Efficient algorithm for polygon overlay for dense map image data sets

S Sitharama Iyengar and Stephan W Miller*

A map consists of a finite set of mutually disjoint polygons. A data structure and efficient algorithm for polygon overlay of dense map image data sets are discussed. The data structure, referred to as the ending-x structure, is a variant of the y-partition structure developed by Merrill which is still widely acclaimed as an optimum raster data structure. The modest refinement of the ending-x structure further reduces raster data volume. More importantly, it can significantly enhance the process of polygon overlay for multiple map image data sets. Considering the density of many map image data sets, and the significance of the polygon overlay problem to spatial data handling, these are significant advances. The paper describes the ending-x data structure and compares the algorithm for polygon overlay suggested by Merrill with one employing the ending-x structure.

Keywords: mapping, image data sets, algorithms

Applications of spatial data, such as land resource management, energy development and environmental impact assessment, today frequently employ computerized systems. These systems develop spatial data from map image data sets, i.e. the data are developed from topographic maps, thematic map overlays and digital imagery captured by satellites or other airborne sensors. Increasingly there is a greater demand for higher resolution spatial data and coverage of more extensive regions. These factors work together to produce what can be termed 'dense map image data sets' (Figure 1).

High-resolution data can be obtained from larger-scaled maps or by data capture devices with higher resolving power than their predecessors. As an example, topographic maps available in the USA are being converted from graphic to digital form at the largest scale uniformly available[1]. This includes contour lines and other features from some 57 000 maps at a scale of 1:24 000. Under procedures used by the US Geological Survey's National Mapping Division, the contours are often digitized in vector format and then converted to a raster format representing a point sample of the surface elevation. Point values are interpolated from the digitized data, typically at a 30 m interval[2]. For an average map in the 1:24 000 scale map series, this sampling density results in approximately 180 000 points.

Also, imagery is being collected by earth resource satellites at higher resolution. The Landsat-4 satellite, launched in July 1982, has a pixel size of 30 m square[3]. This represents nearly a five-fold increase in resolution and data volume over the Landsat-3 satellite which has a pixel size of 79 m by 56 m. Planned systems, such as the French Spot satellite, promise even higher resolution[4].

The growing number of dense data sets means that more efficient data structures and algorithms are needed for map image data. In particular, the efficiency of the polygon overlay operation is important to any spatial data handling system.

As an example, an analysis of water quality in a river basin may involve identifying areas with high potential for soil erosion. Operationally, this may include separate map image data sets. Polygonal regions from land cover, soils and slope zone map data sets must be compared to determine their coincidence. Composite regions with minimal vegetative cover, slope zones with steep slopes and soil regions with poor cohesive qualities are those which would be expected to have the greatest potential for erosion. The process of updating map image data, for example to show land cover changes where clear-cutting of forest has occurred, also requires efficient map overlay. In this case, revised boundaries and codes are compared with existing data. Likewise, efficient methods of spatial search and retrieval are often those which generate a mask image of the zone to be retrieved and then merge this data set with others. For example, forest applications may make it necessary to identify clear-cut areas existing within 1 mile of a stream. Techniques have been developed to generate the corridor zone about

Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803, USA
*Present address: Infragraphics Corp., Huntsville, AL, USA

0262–8856/86/03167–08 $03.00

linear features such as streams[5]. The successful completion of a corridor search, however, requires an efficient polygon overlay method.

Map data can be represented as points, lines and areas. In a spatial database, aerial data sets are the most voluminous[6]. A means of structuring aerial data, where the graphic data consists of labelled, polygonal regions, has been developed and applied in several past research efforts[7]. Such data are captured in vector format and converted to a high-resolution raster format[8]. A modified form of run-length encoding has been employed to reduce the volume of raster data[5]. More exactly, this structure is similar to the y-partition data structure suggested by Merrill[9]. It anticipated the modification to the y-partition structure suggested by Haralick: namely, that the y-partition be reordered into scan-line major order[10]. When the data for all polygonal regions on a map sheet are reordered by scan line, it becomes apparent that beginning-x and ending-x for adjacent regions are not necessary, i.e. that successive ending-x values will suffice to describe the extent of any single region. This modest refinement to the y-partition structure can significantly improve the performance of the polygon overlay procedure over other methods. In particular, it improves the algorithm devised by Merrill which still enjoys widespread support[10–12].

ALTERNATIVE METHODS OF POLYGON OVERLAY

The polygon overlay problem has generated interest in many areas of computer graphics. The task is not only central to spatial data handling, but also to such widely used procedures as hidden line removal for three-dimensional applications. The process of polygon overlay can, of course, be carried out on data in either a vector or raster format.

Efficient algorithms for polygon overlay have been developed for vector representations of data[13,14]. Yet the matter of efficiency for dense map image data format is highly debatable in light of the influence which data volume and complexity have on the algorithms. The volume and complexity of dense map image data sets are such that algorithms for polygon overlay with vector formats are less efficient than any of several alternatives employing raster format[11,15].

