CONTOURING
A TRIANGULATED IRREGULAR NETWORK
IN THEORY AND PRACTICE

Written by Tim Telcik (891892H)
Department of Geographic Information Systems
School of Computing
Curtin University of Technology
GPO Box U 1987
Perth W.A. 6001
Email: tim@marsh.cs.curtin.edu.au

Project Supervisor:
Steve Kessell

14th October 1991
# TABLE OF CONTENTS

List of Figures ........................................ iii  
List of Tables .......................................... iv  
Abstract ................................................ v  
Preface .................................................. vi  

1.0 INTRODUCTION ........................................ 1  

2.0 THE TRIANGULATED IRREGULAR NETWORK .............. 2  
2.1 Triangulation ....................................... 2  
2.2 Brief History ...................................... 3  
2.3 Applications ....................................... 3  
2.4 Advantages Over Regular Networks .................. 3  
2.5 Problems .......................................... 3  
2.6 Topology .......................................... 4  

3.0 THE CONTOURING PROCESS .............................. 6  
3.1 Interpolation ...................................... 6  
3.1.1 Interpolating Points in Triangles ............... 6  
3.2 Threading Contours ................................ 7  

4.0 CONTOURING A TRIANGULATED IRREGULAR NETWORK ... 9  
4.1 Methods of Generating Contour Points ............... 9  
4.2 Methods of Threading and Smoothing Contours ...... 10  

5.0 FALCON CONTOUR MAP ............................... 12  
5.1 Purpose ........................................... 12  
5.2 Development and Design ............................ 12  
5.3 Topology .......................................... 13  
5.4 Algorithms ........................................ 14  
5.4.1 Tracking Contours .............................. 14  
5.4.2 Finding a Contour’s Starting Edge ............. 16  
5.4.3 Finding the Vertex Completing a Triangle ..... 16  
5.4.4 Smoothing Contours ............................ 16  
5.4.4.1 Interpolating Edge Cuts .................... 17  
5.4.4.2 Adjusting Edge Cuts ......................... 18  
5.4.4.3 Interpolating Points Along a Bicubic Spline .. 19  
5.5 Efficiency ......................................... 20  
5.6 Problems .......................................... 20  
5.7 Test Runs .......................................... 21  

6.0 CONCLUSION ......................................... 25  

7.0 BIBLIOGRAPHY ...................................... 26
LIST OF FIGURES

Figure 1. A simple TIN with 10 points .................. 2
Figure 2. The Delauney criterion .......................... 2
Figure 3. A TIN with labelled points and triangles .... 4
Figure 4. (a) Linear interpolation of a triangle ......... 7
(b) Linear interpolation of sub-triangles .......... 7
Figure 5. (a) Generating all TIN contours .............. 9
(b) Generating all triangle contours .......... 9
Figure 6. A bicubic spline. 0,3 are control points; 1,2 are end points .................. 10
Figure 7. A TIN point and neighbour list ................. 13
Figure 8. An edge record ................................. 14
Figure 9. (a) Tracking a closed contour ............... 15
(b) Tracking an open contour ...................... 15
Figure 10. (a) Linearly interpolated contours showing edge cuts .................. 18
(b) The same contour with adjusted edge cuts 18
Figure 11. Bicubic spline across triangle boundaries ... 19
Figure 12. Unsmoothed contours over a 10 point TIN .... 22
Figure 13. Smoothed contours over a 10 point TIN
(adjust = 20%, sample = 0.1 unit) ........ 22
Figure 14. Unsmoothed contours over an 80 point TIN .... 23
Figure 15. Smoothed contours over an 80 point TIN
(adjust = 20%, sample = 1 unit) ........ 23
Figure 16. Unsmoothed contours over a 100 point TIN .... 24
Figure 17. Smoothed contours over a 100 point TIN
(adjust = 20%, sample = 1 unit) ........ 24
LIST OF TABLES

Table 1. Triangle topology for Figure 3. ................. 4
Table 2. Point topology for Figure 3. ..................... 5
Table 3. Timings for contours generated without smoothing . 20
ABSTRACT

A Digital Elevation Model (DEM) called a Triangulated Irregular Network (TIN) can be used to represent a surface such as a terrain. The TIN structure has many applications including contouring. Contouring a TIN requires interpolating points where the contours intersect the triangles and threading the points with a suitable curve. A computer program called Falcon Contour Map has been written to generate smoothed contours over a TIN.
I would like to thank the following:

Curtin’s Computing Science Department for providing equipment and inter-library loans funding;
Steve Kessell for proofing this report and going out of his way to help (as usual);
Paul Rayner for supplying initial contouring references;
Doug Hudson for help with contouring and suggesting an edge list;
Russell Doak for supplying his contouring work;
Andrew Marriott, Phil Dench and Shane Douglas for help with programming in C;
The Water Authority of Western Australia for supplying the TIN data;
Trish Evans for proofing this report and helping me maintain some form of sanity;
Richard Hammer for help with smoothing contours;
Various people over the mailing network including Michael McCullagh, Tom Poiker, David Douglas, Mark Kumler and Paul Joyce;
and finally to the Abacus vendor for all the Mars bars!
1.0 INTRODUCTION

Computers and computer graphics have found an increasing application in the field of cartography. They can be used to represent terrain as a Digital Terrain Model (DTM) or Digital Elevation Model (DEM). A DEM is concerned only with modelling elevation whereas a DTM is concerned with other attributes of the terrain (Burrough 1986).

Data points used to represent terrain may be regular or irregular in pattern and can be represented as a grid or triangular network.

This report examines the process of how smooth contour lines can be generated over a Triangulated Irregular Network (TIN) and discusses how a computer program called Falcon Contour Map was developed to implement this process.

Section 2 deals with the TIN data model, section 3 deals with the contouring process, section 4 deals with contouring a TIN, section 5 describes how Falcon Contour Map (FCM) was developed, and section 6 is a discussion of the TIN as a spatial data model and the applicability of FCM to contouring.

Falcon Contour Map will be used by the Surface Water Branch of the Water Authority of Western Australia to generate contours from their TIN data.
2.0 THE TRIANGULATED IRREGULAR NETWORK

The Triangulated Irregular Network (TIN) is a digital elevation model (DEM) which uses elevation data points from the terrain as vertices of triangles to form a continuous triangular mesh (see Figure 1).

![Figure 1. A simple TIN with 10 data points.](image)

2.1 Triangulation

There are many ways in which a set of data points can be triangulated. One of the best schemes for creating a unique triangulation is the Delauney triangulation (Lee and Schacter 1980, Watson 1981, Watson and Philip 1984).

A Delauney triangulation is based on the criterion that when three points are joined to make a triangle, the circle drawn through its vertices (i.e. circumcircle) will not encompass any other data points (see Figure 2).

![Figure 2. The Delauney criterion.](image)
The Delauney criterion effectively produces nearly equilateral triangles (i.e. interior angles are close to 60 degrees) thereby avoiding long thin triangles, and it results in points being connected with their nearest neighbours in the data set (Jones et al. 1989). The Delauney triangulation is the dual of the Thiessen polygon network (also called Voronoi or Dirichlet regions) and can be computed from the same set of points (Fowler and Little 1979).

2.2 Brief History

Some of the early work on geographic data structures and triangle-based terrain models was by Peucker (1973). This was continued by Peucker and Chrisman (1975) with greater detail placed on topological encoding and adjacency analysis.

The Triangulated Irregular Network (TIN) was initially used for representing bathymetry (Peucker et al. 1976), but was later shown as a surface tool for solving different spatial problems (Peucker et al. 1978). Christopher Gold, (Gold 1978), working independently of Peucker et al., produced similar work on the generation and application of a triangular data structure.

2.3 Applications

TIN’s can be used to calculate slope and aspect of the terrain, produce profiles and hill shading, generate contour lines, solve line of sight problems (Peucker et al. 1978, Poiker 1990) and locate drainage networks and watersheds (Silfer et al. 1987, Jones et al. 1989, Poiker 1990).

2.4 Advantages Over Regular Networks

A triangulated irregular network is preferable to a regular network for representing a surface because it can adapt to a change in the roughness of terrain (Peucker 1980), use fewer data points to represent the same surface, and can be topologically encoded so that adjacency analysis is more easily computed (Aranoff 1989). Kumler (1990) illustrated that a TIN provided a better representation of the terrain than a regular gridded network.

2.5 Problems

The success of any TIN hinges upon the careful selection of points to be used as vertices in the network (Kumler 1990). Due to the TIN’s variable resolution the number of elevation data points should only be large where there is great variation in the terrain. If this is not the case then the TIN may be less efficient at representing the terrain than a gridded network and important terrain features will be missed. Subsequently, the derived products (eg. contour maps, drainage networks) will be inaccurate.
A typical TIN is not a true 3-dimensional (3D) data structure, it is only 2-and-a-half-dimensional (2 1/2 D), because it cannot represent folds in the terrain which appear in the cases of overhanging cliffs and cave entrances.

