

THREE-DIMENSIONAL RECONSTRUCTION OF RELIEFS WITH IRREGULAR FORMS STARTING FROM SERIAL SECTIONS

Serban B. PETRESCU¹, Tudor C. METEA², Catalin F. TUDOSE³

¹Computers Science Department, Politehnica University Bucharest,
Str Matei Elina Voevod, Nr 5, Sector 2, București

neuron@mailbox.ro

²Computers Science Department, Politehnica University Bucharest,
Str. Barbu Nicolae, Nr 8, Bl. 12, Sc. A, Ap. 13, Sector 4, București

mtudor@mailbox.ro

³ITC Networks, str. Eremia Grigorescu,
bl T1, Sc B, Et 3, Ap 10, Pitesti, jud Arges

catalin_tudose@yahoo.com

Threedimensional reconstruction of geographic zones relief is a process with wide applications in GIS type products, especially those that implement models for simulation of various phenomenons that may affect a specific geographic area (floods, terrain slips etc). By threedimensional reconstruction of geographic area relief starting from serial sections, we understand regenerating of the area surface in the form of a set of connected triangles. The results of the reconstruction process may be introduced in variuos threedimensional vizualization programs, or used as an input for relief analysis programs.

Keywords: Three-dimensional modelling, serial sections, level curves, polygon overlapping, tiling, triangularization, simple branching

1. INTRODUCTION

1.1 Phases of relief reconstruction process

During the process of threedimensional reconstruction of a geographic zone relief, there may be identified more stages, each stage representing a separated problem:

Input data processing stage. In this stage, the serial sections are read from a storage device or from a sppecific interface, as a set of level curves, with different altitudes. The format this data is stored depends of each program.

From structural point of view, a level curve is represented as a bidimensional set of points (each point with X and Y coordinates), with a third coordinate Z , common to all points in the set. The order of the points inside is level curve, from geometrical point of view, a level curve behaving as a polygon.

For easying the execution of the following phases, the input data is usually ordered during this phase in different manners, like the following:

- sorting the level curves by altitude

- reindexing of the points inside a level curve, such as the polygons have the same orientation (clockwise or counterclockwise), and the starting point should be an extreme point (e.g. the most left point).

Serial sections connection phase. This phase determines the adjacencies between serial sections, based on the altitude of each serial section and on the overlapping of the level curves in bidimensional plane.

The result of this phase is a connection graph, composed of one or more connection trees, the root nodes of each tree representing the base sections of the relief. The son nodes of a given node represent the serial sections of immediately superior altitude, overlapping the current section.

Connection analysing and processing phase. After setting the connection graph, each tree node is treated separately, and decided if it is processed in its original form, or if it needs additional transformations before introduced in the final processing phase.

In case a node has a single son in the connection graph, we say that the specified node presents a simple branching. Normally, such a connection doesn't need additional processing before it is passed to the final processing stage.

In case a node has more than one son, we say that the specified node presents a multiple branching. The reconstruction algorithms require special solution to build triangulated surfaces in this case. A solution for simple cases was proposed by inserting intermediate points at a level halfway between the two sections, and making it a single contour (Christiansen and Sederberg 1978). More complex situations demand that a polygon be inserted between the multiple contours to complete the surface (Ekoule et al. 1991, Meyers et al 1992).

Other solutions include those suggested by Shantz (1981) and Zyda et al (1987). Which interpolate new contours to decompose the problem to a series of one-to-one correspondences. For a one-to-many connection to be passed to the final stage, the connection needs to be split in more simple connections. The procedure consists in splitting the base level curve of the connection in more curves, each of them surrounding, from geometrical point of view, only one of the level curves represented by the son nodes.

As a result of this phase, the connection graph transformed into an independent set of connections, each connection being determined by a base curve, of lower altitude, and a top curve, of higher altitude. Due to the operations executed during this phase, the base curve may or may not be one of the input sections read during the first phase.

This way, each particular connection gets the form of a generalised cone-body, featured by a large base, a small base, and a lateral surface.

Lateral surface tiling phase. During this final phase, the lateral surfaces of the generalised cone-bodies is approximated by a set of connected triangles. The sets of triangles for each particular connection are then added to form the complete set of triangles that represent the approximation of the relief surface, this being the final result of the three-dimensional reconstruction process.