POLYGON OVERLAY FOR VECTOR FORMAT

For vector representations of data, computing the intersections between overlapping regions can be quite involved. A simple algorithm employing closed polygon boundaries and point-in-polygon testing for two overlapping regions (Figure 2) is as follows.

1. Determine that the bounding x, y minimum and maximum of polygon regions R_k and R_l overlap.
2. Determine that the point (x_{i+1}, y_{i+1}) from R_k lies within R_l.
3. Compute the intersection of segments (x_i, y_i), (x_{i+1}, y_{i+1}) and (x_m, y_m), (x_{m+1}, y_{m+1}).
4. Determine that the point (x_j, y_j) is the first consecutive point from R_k outside R_l.
5. Repeat step 3 for segments (x_j, y_j), (x_{j+1}, y_{j+1}) and (x_n, y_n), (x_{n-1}, y_{n-1}).
6. Points from both polygons, between the intersection points inclusive, form the region of intersection. Insert and delete points as needed to adjust the boundaries of R_k and R_l and to form the bounds of R_m.

More efficient algorithms for polygon overlay have been devised[13,16]. In particular, more recent algorithms take advantage of topological data structures which are increasingly used for map image data[7,17]. In this manner,

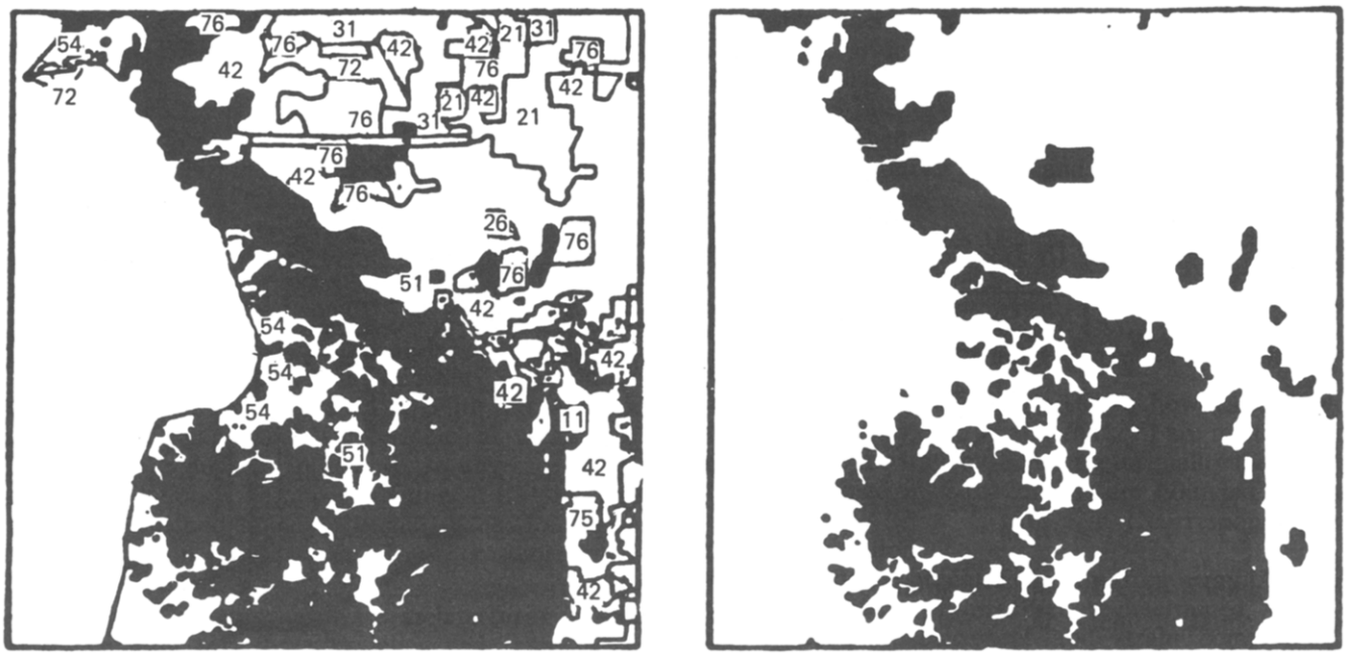

Figure 1. Source map data and a corresponding dense map image data set

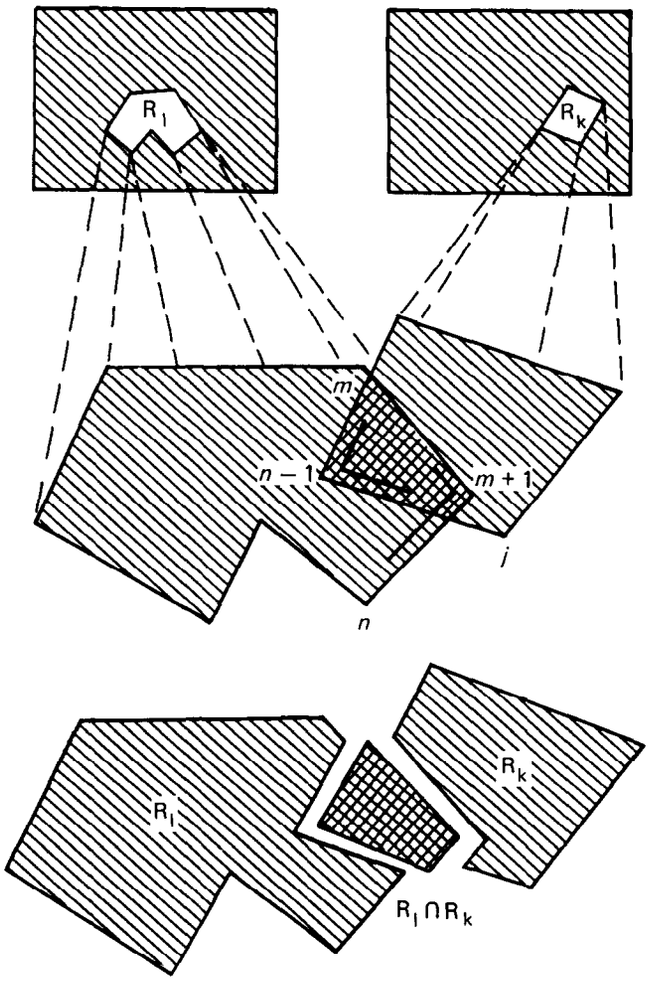

Figure 2. Polygon overlay for closed polygons

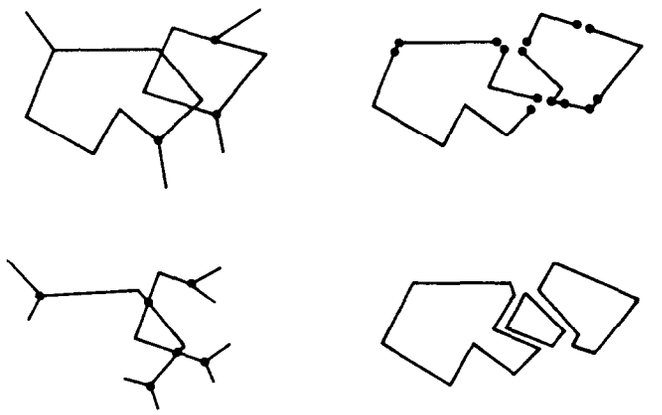

Figure 3. Polygon overlay for arc data

Table 1. Representative data volumes for dense map image data sets

Item	Land use	County	Census	Watersheds
Arcs	7 340	42	419	36
Coordinates	165 384	3 526	10 283	4 850
Vertices per arc	23	84	66	135

Adapted from Reference 7, p 4. Numbers represent average figures based upon a sample of approximately 50 map sheets. One data set was prepared for each of the above categories

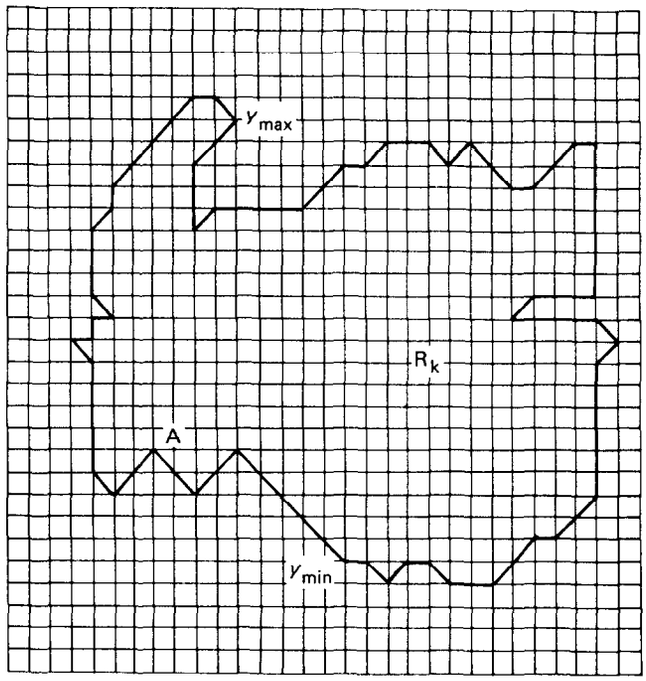

Figure 4. y-partition of region k

the problem can be reduced to one of finding intersections between overlapping arcs or chains, i.e. boundaries between two contiguous regions in a map image data set[13]. Ultimately, the arcs resulting from this process can be rechained to form new closed boundaries for the regions of intersection (see Figure 3). Table 1 summarizes the volume and complexity of arc data for a nationwide resource inventory and can be used to estimate the complexity of polygon overlay for arc data formats.

