Spatially ordered networks and topographic reconstructions

CHRISTOPHER GOLD
College of Geographic Sciences, Lawrencetown,
Nova Scotia, Canada BOS 1MO

and SEAN CORMACK
Department of Computing Science, University of Alberta,
Edmonton, Canada

Abstract. A technique is discussed for obtaining a contour tree efficiently as a by-product of an operational contouring system. This tree may then be used to obtain contour symbolism or interval statistics as well as for further geomorphological study. Alternatively, the tree may be obtained without the computational expense of detailed contour interpolation. The contouring system proceeds by assuming a Voronoi neighbourhood or domain about each data point and generating a dual-graph Delaunay triangulation accordingly. Since a triangulation may be traversed in a tree order, individual triangles may be processed in a guaranteed top-to-bottom sequence on the map. At the active edge of the map under construction a linked list is maintained of the contour ‘stubs’ available to be updated by the next triangle processed. Any new contour segment may extend an existing stub, open two new stubs or close (connect) two previous stubs. Extending this list of edge links backwards into the existing map permits storage of contour segments within main memory until a dump (either to plotter or disc) is required by memory overflow, contour closure, contour labelling or job completion. Maintenance of an appropriate status link permits the immediate distinction of local closure (where the newly-connected segments are themselves not connected) from global closure (where a contour loop is completed and no longer required in memory). The resulting contour map may be represented as a tree, the root node being the bounding contour of the map. The nature of the triangle-ordering procedure ensures that inner contours are closed before enclosing ones, and hence a preliminary contour tree may be generated as conventional contour generation occurs. A final scan through the resulting tree eliminates any inconsistencies.

1. Introduction

In designing a computer-based contouring package, one of the more irritating jobs (other than contour labelling perhaps) is the identification of closed-depression contours in order to specify a special line symbolism. Indeed, in many systems such features are ignored owing to the high developmental, computational and storage costs associated with reprocessing output line segments that should, by rights, have already been delivered to the plotting system and subsequently forgotten.

In the past few years we have been actively involved in the development of high-quality commercial software for contouring (Gold 1978, Contoursoft 1985). Since our objectives included uncompromising data fidelity for any possible distribution of data, a variety of original or unconventional techniques was employed. Gold (1984) describes the design issues involved, which were particularly critical to the market (the oil industry and similar geological applications) for which the software was originally
intended. It was, however, only relatively recently that other potential markets (including topographic and hydrographic mapping) made us re-evaluate issues involved in the display of output, including the flagging of depression contours and other topics involving knowledge of the relationships of adjacent contour strings: in other words, the derivation of contour trees. Concepts of contour trees have been developed by Morse (1969), Mark (1977) and recently by Roubal and Poiker (1985).

2. Basic steps in the contouring procedure

Once this decision was made, a review of our current techniques indicated that most of the tools were already in place. A moral, in fact, emerged. In our work with high-fidelity topographic reconstruction from arbitrary (even awkward) distributions of data such as air, ship, or seismic traverses, we were constrained to preserve the maximum topological information in order to provide definitions of adjacency that plausibly represented the data (as opposed to the ‘pick the nearest two data points in each quadrant’ school of thought). Continuing this approach allowed us to generate the contour strings themselves in a predictable order. Review of the data structures required to maintain the contour strings as a local search process rather than as a global sorting procedure showed that the contour tree has been implicit all along. The moral: it pays to preserve topology.

From the earliest version it was recognized that the most effective way of expressing the neighbourhood relationships between data points was via an automatically-generated triangulated irregular network (or TIN) (Gold et al. 1977, Gold 1978). However, the use of polynomial patches within each triangle as a method of interpolation led to a preoccupation with the shape of the triangle, and hence a limited triangulation algorithm. Later, the stable Delaunay triangulation was implemented, with its underlying dual Thiessen or Voronoi criterion (figure 1). The triangulation algorithm itself has remained essentially unchanged. It starts with a boundary framework, data points are inserted one at a time into the triangulation and the newly generated or modified triangle is optimized by an examination of adjacent pairs of triangles. If the additional fourth vertex falls within the circumscribed circle of the original triangle the quadrilateral formed by the four vertices must be subdivided in the alternative fashion: northwest to southeast, for example, instead of northeast to southwest (see details in Gold 1978).

