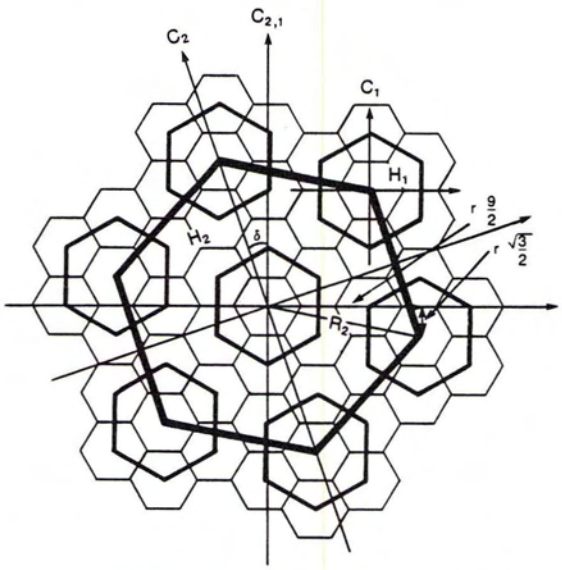

FIG. 5. Coordinate system of first- and second-level dual hexagons.

Therefore, the ratio of R_1 and R_2 is $(r\sqrt{3})/(r\sqrt{21}) = 1/\sqrt{7}$. This is also the self-similarity ratio for the Gosper islands mentioned by Mandelbrot. In general, this constant is also valid for the ratio of R_i and R_{i+1}. The ratio of R_i and R_1 can be expressed as

$$R_i/R_1 = (\sqrt{7})^{i-1}. \quad (3)$$

From Figure 5 one can also see that the tangent of the angle δ of H_2 with respect to the vertical is $(1.5r)/(2.5r\sqrt{3})$, corresponding to an angle of approximately 19.107 degrees. The geometry in the same figure shows that the sine and the cosine of this angle are equal to $\sqrt{3}/(2\sqrt{7})$ and $5/(2\sqrt{7})$, respectively.

We can now define a coordinate system C_i for each H_i, such that the system has its origin at the center of H_i, and the y axis passes through the vertex which has an address beginning with a 1 as in Figure 4. We will refer to this coordinate system as C_i. We define $C_{i,i-1}$ as a coordinate system with the origin at C_i, but with its axes aligned with those of C_{i-1}.

A rotation matrix can now be constructed for rotating C_2 to $C_{2,1}$, (axes horizontal and vertical):

$$\mathbf{M} = 1/(2\sqrt{7}) \begin{pmatrix} 5 & \sqrt{3} \\ -\sqrt{3} & 5 \end{pmatrix} \quad (4)$$

In general, this matrix can also be applied to rotate C_i to C_{i-1}. To rotate from C_i to $C_{i,1}$, a series of rotations must be applied, which can be expressed in a single matrix as follows:

$$\mathbf{M}_{i,1} = \mathbf{M}^{i-1} \quad (5)$$

where $\mathbf{M}_{1,1} = \mathbf{M}^0 = \mathbf{I}$, the identity matrix.

GBT-TO-CARTESIAN

Given the relationships described previously, the following approach for the conversion to Cartesian coordinates is now possible. Each GBT digit g_i represents a vertex of H_i. The unit coordinates of g_i in H_i are given by h_i, the column of H indicated by g_i, but need to be scaled to a size appropriate for H_i according to Equation 3. Then for each level i, C_i can be rotated to $C_{i,1}$, according to Equation 5, and the scaled coordinates can be expressed in the $C_{i,1}$ coordinate system. Repeated addition of the resulting x and y coordinates yields the Cartesian coordinates of the GBT address in $C_{n,1}$, which is centered on the largest center hexagonal aggregate with horizontal and vertical axes, and is therefore the desired coordinate system.