There have been a large number of proposals for the resolution of the tiling problem. Initial graph solutions (Fuchs et al 1977, Keppel 1975) used an optimal approach, based on graph theory, and gave a triangulation that best met a given criterion. Refinements to this approach, to improve upon performance have been suggested by Sloan and Painter (1988). The 'shortest span' method (Christiansen and Sederberg 1978) is one much heuristic, which looks for the shortest of the two possible edges between points in adjacent contours to create a triangular mesh. Heuristic algorithms were also been proposed by Ganapathy and Dennehy (1982).

The heuristic algorithms require contours to be aligned and similar in shape to work successfully. On the other hand, the complex algorithms give better results in most of the cases, but the computational effort is significantly bigger.

Each of the four three-dimensional reconstruction process characteristic phase is featured by a series of specific algorithms. The result of the reconstruction process is not unique, but depends of the algorithms applied at each step of the process. There are algorithms with very good results, but with a very large computational effort, and there are algorithms with good speeds, but with worse results.

Generally speaking, the algorithms for the first two phases of the reconstruction process do not rise problems in their developing, the special cases that may appear during the execution of these phases and that may harden the process, are very few.

In turn, the third and the fourth stage of the reconstruction process are featured by a big number of special cases, that may rise many problems in developing of the algorithms.

There are many algorithms for splitting a complex connection into multiple simple connections, and for the tiling of lateral surfaces of a connection. A part of these algorithms are restricted for a given relief configuration, becoming unusable for special cases where those restrictions are not met.

The exceptions that may affect the validity or invalidity of an algorithm may be of the following types:

- The polygons that represent the serial sections may not be convex.
- Some points of a higher level polygon of a given connection may not have direct visibility to any of the points of the lower level polygon. In the same way, some points of a lower level polygon of a given connection may not have direct visibility to any of the points of the higher level polygon. By direct visibility of one point to another, one should understand that the segment that connects those two points must not intersect any of the segments of the two polygons that form the connection.
- The polygons on the superior level of a connection may not have direct visibility to each other, their direct visibility being affected by the concavity of the base polygon.
- various other exceptions.

The relief forms that do not meet any of these exceptions may be considered simple relief forms. For treating this type of relief forms, there were developed a series of very efficient algorithms from computational effort point of view (linear or logarithmic complexity). Treating these cases do not represent the object of the current document.

The other relief forms may be considered in the complex relief forms category. Treating these cases may need more complex algorithms, which may need a larger computational effort (square or even cubic complexity).

In the current article, there will be presented two such kind of algorithms, the ones implemented in the "Neuron Relief Reconstruction" program, developed by Neuron company:

- a specific algorithm for treating multiple branching and splitting them into more simple branchings.
- an algorithm for treating the tiling problem of the lateral surfaces.

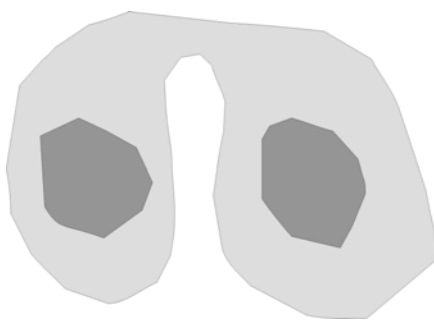


Fig. 1 The polygons on the superior level of a connection may not have direct visibility to each other

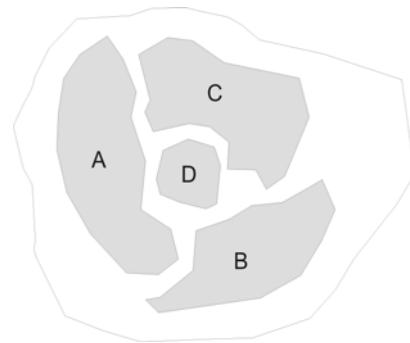


Fig. 2 No point of polygon D has direct visibility to any of the points of the base polygon.

2. ALGORITHM FOR TREATING MULTIPLE BRANCHING

During the three-dimensional reconstruction process, a series of particular cases that may affect the running of the third phase, may appear. From these cases we enumerate:

- The polygons on the superior level of a connection may not have direct visibility to each other, their direct visibility being affected by the concavity of the base polygon. This case is represented in figure 1
- Some polygons on the superior level do not have direct visibility to any of the points of the polygon on the inferior level, due to the position of the other polygons from the higher level. (Fig 2, polygon D).

In this chapter, it will be presented an algorithm for treating multiple branching, with applicability in a very large area of particular cases.

The algorithm presented in this chapter introduces the notion of generalised halfplane.