The final triangulation provides the framework for a reasonable selection of neighbouring points at any location within the map area. The interpolation procedure itself is of the weighted-average variety, the earlier polynomial patch procedure providing an insufficient degree of surface continuity across the boundaries of the triangular patch. Any appropriate weighting function, however, must have one basic property: the weighting value attached to each data point must have decreased to zero by the time that point ceases to be part of the neighbourhood set used for local interpolation (Gold 1984). This condition is violated in most available contouring programs, either internally, or else indirectly by permitting the user to specify independently the weighting function and some combination of the number of points to be averaged and/or the appropriate search radius. In our case zero point-weighting at the edge of the neighbourhood set is achieved by using a weighting function derived from the triangulation itself, and in addition using the triangulation to select neighbouring data points.
3. Spatially ordered triangular networks and contour generation

Once a satisfactory interpolation function has been obtained, the actual procedure for contour generation must be developed. Ours was based on three considerations.

(1) Any procedure for grid generation is counter-productive since it prohibits, in the general case, the preservation of surface elevation values at every data point as well as leaving unresolved the saddle-point problem (since four elevations do not usually form a plane).

(2) Regular subdivision of each previously-described triangle eliminates the objections to consideration (1) since, by including each original data point, it provides an appropriate sampling scheme for the interpolation model (see Gold 1984).

(3) Any triangulation may be ordered as a binary tree when considered from a fixed viewpoint (Gold and Maydell 1978).

The tree ordering of consideration (3) is basic to the process of compiling contours, and needs to be described further.

A binary tree exists when each object (in this case a triangle) is accessed by only one ‘parent’ and itself accesses at most one left ‘son’ and one right ‘son’. This tree may be traversed in a variety of orders. The method described here used pre-order traversal to select each ‘new’ triangle to process, so that it is adjacent to previously-completed
portions of the map. In order to do this it must be able to label each edge of a proposed
triangle as facing 'up' or 'down' with respect to some arbitrary viewpoint.

If the viewpoint for the tree is considered to have average X and very large Y
coordinates (i.e. it is to the 'north'), and each triangle has edges numbered 1 to 3 in an
anticlockwise direction with edge 1 being the edge by which the triangle was entered,
three possible configurations exist. Since we are starting with the 'northernmost'
triangle and we enter the current triangle by edge 1, edge 1 must face upwards (1U)
towards the viewpoint. Edges 2 and 3 must each face up or down, but clearly they may
not both, along with 1, face up! Few complications exist where both face down (2D 3D):
here, in the fashion of binary trees, one edge (3D) is placed on a stack and subsequent
processing is performed on the new triangle adjacent to the 2D edge. Since we are
interested in a 'depth-first' (pre-ordered) traversal of the binary tree, all contour
segments are generated in the current triangle before proceeding further.

There remain '2U 3D' and '2D 3U' type triangles. Our objective is to complete the
map by ordering the parts (triangles) so that uppermost parts are completed before
lower parts, i.e. the map grows downwards by accretion, and there is guaranteed
monotonicity in the X direction of the 'active' map edge. Consequently no transfer of
control takes place in an 'up' direction. However, a little work with pencil and paper
will confirm that retaining both '2U 3D' and '2D 3U' triangles permits branches of the
tree to overlap. This occurs because edges 1 and 2 or edges 1 and 3 face upwards and
may thus be entered from either possible parent; thus the current triangle and its
descendants will be repeated in the branch of each parent. To prevent this, we have
chosen to reject entirely all '2D 3U' triangles as they will be handled satisfactorily when
entered from another edge. Hence for '2D 3U' triangles a branch of the tree is
completed, and we return to the stack for the next triangle forming the start of a new
branch. Consequently maps accrete in broad swaths moving in a northeast-to-
southwest direction. Figure 2 shows an example. (Note that vertical edges must be
handled consistently: for us they are 'down' when edge 2, 'up' when edge 3.)