The process can be driven by looping through the GBT digits, starting with the least significant digit. For each, the following steps are performed:

(1) Select h_i from H, according to g_i.
(2) Obtain the scaled coordinates

$$h_i' = h_i(\sqrt{7})^{i-1}. \quad (6)$$

(3) Rotate to $C_{i,1}$ as follows:

$$h_i'' = \mathbf{M}^{i-1}h_i'. \quad (7)$$

(4) Perform the addition

$$x_i = x_{i-1} + h_i''. \quad (8)$$

Substitution of Equation 6 into Equation 7 and the result into Equation 8 yields the following single step per digit:

$$x_i = x_{i-1} + \mathbf{Q}^{i-1}h_i \quad (9)$$

where

$$\mathbf{Q} = 1/2 \begin{pmatrix} 5 & -\sqrt{3} \\ \sqrt{3} & 5 \end{pmatrix}. \quad (10)$$

A computational simplification can be made by precomputing

the powers of **Q**. For n GBT digits, the absolute values of these powers can be stored in a 2 by n matrix **P**. The computation time required for the conversion of each address can be further reduced by premultiplying **Q** and h of Equation 9. This requires storage of a matrix **S** of size 2 by 6 by n. For example, for the conversion of 9 digit addresses, this matrix will contain 108 elements.

The GBT-to-Cartesian algorithm consist of two steps. In the preprocessing step the matrices **P**, **Q**, and **S** are computed, making use of the symmetry of the absolute values in **P**, **Q**, and **S**:

```
procedure gbt2xydata (r : real;
                      var S : Sarray);
const
  maxdgts = 9;
var
  h: array [1..2, 1..6] of real;
  P: array [1..2, 1..maxdgts] of real;
  Q: array [1..2] of real;
  i,k: integer;
  sqrt3, sqrt3d2: real;
begin
  sqrt3 := sqrt(3); sqrt3d2 := sqrt3/2;
  P[1,1] := 1; P[2,1] := 0;
  Q[1] := 2.5; Q[2] := -sqrt3d2;
  h[1,1] := 0; h[2,1] := sqrt3;
  h[1,2] := 1.5; h[2,2] := -sqrt3d2;
  h[1,3] := 1.5; h[2,3] := sqrt3d2;
  h[1,4] := -1.5; h[2,4] := -sqrt3d2;
  h[1,5] := -1.5; h[2,5] := sqrt3d2;
  h[1,6] := 0; h[2,6] := -sqrt3:
  for i := 2 to maxdgts do
    begin
      P[1,i] := Q[1] * P[1,i-1]-Q[2] * P[2,i-1];
      P[2,i] := Q[1] * P[2,i-1]+Q[2] * P[1,i-1];
    end;
for i := 1 to maxdgts do
  for k := 1 to 6 do
      begin
        S[1,k,i] := r * (P[1,i] * h[1,k] + P[2,i] * h[2,k]);
        S[2,k,i] := r * (P[1,i] * h[2,k] - P[2,i] * h[1,k]);
      end;
end;
```

The algorithm for the GBT-to-Cartesian conversion is then:

```
procedure gbt2xy (S : Sarray;
                  g : gbtarray;
                  var x,y : real);
const
  maxdgts = 9;
var
  i,k: integer;
begin
  x := 0; y := 0;
  for i := 1 to maxdgts do
    begin
      k := g[i];
      if (k <> 0) then
        begin
          x := x + S[1,k,i];
          y := y + S[2,k,i];
        end;
    end;
end;
```

This algorithm requires only n additions per GBT address on the average and performance is clearly of the order O(n). Conversion to Cartesian coordinates is therefore not as much of a stumbling block to the use of hexagons as is sometimes anticipated.

CARTESIAN-TO-GBT

The reverse process is somewhat more complicated. One reason is that it cannot be a direct conversion, but must be an approximation in which some accuracy will be lost. The main idea behind the method is to first convert to a coordinate system that is more in line with hexagonal tessellation, to interpolate on a grid based on this system, and then to convert the interpolated values to GBT addresses. We will again make use of the dual representation and repeat the process for each GBT digit.