Most efficient algorithms for polygon overlay involve presorting of the data[14]. It quickly becomes apparent why this is necessary. Where m and n are the number of arcs in two data sets being compared, a simple algorithm for overlay would involve $(2mn)$ comparisons on average to identify arcs whose bounding x, y coordinates overlap. For every overlap detected, $(2kl)$ comparisons are needed to isolate overlapping segments. Here, k and l represent the average number of vertices for an arc in the respective data sets.

The interaction effect between the two data sets makes it difficult to determine exactly how many overlaps to anticipate. In the best case, one of the two data sets will be decidedly less dense. Even under these conditions, the bounding x, y coordinates of arcs from the more dense data set can be expected to overlap with two arcs from the other. Thus, the total number of comparisons needed to isolate overlapping line segments is $(mn)*(k2l)2$.

For an intersection of land use by county, basically a case involving a dense, convoluted data set and a sparse, regular one, a total of [(42)(7340)(2)(23)(2)(84)(2)] or 4.765×10^9 comparisons can be expected. Of course, this does not complete the overlay task. It simply avoids the calculations for line intersection where segments do not overlap at all. For a further discussion of more efficient algorithms, see Reference 14.

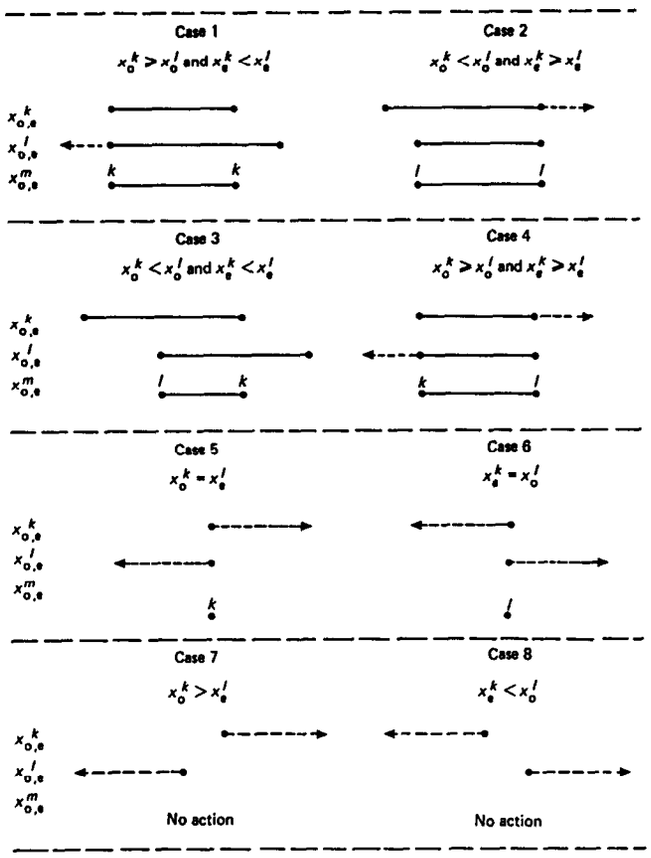

Figure 5. Cases for resolution of overlap between two odd–even intervals

POLYGON OVERLAY FOR RASTER FORMAT

In his original paper, Merrill described an efficient algorithm for polygon overlay for the y-partition data structure[9]. For discussion purposes, this structure and the algorithm are briefly reviewed.

y-partition data structure and algorithm

The y-partition structure is considered a variant of run-length encoding. The basic entity was considered by Merrill to be a closed region whose bounding points are located as discrete points in a rectangular grid (see Figure 4). Any such region can be decomposed into a series of related horizontal segments. In Figure 4 the region R_k is decomposed into n_k segments or intervals of x in the range y_{min} to y_{max}. For each discrete value of y, denoted as y_p, an even number of x-values define the extent of R_k along y_p. Successive odd–even pairs define the beginning and end of R_k. This can be denoted as (k, y_p, x_o, x_e).

Described in the context of the y-partition data structure, the polygon overlay problem requires the resolution of overlap between all ordered pairs for two overlapping regions, R_k and R_l. The process creates three regions: adjusted versions of R_k and R_l, and the new region R_m.

Using Merrill's notation, the process of polygon overlay involves a series of checks to determine the exact relationship between two odd–even pairs suspected of overlapping (see Figure 5). Prior to this, the two candidate regions, k and l, must be identified by comparing their y_{min} and y_{max} bounds. In his discussion of the y-partition structure, Haralick suggested reordering the data so that intervals for all data sets are sorted into y_p-major order, then monotonically increasing by x-coordinates[10]. He referred to data reordered in this manner as 'interval lists'. This refinement facilitates the process of polygon overlay for complete map image data sets which proceeds most efficiently on a scan-line-by-scan-line basis. It avoids checking for overlap in the y-dimension and allows the process to focus on resolving overlap in the x-dimension.