2.1 Double branching case

For algorithm description, we consider at the first step the case of a double branching (Fig. 3), this algorithm being extended, at the second step, to multiple branching cases.

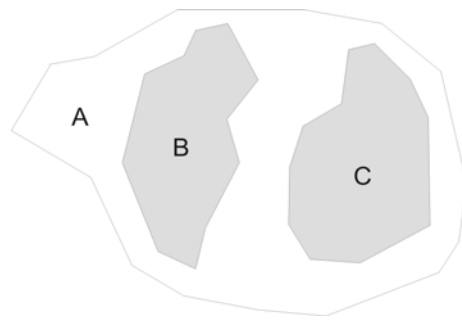


Fig. 3. A simple case of double branching

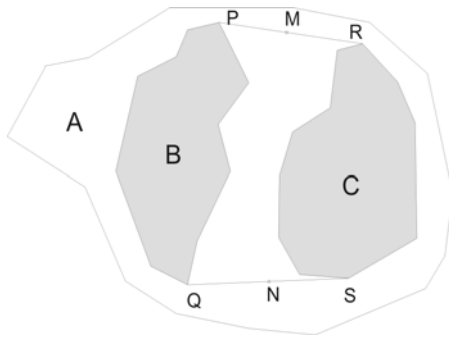


Fig 4 The two polygons B and C, the delimitation segments PR and QS, and their middles M and N.

The curves B and C have the same altitude, and are placed on the level immediately superior to the level of curve A.

Before proceeding to the execution of the algorithm, all the polygons involved in the process must be ordered in the same way (clockwise or counterclockwise), and the starting point must be placed at an extremity of the polygon (e.g. the most left position).

The first step of this algorithm is the determination of the delimitation straight lines D respectively E, as being the two straight lines that connect one point from each polygon, without intersecting any other of the segments of the two polygons.

We do not insist in describing the algorithm for determining these two lines. The simplest way (but not the most efficient from computational effort point of view), is analysing all possible segments that may be obtained from one point of each polygon, and verifying the property of a delimitation line. The only condition for this algorithm to work, is that the surrounding convex polygons of the two level curves should not be contained one in the other.

With these lines determined, only the useful parts of them will be kept. The useful part of a delimiting line consists of the segment between the points of the two polygons (PR, respectively QS, Fig 4)

There may be two cases that may determine an easy different way in later steps of the algorithm:

- the two delimitation lines are parallel
- the two delimitation lines intersect in a point O

This way, the two polygons B, respectively C, were split each of them into two portions:

- one exterior portion, which have the property that no point of this portion has direct visibility to the other polygon.

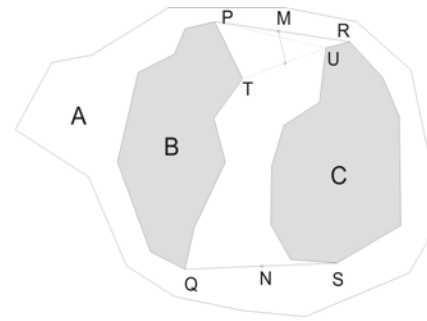


Fig 5 The beginning of the frontier delimitation problem. The TU segment was chosen and its middle was selected as next frontier point.

- one interior portion, with the property the points on the convex surrounding have visibility to at least one point of the other polygon.

Other two important points are the middles of the two delimitation segments, M, respectively N.

The second step of this algorithm consists of generating of a segmented line between these two points (M and N), that keeps an equal distance to the points of the two polygons, in other words, to function as a frontier between polygons B and C). For this, the two interior portions of the polygons are followed from points P and R to points Q and S. As a first segment, it is considered the segment PR. The middle of this segment is placed as the first point in the point set of the frontier.

We consider T, respectively U, the next points on the interior portions of the polygons B, respectively C. The next segment is chosen between segments PU, TR, respectively TU. The criteria of choosing the next segment depends of the parallelism of the D and E straight lines. In case 1 (the two lines are parallel), the segment that forms the smallest angle with any of the two lines is chosen. In case 2 (the two lines intersect in O point), the segments that forms the smallest angle with the O point is chosen. The middle of chosen segment is introduced as the next point of the frontier.