At each aggregate level the aggregate centers can be connected to form a skewed grid as shown in Figure 6. The grids are again rotated relative to each other with the angle δ shown in Figure 5. Each grid point is assigned a relative GBT address appropriate to the aggregate level with 0 at the center of the grid. At the basic hexagon level this matrix is as follows:

$$\mathbf{K} = \begin{pmatrix} - & 34 & 36 & 21 & - \\ 16 & 1 & 3 & 25 & 20 \\ 53 & 5 & 0 & 2 & 24 \\ 50 & 52 & 4 & 6 & 61 \\ - & 56 & 41 & 43 & - \end{pmatrix} \tag{11}$$

For each aggregate i the addresses in **K** are multiplied by 10^{i-1}. In this way the relative GBT address for the distance-wise nearest aggregate center can be found on the grid, and can be added to an accumulative GBT sum which, after processing of the least significant digit, is the desired GBT address. The grid positions marked with a hyphen in the matrix **K** are never referenced.

The procedure for defining the Cartesian coordinates on the skewed grids is to shift the origin for the Cartesian system to the current GBT address, to rotate from $C_{i,1}$ to C_i, and then to convert C_i to the skewed coordinate system S_i.

The rotation from $C_{i,1}$ to C_i is exactly counter to the rotation expressed by $\mathbf{M}_{i,1}$ (Equation 5). The rotation matrix is, therefore, equal to

$$\mathbf{M}_{i,1}^{-1} = \mathbf{M}^{-1(i-1)} \tag{12}$$

where

$$\mathbf{M}^{-1} = 1/(2\sqrt{7}) \begin{pmatrix} 5 & \sqrt{3} \\ -\sqrt{3} & 5 \end{pmatrix} \tag{13}$$

The relation between C_1 and the skewed grid is shown in Figure 7.

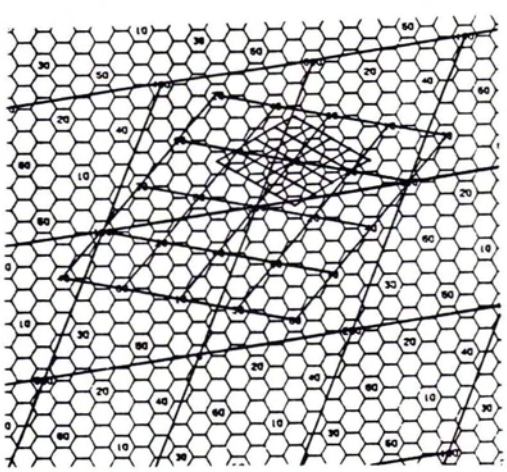

FIG. 6. Skewed grids connecting aggregate centers.

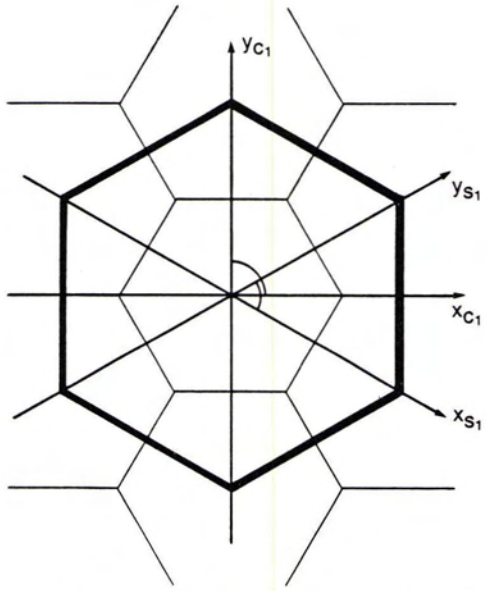

FIG. 7. Relation of dual hexagon and skewed coordinate systems.

The transformation from coordinates in C_1 to S_1 is accomplished through multiplication with a 2 by 2 matrix **W**:

$$\mathbf{W} = \begin{pmatrix} \sqrt{3}/3 & -1 \\ \sqrt{3}/3 & 1 \end{pmatrix}. \tag{14}$$

This matrix can derived by assuming the matrix elements to be unknown, and then to develop linear expressions for four pairs of known coordinates in both systems, from which the matrix elements can be solved. One more vital element in this process is the grid spacing of the points in the skewed grids, which is equal to the spacing between aggregate centers, and can be expressed as (see Equations 2 and 3):

$$s_i = r\sqrt{3}(\sqrt{7})^{i-1} \tag{15}$$

The algorithm can now be outlined as consisting of the following steps for each GBT digit at position i, starting with the most significant digit:

(1) Subtract the Cartesian coordinates xo of the current GBT address from the input Cartesian coordinates ($xo = 0$ for the most significant digit):

$$x_i = x - xo \tag{16}$$

(2) Rotate to C_i:

$$x_i = \mathbf{M}_{i,1}^{-1}\, x_i \tag{17}$$

(3) Transform to the skewed coordinate system:

$$x_i'' = \mathbf{W}x_i' \tag{18}$$

(4) Divide by the appropriate grid spacing and round to the nearest integers:

$$k_i = round(x_i''/s_i) \tag{19}$$

where k_i (1) and k_i (2) will range from -2 to $+2$

(5) Obtain the new GBT offset:

(20)

where the addition and the subtraction for the indices of the matrix **K** are performed to transform to the real storage locations for the elements of **K**.

(6) Add the GBT offset to the current GBT address, using GBT addition:

$$g_i = g_{i-1} + go_i \tag{21}$$

(where g_i refers to the currently accumulated GBT address for the i^{th} digit, and not to the i^{th} digit of the current GBT address).

(7) Convert the current GBT address to Cartesian coordinates xo, using the GBT-to-Cartesian algorithm described earlier.

An alternative to this last step is to convert k_i of Equation 19 back to coordinates on the skewed grid using the reverse of Equations 18 and 17. However, this involves multiplication, whereas the GBT-to-Cartesian conversion can be done with addition only.

The matrices of steps Equations 17 and 18 and the division of Equation 19 can be simplified as

$$\mathbf{U}_i = 1/r \begin{pmatrix} 1/3 & -1/\sqrt{3} \\ 1/3 & 1/\sqrt{3} \end{pmatrix} \begin{pmatrix} 5/14 & \sqrt{3}/14 \\ -\sqrt{3}/14 & 5/14 \end{pmatrix}^{i-1}. \tag{22}$$

Preprocessing consists of precomputing the values of **U**, which form a 2 by 2 by n matrix.

The Cartesian-to-GBT algorithm therefore also consist of two steps. The preprocessing step can be organized as follows:

```
procedure xy2gbtata (r: real;
                     var K : Karray;
                     var U : Uarray);
const
  maxdgts = 9;
var
  W: array [1..2] of real;
  T: array [1..2,1..maxdgts] of real;
  S: array [1..2] of real;
  i: integer;
begin
  T[1,1] := 1; T[2,1] := 0;
  S[1] := 5/14; S[2] := sqrt(3)/14;
  W[1] := 1/3; W[2] := -1/sqrt(3);
  for i := 2 to maxdgts do
    begin
      T[1,i] := S[1] * T[1,i-1] - S[2] * T[2,i-1];
      T[2,i] := S[2] * T[1,i-1] + S[1] * T[2,i-1];
    end;
  for i := 1 to maxdgts do
    begin
      U[1,1,i] := (W[1] * T[1,i] - W[2] * T[2,i])/r;
      U[1,2,i] := (W[1] * T[2,i] + W[2] * T[1,i])/r;
      U[2,1,i] := (W[1] * T[1,i] + W[2] * T[2,i])/r;
      U[2,2,i] := (W[1] * T[2,i] - W[2] * T[1,i])/r;
    end;
  K[2,1,1] := 3;K[3,1,1] := 3;
                  ⋮
  K[3,5,2] := 1; K[4,5,2] := 3;
end;
```

Note that the values of **K** are also assigned in this step, and are not all listed. The algorithm for the Cartesian-to-GBT address conversion is

```
procedure xy2gbt (U : Uarray;
                  K : Karray;
                  S : Sarray;
                  A : addtab;
                  var x,y, real;
                  var g : gbtarray);
const
  maxdgts = 9;
var
```

```
  xg, yg: real;
  xs, ys: real;
  go, gs: gbtarray;
  i,j,k1,k2 : integer;
begin
  xg := 0; yg := 0;
  for i := 1 to maxdgts do
    g[i] := 0;
  for i := maxdgts - 1 downto 1 do
    begin
      xs := x - xg;
      ys := y - yg;
      k1 := 3 + round(U[1,1,i] * xs + U[1,2,i] * ys);
      k2 := 3 - round(U[2,1,i] * xs + U[2,2,i] * ys);
      for j := 1 to maxdgts do
        go[j] := 0;
      go[i] := K[k1,k2,2];
      go[i+1] := K[k1,k2,1];
      gbtadd(A,g,go,g);
      if i <> 1 then
        gbt2xy(S,g,xg,yg);
    end;
end;
```