2.6 Topology

Poiker (1990) notes two methods for storing TIN's:

1. Triangle by triangle (see Figure 3 and Table 1).
   For each triangle store:
   (i) a reference identification
   (ii) \((x,y,z)\) coordinates of the vertices
   (iii) reference identifications of neighbouring triangles
   (n.b. neighbours are ordered clockwise starting from the north, and -1 represents a neighbour outside the TIN)

![Figure 3. A TIN with labelled points and triangles.](image)

<table>
<thead>
<tr>
<th>ID</th>
<th>V1((x,y,z))</th>
<th>V2((x,y,z))</th>
<th>V3((x,y,z))</th>
<th>TRI NBRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3(5, 3, 3)</td>
<td>2(15, 0, 2)</td>
<td>1(0, 0, 0)</td>
<td>B -1 C</td>
</tr>
<tr>
<td>B</td>
<td>4(11, 5, 4)</td>
<td>2(15, 0, 2)</td>
<td>3(5, 3, 3)</td>
<td>E A D</td>
</tr>
<tr>
<td>C</td>
<td>7(5, 10, 9)</td>
<td>3(5, 3, 3)</td>
<td>1(0, 0, 0)</td>
<td>A -1 I</td>
</tr>
<tr>
<td>D</td>
<td>5(7, 7, 4)</td>
<td>4(11, 5, 4)</td>
<td>3(5, 3, 3)</td>
<td>H B G</td>
</tr>
<tr>
<td>E</td>
<td>6(12, 9, 8)</td>
<td>2(15, 0, 2)</td>
<td>4(11, 5, 4)</td>
<td>F B H</td>
</tr>
<tr>
<td>F</td>
<td>10(15, 15, 2)</td>
<td>2(15, 0, 2)</td>
<td>6(12, 9, 8)</td>
<td>-1 E M</td>
</tr>
<tr>
<td>G</td>
<td>7(5, 0, 9)</td>
<td>5(7, 7, 4)</td>
<td>3(5, 3, 3)</td>
<td>J E C</td>
</tr>
<tr>
<td>H</td>
<td>6(12, 9, 8)</td>
<td>4(11, 5, 4)</td>
<td>5(7, 7, 4)</td>
<td>K E D</td>
</tr>
<tr>
<td>I</td>
<td>9(0, 5, 6)</td>
<td>7(5, 10, 9)</td>
<td>1(0, 0, 0)</td>
<td>L C -1</td>
</tr>
<tr>
<td>J</td>
<td>8(10, 11, 7)</td>
<td>5(7, 7, 4)</td>
<td>7(5, 10, 9)</td>
<td>L K G</td>
</tr>
<tr>
<td>K</td>
<td>8(10, 11, 7)</td>
<td>6(12, 9, 8)</td>
<td>5(7, 7, 4)</td>
<td>M H J</td>
</tr>
<tr>
<td>L</td>
<td>9(0, 5, 6)</td>
<td>8(10, 11, 7)</td>
<td>7(5, 0, 9)</td>
<td>N J I</td>
</tr>
<tr>
<td>M</td>
<td>10(15, 15, 2)</td>
<td>6(12, 9, 8)</td>
<td>8(10, 11, 7)</td>
<td>N F K</td>
</tr>
<tr>
<td>N</td>
<td>10(15, 15, 2)</td>
<td>8(10, 11, 7)</td>
<td>9(0, 5, 6)</td>
<td>-1 M L</td>
</tr>
</tbody>
</table>

Table 1. Triangle topology for Figure 3.
2. Points and their neighbours (see Figure 3. and Table 2).

For each point store:

(i) an identification number
(ii) an \((x,y,z)\) coordinate
(iii) identification numbers of neighbouring points

(n.b. neighbours are ordered clockwise starting from the north, and \(-1\) represents a neighbour outside the TIN)

<table>
<thead>
<tr>
<th>ID</th>
<th>((x,y,z))</th>
<th>POINT NEIGHBOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0, 0, 0</td>
<td>9 7 3 2 -1</td>
</tr>
<tr>
<td>2</td>
<td>15, 0, 2</td>
<td>10 -1 1 3 4 6</td>
</tr>
<tr>
<td>3</td>
<td>5, 3, 3</td>
<td>7 5 4 2 1</td>
</tr>
<tr>
<td>4</td>
<td>11, 5, 4</td>
<td>6 2 3 5</td>
</tr>
<tr>
<td>5</td>
<td>7, 7, 4</td>
<td>8 6 4 3 7</td>
</tr>
<tr>
<td>6</td>
<td>12, 9, 8</td>
<td>10 2 4 5 8</td>
</tr>
<tr>
<td>7</td>
<td>5, 10, 9</td>
<td>8 5 3 1 9</td>
</tr>
<tr>
<td>8</td>
<td>10, 11, 7</td>
<td>10 6 5 7 9</td>
</tr>
<tr>
<td>9</td>
<td>0, 15, 6</td>
<td>10 8 7 1 -1</td>
</tr>
<tr>
<td>10</td>
<td>15, 15, 2</td>
<td>2 6 8 9 -1</td>
</tr>
</tbody>
</table>

Table 2. Point topology for Figure 3.