This process of spatial ordering deserves more attention than we believe it has yet
received: not only areas (triangles) may be handled in this way, but also lines (edges) and
points (vertices) for any arbitrary triangulation or its dual. For example, traversing the
Delaunay triangulation of figure 1 in this fashion would permit efficient plotting of the
polygon boundaries, or processing of the polygons themselves in a top-to-bottom
order. Figure 3 (Gold and Maydell 1978) repeats the process of figure 2 but with the
viewpoint falling within the map area and some arbitrary criterion for stopping
applied.

In the example of figure 3, however, two further problems exist: firstly, the
processing of one branch of the tree in centre-outwards order does not guarantee that
the overlapped branch to its left (i.e. clockwise) has already been processed; and
secondly there is no guarantee that branches may not mutually exclude each other from
further progress, producing a deadlocked situation (G. Nagy, personal communi-
cation, 1986). The first case is readily handled by a minor modification of the tree-
ordering process so that triangles with two upward-facing edges are not processed
until they have been accessed by both parents. The second case can be handled where
(1) the triangulation is Delaunay and (2) the viewpoint is the circumcentre of a triangle.
This can be shown since Voronoi polygons are the dual representation of Delaunay
triangles, with each triangle being replaced by a node at its circumcentre, the nodes
being linked to form convex polygons around the data points. If the viewpoint is at one
of these nodes and all the polygons are convex, simply ensuring that, in traversing the
Figure 2. A triangulation ordered as two binary trees, formed from the north. The first starts with triangle 1, the second with triangle 8. The numbers of triangles represent their order of processing. Heavy lines represent boundaries between branches. (From Gold and Maydell 1978.)

tree, movement is from nodes closer to the viewpoint to nodes further away is sufficient to guarantee traversability. This process has obvious potential for two- (or more) dimensional neighbourhood searches.

It should be noted that when edges of triangles or dual-polygons are of interest they are processed whenever control is transferred from a parent triangle to its left son, or whenever its right son is put on the stack. When ordering vertices of triangles or the equivalent dual-polygons are required, processing takes place whenever a new vertex is encountered, i.e. not at all for ‘2U’ triangles, and once only for the basal vertex of ‘2D’ triangles (see figure 4). This processing is consistent with the fact that two new triangles are created for each data point inserted into a triangulation. If boundaries are ignored then there must be as many ‘2U’ triangles as there are ‘2D’.

Individual triangles can now be collected in some useful sequence: how are they to be processed once reached? As specified in consideration (2) at the beginning of this section, each triangle is regularly divided (as in figure 4) and elevation values estimated at each vertex. Basic orientations of triangles made available for processing by the process of tree traversal can be of only two types, ‘2U’ and ‘2D’, as illustrated in figure 4. Since edge 1 is assumed to be facing upwards and edge 3 downwards, ‘2D’ has edge 1 attached to the existing map. The processing of a sub-triangle consists of stringing any
necessary contours through it (an easy task in a triangle). Thus any contour strings crossing edge 1 must be appended to those existing at the same location on the active map edge. Edges 2 and 3 do not connect to the existing map, and any new contour strings initiated by interpolation within the triangle must be formed (‘opened’) across these two edges. In contrast, triangle ‘2U’ has edges 1 and 2 attached to the existing map, and any contour strings starting and finishing across these two edges must join (‘close’) two previously-existing contour strings.

Also illustrated in figure 4 is the necessary sequence of processing for sub-triangles within each major triangle. If N is a resolution factor specified by the user, each sub-triangle is subdivided into N x N flat triangular facets (all type ‘2D’ or ‘2U’) with interpolated values at their vertices. These are easy to contour, but the sub-triangles must be processed in a sequence consistent with the process of map accretion described earlier.

In keeping with the previously stated philosophy of preserving whatever topological information is available, interpolated contour line segments generated within sub-triangles each have a ‘from’ end and a ‘to’ end, and must be connected head-to-tail. This is of value for labelling, since numbers are written in order along the string, for area calculations and for the data structures used to maintain ordered lists of contour segments before plotting. Contour direction is defined as having higher ground on the left.
4. The contour building process

The ordered triangulation guarantees that we have map accretion radially outwards from some arbitrary viewpoint; if this viewpoint is significantly ‘north’ of the map sheet, accretion is from ‘north’ to ‘south’. East-west monotonicity of the active map edge permits the maintenance of a doubly-linked list of contour ‘stubs’ in the completed portion of the map that are available to be updated by the next triangle processed. Owing to the process of ordering triangles the next contour string to be updated is usually close to the last, and after each update the linked list is maintained with nodes ordered primarily on the X coordinate, and secondarily on the Y. This ‘main’ chain is thus an effective aid in updating contours as new segments are generated.