The algorithm for polygon overlay can be implemented as a series of hierarchical checks on the odd and even intervals suspected of overlapping (see Figure 5). The number of possible cases of overlap is simply the result of the number of combinations of two odd–even pairs: $2^{(4-1)}$ or 8. A straightforward implementation of the information in Figure 5 in FORTRAN-77 is given in Appendix 1.

Ending-x data structure and algorithm

The algorithm described in Appendix 1 is a direct translation of Merrill's original. When the refinements of Haralick are incorporated, however, it becomes apparent that further improvements can be made in the data structure which improve the overlay process.

A cursory examination of the interval lists proposed by Haralick reveals the redundancy evident in the data when it is rearranged into the y_p-major form. For any two successive entries in the list, the ending-x of the first must be one less than the beginning-x of the second, or the topology of the data set can be questioned. This pattern can be repeated for any map data set, for any values of y_p in a data set and for all x-values in the range of x_{min} to x_{max} for an entire data set (see Figure 6).

By taking advantage of the 'region coherence' of data sets ordered as interval lists, all beginning-x coordinate values can be deleted from the data, thus saving space. The remaining x-value, the ending-x coordinate and the region index are sufficient for manipulation and comparison of two or more map image data sets. At the expense of an explicit definition of the x-interval, which

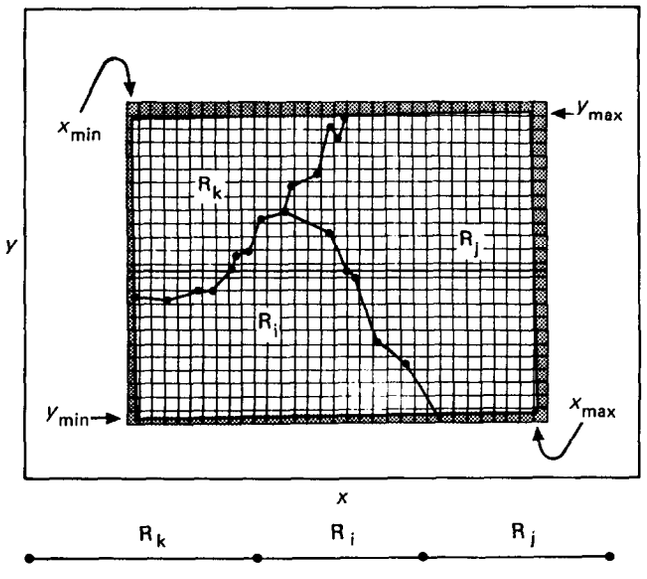

Figure 6. Region coherence of interval lists

is a minor loss considering the ease with which it can be restored, a significant improvement in the overlay process is possible.

A major limitation of the comparison of interval lists is that two lists cannot be compared effectively in a single pass. This is the consequence of having to check both beginning and end coordinates explicitly and the number of combinations which can arise when three or more data sets are involved. While the interval list method can be used recursively or even improved in light of the ordering of x-values along y_p, a much simpler algorithm is possible when only ending-x values need be compared.

EXAMPLE OF RETRIEVAL AND ANALYSIS INVOLVING COMPARISON

The first step in conducting higher-level retrieval and analysis operations for spatial data involves a comparison of the data sets involved. At least two map image data sets must be compared for this process. In past applications involving the ending-x structure, data has been captured in vector format and then converted to a grid system. In essence, the original x, y coordinate data is resampled to a coarser resolution. Typically, map image data has been resampled by from five to ten of the original coordinate units. The output coordinate data is then arranged as described previously for the y-partition and ending-x structures.

To illustrate the overlay process for the ending-x structure, a section of detailed land use and detailed soil data are shown in Figure 7. The original map overlays are shown in the top portion of the figure. Digitized representations of this data are presumed to approximate the original data precisely. In the middle portion of the figure, the raster approximation of the original polygon shapes can be seen more clearly. The ending-x values of the regions along the first scan line are shown in the bottom portion of the figure. The horizontal intervals have been 'rounded' spatially to the nearest whole column for the resample grid. In the example, a fairly coarse resample grid has been used for illustration purposes.

In Tables 2 and 3, the values for land use and soils along the first scan line in Figure 6 are depicted as lists. The overlay process actually compares successive ending-x values in each of the two lists searching for the current minimum out of two candidates. In the first comparison, region 1 from each of the overlays is compared. In this case, the first region for land use has a value of '113' and the horizontal segment extends from an ending-x value of 1 (which is implied) to an ending-x value of 6. For soils, the code value is 'BR' and it extends from an ending-x of 1 to an ending-x of 8. Obviously, the minimum x is 6 and so the first entry in the composite list (Table 4) reflects the composite values, i.e. '113-BR', and the minimum ending-column.

For the next comparison, region 2 from land use is compared to region 1 of the soils. Note that for the data set having the minimum x-coordinate value it is necessary to advance to the next member in the list before the next comparison. In this iteration, an ending-x position of 8 is the lesser of the two and so the composite attributes of '1231-BR' are entered in the list shown in Table 4. This process continues until both lists have been exhausted. The lists are then filled with data from the next scan line and this continues until all intervals for both data sets have been processed.