Slope analysis, such as drainage networks, needs a triangle based topology, whereas traversing procedures, such as contouring and line of sight problems, work best with a point based topology. Provided one topology can be extracted from the other in close to linear time either approach can be used (Poiker 1990).
3.0 THE CONTOURING PROCESS

Contouring is a two stage process:

(i) Interpolating points on the contour lines

(ii) Threading contour points with some type of curve

This process has been described by Elfick (1979), Petrie and Kennie (1987), Gold (1986), Peucker (1980), Dobkin et al. (1990), and Lewis and Robinson (1978).

3.1 Interpolation

Interpolation is the process by which missing data points are estimated. The derived values are not the 'true' values, only a mathematical 'best guess' based on the known values (Aranoff 1989).

Interpolation is based on the principle that points very close together in space will have similar values (Burrough 1986). In almost all cases the required values must be interval or ratio scaled (Waters 1990).

3.1.1 Interpolating Points in Triangles

Elfick (1979) lists two methods for interpolating points within a triangle:

(i) Linear interpolation: computing the required value on the triangle’s edge using the edge’s end points.

(ii) Surface fitting: fitting a surface to the six points defined by the apex points of each triangle and the three associated triangles.

Elfick (1979) also notes that the interpolation process should depend on the nature of the original data and the type of result required. The interpolating function is important because it determines which type of smoothing will be required.

There are many methods of local triangle-based interpolation. Jones and Wright (1991) compared the applicability of six different interpolation schemes. The two methods of triangle surface fitting (or patching) illustrated were:

(i) A weighted average of planar functions defined at the triangle vertices (Franke 1982).

(ii) The Clough-Tocher finite element method (Clough and Tocher 1965; Lancaster and Salkauskas 1986).

The choice of interpolation scheme has a dramatic effect on the contours. Several different interpolation schemes should be tested (i.e. implemented) before adopting a final scheme (Jones and Wright 1991).
McCullagh (1981, 1988, 1991) describes a method in which each triangle is divided into a set of smaller triangles based on the length of the longest side. A surface patch (e.g. Akima 1978, McCullagh 1981) is applied to the triangle to determine the heights of the sub-triangle vertices. A tracking procedure is then applied to the sub-triangles to calculate the intersection points along the triangle edges and pass a straight line through them (see Figure 4).

![Figure 4. (a) Linear interpolation of a triangle (b) Linear interpolation of sub-triangles](image)

Sabin (1985), Schut (1976), Barnhill (1974), and Lancaster and Salkauskas (1986) describe other methods of interpolating data points within a triangle.

### 3.2 Threading Contours

Once the points on the contour line have been found (interpolated), the next step is to fit (thread) a suitable curve through the points to form a contour line.

The fastest and most robust method is to join each of the points in the contour line with a straight line. This gives a good first impression of how the contours cross the terrain. Provided the data set is large this will produce relatively smooth contours because of the short distance between points.

Although the surface is continuous in value (called $C^0$), linear interpolation (i.e. fitting a plane to the triangle vertices) is not very appealing because of the angularity that occurs at the intersection of neighbouring triangles (Miller 1988; Dale and McLaughlin 1988; McCullagh 1988; Petrie and Kennie 1987; Poiker 1990).

If smoother contour lines are required, a curve fitting technique such as a spline can be applied to each contour. This produces a curve which is continuous at several points and thus less angular in appearance. Unfortunately, it creates the problem of contours crossing when the contour lines are close together in a steep region of land (Petrie and Kennie 1987; Dale and McLaughlin 1988; Sabin 1985).
Smoothing a line in vacuo can be a very dangerous thing to do; it is possible to make the lines individually smooth, but it is not possible to ensure absolutely that they do not cross (McCullagh 1988).
4.0 CONTOURING A TRIANGULATED IRREGULAR NETWORK

Generating contour lines over a TIN requires:

(i) Generating (interpolating) contour points over the TIN.

(ii) Threading the interpolated contour points with some curve. A straight line can be used if no smoothing is required, or a spline or surface interpolation can be used if smoothing is required.