However, once the appropriate string has been located something must be done with the additional pairs of coordinates so identified. With the advent of economic memory we decided to preserve all coordinates in memory as long as possible, even in a personal computer environment. Above the main chain, therefore, are the ‘upper links’, preserving as many coordinate pairs as possible before flushing some or all of them out to the plotter or to a plot description file. Five fields are required by a main chain link: left, right and up pointers as well as X and Y coordinates already in the system. The upper links contain pointers in one direction only, from the ‘from’ end of a contour string towards its ‘to’ end.

Figure 5 illustrates the main components of the data structure: the main chain along the accretion edge and the upper links containing the accumulated contour strings. Each pair of X, Y coordinates in the main chain is duplicated in the closest associated upper link. The upper linked lists start at the first upper link at the ‘from’ end and terminate at the first upper link at the ‘to’ end. This path is followed if links are to be flushed out for plotting, which occurs if

1. a loop is completed;
2. sufficient space is accumulated to accommodate a label or label gap, and the link memory space is needed;
3. label space has not been found, but link memory has been exhausted.
Figure 5. Data structure for contour building. Boxes represent links containing pointers plus an (X, Y) coordinate pair. The solid lines (dark arrows) represent the contour strings having coordinate pairs added at the accretion edge. Open arrows on the upper links represent pointers to adjacent links along the contour string. Diamonds represent null pointers. The main chain along the accretion edge has left and right pointers to adjacent contour ends as well as an up pointer. ‘S’ represents the special status link containing a pointer preserving loop closure (dashed line).

Even if part of the upper linked list has been freed after plotting the list still completes a ‘from-to’ loop, but with a flag indicating that some links have been eliminated.

An additional feature of the main chain is an additional ‘status link’ above the main link at the ‘from’ end. While this carries general information about the string, such as elevation and type of line, its main function is to complete the loop formed by the upper links by pointing to the ‘to’ end main chain link. This status link was installed to help detect if the loop closure was partial or complete, but it is also necessary for the generation of contour trees.

As was discussed under triangle ordering, contour strings may be opened, updated or closed at the map accretion edge. Figure 6 indicates the structure installed on both sides of an accretion edge at the opening of a new loop. Note that in all cases the precise ordering of main chain elements is a function of their X, Y coordinates. Figure 6 also illustrates the updating of this original structure with an additional pair of coordinates on the ‘from’ end and again at the ‘to’ end.

The last alternative, loop closure, consists initially of an update where one end of a line segment matches an available main chain link. The other end, however, also matches another main chain link; one ‘from’ link and one ‘to’ link should, in the end, have the same coordinates, for otherwise our topological structure is in error. Figure 6 shows the result. Five links may be deleted and the upper links connected above the main chain. In addition the status link pointer must be updated, or, more properly, the surviving ‘to’ main link is relocated to the location expected by the status link, since the direction of the upper link pointers precludes access to the remaining status link.

Apart from updating and cleaning up the links (or, in practice, a more efficient combination), the remaining question is whether the closure merely joins two previously disparate contour segments, or whether a complete loop has in fact been
obtained. The answer is readily found by examining the status pointer. If it indicates that the ‘to’ end of the upper linked list matches the coordinates of the ‘from’ end after updating, the loop is complete. In this case the coordinates are sent to the plotter or plot description file and the upper and main chain links returned to the available list. Note that before plotting the data no searches along the upper linked lists are required; the status pointer effectively acts as an elastic band, ensuring that from opening a loop a complete circuit is always maintained, as in figure 6.

5. Contour tree generation

It is interesting to examine the status of a partially completed map at the time of closure of a contour loop, for example figure 5 with the central loop just completed. The
accretion edge can be considered to be an arbitrary line across an arbitrary contour map. The following rules apply:

1. Any contours enclosed by the completed loop must themselves have been completed at the previous stage and eliminated from the main chain.
2. Any contours enclosing the completed loop must not yet have been completed and must have ends recorded in the main chain.
3. Contour strings currently enclosing the completed contour must have ‘from’ and ‘to’ ends in the main chain on opposite sides of the closure point of the completed loop.
4. Since contours are defined as keeping high ground to the left, a higher contour will have its ‘from’ end to the right of the completed contour, and its ‘to’ end to the left. The reverse is the case for lower contours.