Table 2. List of ending-x positions for scan line 1 for land use and land cover

Region (R)	Attribute value	Ending-x
1	113	6
2	1231	12
3	111	17
4	113	22
5	1231	24
6	112	43
7	1231	48
8	112	60

CONCLUSIONS

The ending-x structure offers advantages in terms of reducing the data volume of raster data sets. More importantly, it offers an improvement in polygon overlay for multiple map image data sets. This type of operation is significant for map data handling because of the increasing resolution of raster elements, the size of geographic areas being covered at high resolution, and the degree of responsiveness required from map data

Table 3. List of ending-x positions for scan line 1 for detailed soils data

Region (R)	Attribute value	Ending-column
1	BR	8
2	GfC2	16
3	Fa	26
4	GfC2	29
5	Ch2	31
6	GfB	33
7	Fa	39
8	A1C	49
9	GfC2	55
10	GfB	57
11	GfC2	60

Table 4. List of ending-x positions along scan line 1 for composite of land use and soils

Regions	Attribute values	Ending-x
1–1	113–BR	6
2–1	1231–BR	8
2–2	1231–GfC2	12
3–2	111–GfC2	16
3–3	111–Fa	17
4–3	113–Fa	22
5–3	1231–Fa	24
6–3	112–Fa	26
6–4	112–GfC2	29
6–5	112–Ch2	31
6–6	112–GfB	33
6–7	112–Fa	39
6–8	112–A1C	43
7–8	1231–A1C	48
8–8	112–A1C	49
8–9	112–GfC2	55
8–10	112–GfB	57
8–11	112–GfC2	60