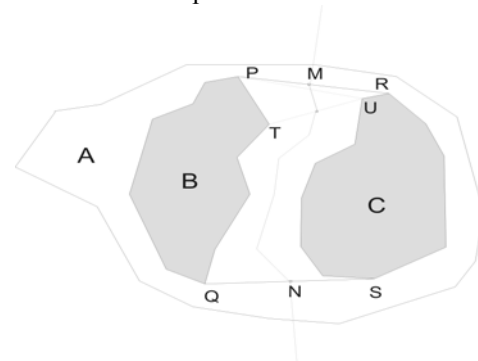


Fig 6 The finalization of the separating process. The M to N segmented line is completed with two halflines, determining two generalised halfplanes.

This action is repeated until Q and S points are reached, the middle of QS segment (the N point) being the last point of the frontier between the two polygons. The segmented line, obtained this way, is completed with two halflines, starting from points M, respectively N, with the direction ortogonal on PR, respectively QS, and with the sense against the segmented line. This way, we obtain an infinite length frontier line, that splits the plane in two generalysed halfplanes, one containing the B curve, and the other containing the C curve.

The last step of this algorithm is computing the intersection of the base curve A with each of the two halfplanes, splitting it into two polygons, one surrounding curve B and the other surrounding curve C, in a bidimensional perspective. If the frontier line intersects the base polygon in more than one contiguous portion, it can be demonstrated that only one of this portions could determine the splitting the base polygon in two parts such as one surrounds polygon B and the other polygon C.

2.1 Multiple branching case

The same algorithm described in the previous chapter may be applied, with small modifications, in a multiple branching case, with N polygons on the superior level. The algorithm applies in N-1 major steps, the result of each major step, being the frontier that separates one polygon from the others. The current polygon of a major step will not be included in the following steps, as well as the portion of the base polygon that was determined for the current polygon. In other words, each major step operates with a smaller base polygon and with a number of top polygon smaller by 1 from the previous steps. This way, the last step will operate with only two top polygons, and with a base polygon that surrounds only those two polygons, and neither of the other top polygons.

The algorithm for multiple branching is the following:

- the whole base polygon is the *reference base polygon* for the current major step
- for each polygon C_0 to C_{N-2} on the superior level
 - the C_i polygon is chosen as *reference top polygon* for the current major step
 - for each polygon C_{i+1} to C_{N-1} , the frontier that separates it from the reference top polygon; the generalysed halfplane that contains the C_i polygon is also determined
 - all the halfplanes obtained at this major step that contain the C_i polygon will be intersected to form another generalysed half plane, that also contains the C_i polygon. The intersection between this halfplane and the base polygon forms the *base polygon portion* that surrounds the C_i polygon.
 - this *base polygon portion* is extracted from the *reference base polygon*

3. ALGORITHM FOR TILING THE LATTERAL SURFACES

In the process of latterals surfaces tiling, a series of particular cases that may affect the algorithm may appear:

- The polygon from higher level is not completely surrounded by the base polygon (Fig. 7)
- The two level curves (the top and the base one) have points that do not have direct visibility to any of the points of the other polygon (Fig 8).
- One of the two polygons has a very long edge, compared to the edges on the same side of the other polygon (Fig. 9)

In first exception case, regardless of the solution, the result will contain triangles oriented down with their front side (the normal of the triangle plane has a negative Z component)

In the second exception case, the inconvenient may be eliminated by inserting additional points between the top and base planes. The algorithm of inserting and treating these points is described later in this chapter.

The third exception case may determine some algorithms to select improper triangles in the tiling process. The solution could be the splitting of the long edge into smaler edges, thus inserting points in the polygon without affecting its look.

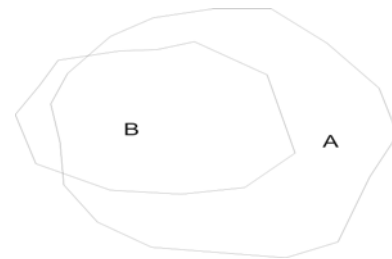


Fig 7 The B polygon from (higher level is not completely surrounded by the A polygon (lower level)

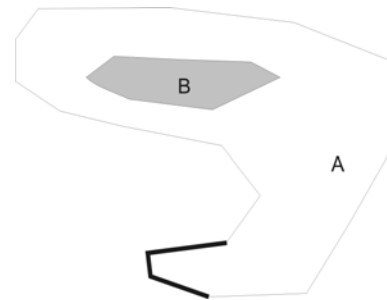


Fig. 8 The highlighted portion on the A polygon doesn't have direct visibility to any of the point of the B polygon

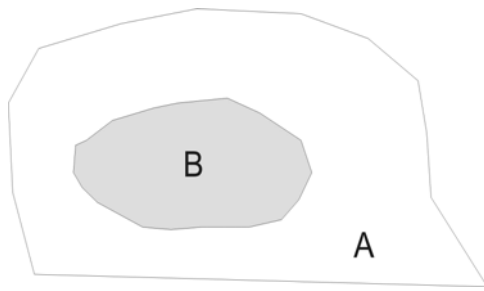


Fig. 9 One edge of the A polygon is very long compared to the segments of the B polygon

The algorithm described in this section is a cubic complexity algorithm, that assures validity to a great number of special cases.