From the conditions described above, searching left along the main chain for the first ‘from’ end of a contour string, whose ‘to’ end (via the status pointer) is to the right of the completion point, will give the currently next higher enclosing contour. Searching to the right for a ‘from’ end, and checking if its ‘to’ end is to the left of the closure point, will give the currently next lower enclosing contour. The nearer in of these two is the currently enclosing contour. Of course, either (or neither) of these may be found. In this last case the contour just closed is a root. Since in the general case map boundaries may truncate several contours, a typical map will be a forest, i.e. several trees.

Given the relationships just discovered between a ‘child’ (the contour loop being closed) and the ‘parent’ (the currently enclosing, but not yet completed, contour), a series of links may be generated, one for each time a closure occurs, that forms the basis for a contour tree of the terrain model generated from the input data points. Since the process of contour closure generates leaves before roots, subsequently reprocessing in reverse order gives the links of the contour tree in root-to-leaf order, permitting a simple process of tree-building.

The links formed on the contour-generation pass are not, however, necessarily correct since they relate completed loops with those apparently surrounding them at that stage of map accretion. In some cases what appears to be an enclosure may

1. not close all within the map sheet; or
2. double back to form a horseshoe-shaped region only partially enclosing the previously completed loop.

These cases are readily caught in the tree-building process. In case (1), the link is ignored where the parent is never enclosed. In case (2), two links are generated, the first to the spurious parent and the second from there to the true parent enclosing both. An example is shown in figure 7, where a variety of closure events is illustrated on an imaginary contour map, along with the resulting contour tree.

Spurious relationships between nodes are eliminated from the preliminary tree by the following process:

1. Check pairs of nodes (contour strings) in the contour tree. The link is valid if
   a. the parent and child contours have the same height value, but they have the opposite sense, i.e. the parent is a clockwise loop and the child anticlockwise, or vice versa;
   b. the parent is the next highest contour, and both are clockwise loops; or
Spatially ordered networks and topographic reconstructions

Closure

Figure 7. (a) Imaginary contour map with closures due to north-to-south accretion. Large numbers represent the contour value; subscripts represent labels to distinguish individual loops. Heavy horizontal arrows indicate the apparent enclosing parent at the time of closure. Circular arrows indicate the sense (clockwise or anticlockwise) of the just-completed loop. Note the erroneous linkage between 2, and 3,. (b) Resulting contour tree. The dashed line shows correction of $2_1$-$3_2$ link to $2_1$-$2_2$. (c) the parent is the next lowest contour, and both are anticlockwise loops. (It should be noted that the sense of a contour loop may be determined at closure, but not before. Thus, the sense of a child is known at the time of closure, but the sense of the parent is unknown until it, in turn, is closed.)

(2) If a link is valid no change is needed. If the link is invalid the child should be linked tentatively to the next node up the tree (towards the root) and step (1) repeated. In practice, adjustment of links is a fairly infrequent occurrence.

6. Conclusions

The contour tree is generated from input elevation data essentially as a by-product of the contouring process. Nonetheless, the algorithms require careful development in order to preserve the required properties of spatial ordering that are needed to collect contour strings efficiently, to maintain the basic topological relationships at any stage of map generation, and to generate contour closures in a predictable sequence. The contour trees may be of interest in their own right for topographic analysis or they may form the basis for some additional output product. One example is closed-depression
annotation, where a specific type of line is assigned to contours whose node in the contour tree is a leaf (i.e. it encloses no contours itself) and which forms a clockwise loop (enclosing a depression). Other applications include contour interval shading where the shading algorithm requires interior and exterior boundaries of polygons, calculation of the hypsometric curve and similar area or volume analyses. While the emphasis in this study has been on using individual spot heights, other data on elevation, including flight-lines or digitized contour strings, are equally appropriate. In addition to contour generation and validation, the intrinsic concept of ordering within a triangular network should be of value in many other applications.

References