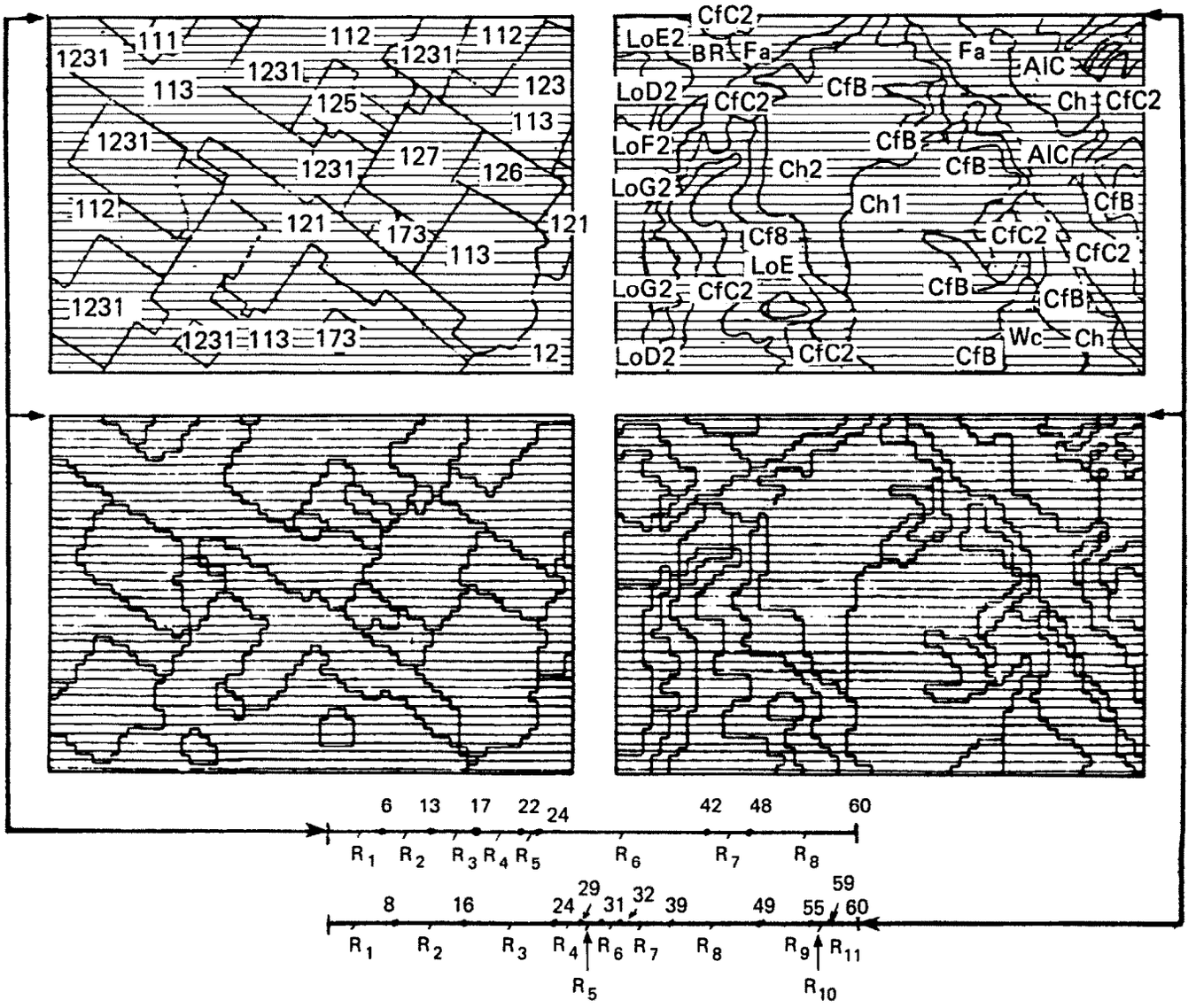

Figure 7. Deriving composite data for ending-x raster format data

handling systems. The modified y-partition data structure and an algorithm for computing intersections between two or more coterminous data sets have been presented. These improvements allow efficient comparison and retrieval operations which will allow larger volumes of spatial data to be processed. This should at least maintain or perhaps even improve the responsiveness of map data handling systems.

REFERENCES

1 **Southard, R B** 'A geological survey perspective on digital cartography' *Proc. Auto-Carto IV Symp.* (1980)

2 **Hallam, C** 'Obtaining maps and data from the US Geological Survey' *Ann. Conf. Urban and Regional Information Systems Association* (1981)

3 **US Geological Survey** *EROS Data Center Landsat Data Users Notes, 23* (1983)

4 **Begni, G** 'Selection of the optimal spectral bands for the SPOT satellite' *Photogrammetric Eng. Remote Sensing* Vol 48 No 10 (1982)

5 **Miller, S W** 'Compressing satellite imagery for geographic information systems applications over extensive regions' *Proc. Pecora VII Symp.* (1982)

6 **Tomlinson, R F and Boyle, A R** 'The state of development of systems for handling natural resources inventory data' *Cartographica* Vol 18 No 4 (1981)

7 **Mitchell, W, Anderson, E, Guptill, S, Fegeas, R and Hallam, C** 'GIRAS: Geographic Information Retrieval and Analysis System' *US Geological Survey Professional Paper 1059* (1977)

8 **Guptill, S C** 'The impact of computer graphics, data manipulation, software, and computer equipment on spatial data structures' *1st Int. Advanced Symp. Topological Data Structures* (1978)

9 **Merril, R D** 'Representation of contours and regions for efficient computer search' *Comm. ACM* Vol 16 No 2 (1973)

10 **Haralick, R M** 'A spatial data structure for geographic information systems' in **Freeman, H (ed.)** *Map data processing* (1980)

11 **Claire, R W and Guptill, S C** 'Spatial operators for selected data structures' *Proc. Auto-Carto V Symp.* (1982)

12 **Peuquet, D J** 'Raster processing: an alternative

approach to automated cartographic data handling' *American Cartographer* (1979)

13 **Freeman, H** 'Analysis and manipulation of lineal mapped data processing' in **Freeman H (ed.)** *Map data processing* (1980)

14 **Pavlidis, T** *Algorithms for graphics and image processing* (1982)

15 **Zobrist, A L** 'Data structures and algorithms for raster data processing' *Proc. Auto-Carto IV Symp.* (1980)

16 **White, D** 'A design for polygon overlay' *2nd Int. Advanced Symp. Topological Data Structures, Harvard University* (1979)

17 **Peucker, T K and Chrisman, N** 'Cartographic data structures' *American Cartographer* Vol 2 No 1 (1975)

BIBLIOGRAPHY

Miller, S and Iyengar, S S 'Representation of regions of map data for efficient comparisons and retrieval' *Proc. IEEE CVPR Conf.* (May 1983)

APPENDIX 1: ALGORITHM FOR POLYGON OVERLAY USING INTERVAL LISTS

POLYGON OVERLAY FOR RASTER DATA

MERRILL'S ALGORITHM: EIGHT CASES OF POSSIBLE OVERLAP BETWEEN TWO INTERVAL LISTS ALONG SCAN-LINE AT YP

ARGUMENT LIST:

XODD — A 2-D ARRAY CONTAINING ORDERED BEGINNING X-COORDINATES FOR TWO LISTS

XEVEN — AS 'XODD' FOR ENDING X-COORDINATES

I, J — POINTERS TO THE CURRENT INTERVAL IN EACH OF THE RESPECTIVE LISTS. I.E. XODD(1,I) CONTAINS THE BEGINNING X-COORDINATE FOR THE FIRST DATA SET, XODD(2,J) THE BEGINNING X-COORDINATE FOR THE SECOND DATA SET