This algorithm requires far more computation time than the GBT-to-Cartesian conversion because, for each digit, a GBT addition and conversion to Cartesian is required. The performance is therefore of the order of $O(n^2)$. The algorithm will work correctly for $n = maxdgts - 1$ digits. The reason is that in the conversion process temporary GBT addresses of *maxdgts* digits may be generated. The algorithm has been extensively tested for addresses up to eight digits.

CONCLUSION

We have presented algorithms to convert Cartesian coordinates from and to GBT addresses. The GBT-to-Cartesian conversion is an order of magnitude faster than the reverse process. This fact can be exploited when resampling a square raster image to a hexagonal representation by driving the resampling from GBT space, rather than from Cartesian coordinate space. Depending on the resampling technique and hexagonal resolution, this may mean that certain raster cells may never be referenced. For these cells, the reverse process can be invoked, in order to force a GBT representation.

The problem of errors resulting from the conversion has not been addressed in this paper, and could be a topic for an another analysis. No resolution is lost due to the difference in spatial addressing systems when converting from GBT to Cartesian, other than as a result from the computational process, but in the other direction, the selected size of the hexagon will determine the loss of resolution. However, such considerations depend on the application of the GBT address, which may be to organize data into sept trees, or to serve as a replacement for the Cartesian coordinates.

There are definite advantages that can be obtained from the hexagonal tessellation, such as the greater angular resolution and the absence of four-way and eight-way connectedness problems associated with square rasters. We hope that efficient algorithms to convert from and to GBT space will promote the use of these properties.

ACKNOWLEDGMENTS

The author would like to thank Stuart W. Doescher of the EROS Data Center for his efforts in converting an early version of this paper to Latex, and he would like to thank the reviewers for their valuable comments. The author is indebted to Dennis L. Murphy for further stimulating his interest in hexagons. The work reported upon herein was performed under U. S. Geological Survey contract 14-08-0001-22521.

REFERENCES

Ahuya, N., 1983. On Approaches to Polygonal Decomposition for Hierarchical Image Representation. *Computer Vision, Graphics, and Image Processing*, Vol. 24, No. 2, pp. 200–214.

Crawley, B.G., 1975. Automatic Contouring on the Gestalt Photomapper, Testing and Evaluation. *Proceedings of Orthophoto workshop III*, American Society of Photogrammetry, San Antonio, Texas, 1–11.

Crettez, J., 1980. A Pseudo-Cosine Transform for Hexagonal Tessellation with an Heptarchical Organization. *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*, IEEE Computer Society, pp. 192–194.

Gibson, L., and D. Lucas, 1982. Spatial Data Processing Using Generalized Balanced Ternary. *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*, IEEE Computer Society, pp. 566–571.

———, n.d.. *Line and Edge Extraction Using Hierarchical Methods*. Report for Contract F49620-81-L-00039, U.S. Air Force Office of Scientific Research.

Knuth, D.E., 1969. *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*. Addison Wesley Publishing Company, pp. 173–176.

Lucas, D., and L. Gibson, 1970. *A System for Hierarchical Addressing in Euclidean Space*, Unpublished Document, pp. 1–11.

Mandelbrot, B.B., 1982. *The Fractal Geometry of Nature, Chapter 6, "Snowflakes and other Koch curves."* W.H. Freeman and Company, San Francisco, pp. 46–47.

Samet, H., 1984. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys*, Vol. 16, No. 2, pp. 186–260.

van Roessel, J., P.G. Langley, and T.D. Smith, 1978. TIMBER-PAK—A Second Generation Forest Management Information System, Integrated Inventories of Renewable Natural Resources: *Proceedings of the Workshop*, Rocky Mountain Forest and Range Exp. Station General Technical Report RM-55, pp. 360–374.