The first step of this algorithm is creating a list with all possible segments formed by one point of each polygon, and sorting this list after the length of the segments, such as the smallest segments be first. The second step represents the selecting from these $M \cdot N$ segments of a number of at most $M+N$ segments, for the tiling process. The selection algorithm is the following:

- The output segments list starts with 0 segments.
- All the segments of the input list are analysed in the order of the length. A segment may be inserted in the output list if it doesn't intersect any of the two polygons segments, and neither any of the segments already inserted in the output list.
- The process stops in the moment the output list has $M+N$ elements, or when the input list reaches the end (the particular case where there are zones without visibility to the other polygon of the connection).

In the third step of the algorithm, the succession of the segments on the lateral surface must be determined, and the triangles are generated from consecutive segments with a common point.

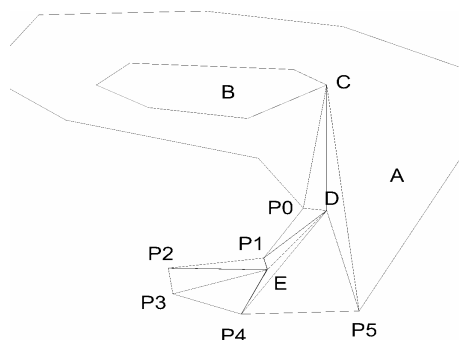


Fig 10 The cornering problem

In this phase all the points of the two polygons are analysed if they appear or not in the output segment list. If a sequence of points from one or another polygon is missing, the triangle generated with

surrounding points of those regions is hold as temporary. We may say that we found a cornering case.

For example, in Fig 10, the sequence P1 to P4 do not have visibility to the B polygon, so, no segments including these points will be inserted in the output list of segments. The surrounding points of this sequence are P0, respectively P5. The triangle C-P0-P5). The weight center of this triangle (D) is used as an auxiliary point for further processing. The triangles CP0D and CDP5 are included in the triangle list, and visibility of the points P1 from P5 is reevaluated from the point D. The points P1 and P4 become visible from D, so the triangles DP0P1, DP1E, DEP4 and DP4P5 are added to the triangle list (E is the weight center of DP1P4). At the next iteration with point E, we will see also the points P2 and P3, and the triangles EP1P2, EP2P3 and EP3P4 are found. The altitude of weight centers will be at one-third from the base and two-thirds from the top of the triangles where they are defined.

REFERENCES

- Christiansen HN., Sederberg TW., (1978) *Conversion of complex contour line definitions into polygonal element mosaics*, ACM Comput Graph 12:187-192
- Cook LT, Cook PN, Lee KR, Batnitzky S, Wong BYS, Fritz SL, Ophir J, Dwyer SJ, Bigongiari LR, Templeton AW (1980) *An algorithm for volume estimation based on polyhedral approximation*, IEEE Transactions and Biomedical Engineering, Vol 27
- Ekoule AB, Peytin FC, Odet L (1991) *A triangulation algorithm from arbitrary shaped multiple planar contours*, ACM Trans Graph 10:182-199
- Fuchs H, Kedem ZM, Uselton SP (1977) *Optimal surface reconstruction from planar contours*, Commun ACM 20:693-702
- Ganapathy S, Dennethy TG (1982) *A new general triangularization method for planar contours*, ACM Comput Graph 16:69-75
- Keppel E (1975) *Aproximation of complex surfaces by triangularization of contour lines*, IBM J Res Devel 19
- Meyers D, Skinner S, Sloan KR (1992) *Surfaces from contours*, ACM Trans Graph 11:2-11
- Shantz M (1981) *Surface definition for branching contour objects*, Comput Graph 15:242-270
- Sloan KR, Painter J (1988) *Pessimial guesses may be optimal: a counterintuitive search result*, IEEE Trans Patt Anal Mach Intell 10:949-955
- Zyda MJ, Jones AR, Hogan PJ (1987) *Surface construction from planar contours*, Comput Graph 11:393-408