RINDX — A SCALAR ARRAY CONTAINING REGION CODES FOR THE INTERVALS POINTED TO BY I AND J

XBEG,
XEND — ODD, EVEN X-COORDINATES IDENTIFYING THE COMPOSITE REGION FORMED BY K,L INTERSECTION

```
SUBROUTINE RESOLV (XODD,XEVEN,
I,L,KR,LR)
REAL XODD(2,1000), XEVEN(2,1000)
INTEGER RINDX(2,1000)

CASES 1 AND 4 (REFER TO FIGURE 5)

IF (XODD(1,I) .GE. XODD(1,J)) THEN
   XBEG = XODD(K)

   CASE 4
  (IF (XEVEN(1,I) .GE. XEVEN(2,J)) THEN
        XEND = XEVEN(2,J)
        CALL REPORT(XBEG,XEND,RINDX,I,J)
        RETURN
   ELSE
   CASE 1
         XEND = XEVEN(1,I)
        CALL REPORT(XBEG,XEND,RINDX,I,J)
        RETURN
ENDIF

CASES 2 AND 3

IF (XODD(1,I) .LT. XODD(2,J)) THEN
   XBEG = XODD(2,J)
   CASE 3
   IF (XEVEN(1,I) .LT. XEVEN(2,J)) THEN
        XEND = XEVEN(1,I)
        CALL REPORT(XBEG,XEND,RINDX,I,J)
        RETURN
   ELSE
   CASE 2
         XEND = XEVEN(2,J)
         CALL REPORT(XBERG,XEND,
         RINDX,I,J)
         RETURN
ENDIF

CASE 5

IF (XODD(1,I) .EQ. XEVEN(2,J)) THEN
   XBEG = XODD(1,I)
   XEND = XODD(1,I)
   CALL REPORT(XBEG,XEND,RINDX,I,J)
   RETURN
ELSE

CASE 6

IF (XEVEN(1,I) .EQ. XODD(2,J)) THEN
   XBEG = XODD(2,J)
   XEND = XODD(2,J)
   CALL REPORT(XBEG,XEND,RINDX,I,J)
   RETURN
ELSE

CASES 7, 8 (IMPOSSIBLE IF LISTS ARE
MONOTONIC AND TOPOLOGY IS CORRECT)

IF ((XODD(1,I) .GT. XEVEN(2,J)) .OR.
   (XEVEN(1,I) .LT. XODD(2,J)) THEN
   XBEG = 0.0
   XEND = 0.0
   RETURN
ELSE
RETURN
END
```

APPENDIX 2: SIMPLIFIED ENDING-X ALGORITHM FOR POLYGON OVERLAY

POLYGON OVERLAY FOR RASTER FORMAT
ENDING-X ALGORITHM
RESOLVES OVERLAP BETWEEN MULTIPLE LISTS OF ENDING-X AT YP

ARGUMENT LIST:

ENDX —A 2-D ARRAY CONTAINING ENDING-X VALUES FOR ALL DATA SETS FOR THE CURRENT SCAN LINE AT YP

RINDX —A 2-D ARRAY WHICH HOLDS REGION INDEX VALUES CORRESPONDING TO THE LOCATIONS IN ENDX

NDS —THE NUMBER OF DATA SETS TO BE COMPARED(20 MAX)

IPTR —A SCALAR ARRAY CONTAINING POINTER VALUES FOR ENDX AND RINDX ARRAYS FOR EACH DATA SET 1 TO NDS

XMAX —THE MAXIMUM X-VALUE FOR ALL DATA SETS. ALL DATA SETS ARE PRESUMED TO FILL THE INTERVAL XMIN TO XMAXC

OTHER VARIABLES:

CINDX —A SCALAR ARRAY CONTAINING THE RINDX VALUES AT THE CURRENT ITERATION. THESE REGION INDEX VALUES ARE ASSOCIATED WITH THE MINIMUM X VALUE FOR THE ITERATION

XCMIN —THE MINIMUM OF THE 'NDS' CANDIDATES IN THE ENDX ARRAY

```
SUBROUTINE RESOLV (ENDX,RINDX,NDS,
IPTR,XMAX)
REAL ENDX(NDS,1000), RINDX(NDS,1000),
IPTR(20), CINDX(20)

LOOP UNTIL THE CURRENT MINIMUM IS
EQUAL TO THE OVERALL MAX

DOWHILE XCMIN < XMAX
      INITIALIZE POINTERS TO 1 FOR ALL
      DATA SETS
      DO 100 M = 1,NDS
            IPTR(M) = 1
100   CONTINUE

      USE FUNCTION FINDMINX TO FIND
      SMALLEST X-VALUE

      XCMIN = FINDMINX (ENDX,NDS)

      REPORT VALUES FOR XCMIN AND ALL
      CURRENT RINDX VALUES

      CALL REPORT(XCMIN,RINDX,
      NDS,IPTR)

      UPDATE POINTERS AS NEEDED

      DO 200 M = 1,NDS
            IF(ENDX(M,IPTR(M)) .EQ. XCMIN)
            THEN
                  IPTR(M) = IPTR(M) + 1
            ELSE
200         CONTINUE
      ENDWHILE
      RETURN
      END
```