4.1 Methods of Generating Contour Points

There are two basic methods for generating contour points over a TIN:

(i) Tracking (or following) every contour through the TIN and outputting them as they are completed (Peucker 1973, Peucker et al. 1978, Poiker 1990) (see Figure 5a).

(ii) Finding all of the contour intersection points within each triangle and linking them into open and closed contours when all of the triangles have been used (Haverlik and Krcho 1973, Gold 1978, Elfick 1979, Doak 1990) (see Figure 5b).

While the tracking approach requires minimal processing when performing a simple linear interpolation, it is not practical to use a local surface patch for interpolation because a triangle’s patch would need to be calculated every time it was crossed by a contour.

Lewis and Robinson (1978, p.330) commented that ‘the advantages of this method (i.e. tracking) as opposed to that of evaluating all the desired contours in each consecutive triangle is that sensible labelling of the contours may be easily achieved and the contour smoothed if required’.

![Figure 5](image)

**Figure 5.** (a) Generating all TIN contours (b) Generating all triangle contours
Finding all contours within a triangle is better suited for using a local surface patch because all of the contours within a triangle are interpolated using the same surface patch which only needs to be calculated once. However, this approach has the overhead of requiring interpolated contour points to be linked into open and closed contours.

### 4.2 Methods of Threading and Smoothing Contours

Threading is the process by which a curve is passed through all the points in a contour. A B-spline is not an appropriate curve fitting technique for smoothing because it passes a curve near the contour points, not through them, and results in unwanted high curvature when contours are close together.

Miller (1988) found that spline techniques were too compute-intensive and described a method of locally adjusted curve approximations. However, this method, like the B-spline, passes a curve near the data points, not through them.

Cline (1974) and Lauzzana and Penrose (1990) describe methods of curve fitting and smoothing using a spline under tension which can be adjusted to change the amount of curvature required. While Cline’s approach is a complete spline using all of the contour data points as controls, it is relatively efficient because it only solves a tri-diagonal matrix and does not require any messy pivoting or full inversion (Joyce 1991). Lauzzana and Penrose introduce a piecewise bicubic spline generated between two end points and using points either side as controls (see Figure 6).

![Figure 6. A bicubic spline. 0,3 are control points; 1,2 are end points.](image)

Akima (1970) describes a method of curve fitting for smoothing which passes through the data points and produces a curve comparable to that drawn by a draughtsman (i.e. only minimal curvature).

A triangle within the network is usually represented by a plane fitted to its vertices. By fitting a 3D surface patch to a triangle and its neighbours, contours will be smoother because
the interpolated points are continuous at the triangle’s first derivative (called $C^1$).

The increased quality of smooth contours exacts a price in terms of computation. Whereas a linear contour interpolation is cheap to compute, the smoothed surface represents an investment in computer time proportional to the level of smoothness required (McCullagh 1988).
5.0 FALCON CONTOUR MAP

5.1 Purpose

I found myself at the Surface Water Branch of the Water Authority of Western Australia asking if there were any Geographic Information Systems (GIS) related problems which required solving. After some discussion (and a good many ‘ums’) it was revealed there was the need for a program to contour a Triangulated Irregular Network (TIN) because the commercial contouring programs in use were too slow and resulted in contours which crossed.

Here began my quest. I needed to write a program that created smoothed contours over a TIN and did so in reasonable time. Doak (1990) wrote a TIN contouring program which would have provided an excellent starting point; unfortunately this work was unavailable until this semester. As a result my program takes a different approach to Doak’s in terms of the TIN topology used, the storage requirements for the topology, and the method of generating contours. A literature search during the first semester of this year (state and overseas) provided the necessary contouring information.

A computer program called Falcon Contour Map (FCM) has been written in ANSI C. It reads TIN point topology from an ASCII data file, generates smoothed contours over a TIN and outputs the contours to a monitor and data file. Writing began on the 20th of July this year and finished on the 5th of October. The completed program was delivered to the Water Authority on the 11th of October 1991.

Contour data output from FCM will be used as input to a Computer-Aided Design (CAD) package called MicroStation (Intergraph Corporation 1991) for further processing.

5.2 Development and Design

The hardware and software platform on which Falcon Contour Map is required to operate have greatly influenced its design.

The C program code is to be compiled using MetaWare’s High C compiler (MetaWare Inc. 1990) and run using a Disk Operating System (DOS) extension called PharLap (PharLap Software Inc. 1991). This combination enables an IBM compatible PC with an Intel 80386 Central Processing Unit (CPU) to use the hard disk as an extension to main memory (i.e. virtual memory). Hence, the program is only limited by the storage space available on the hard disk.

The virtual memory available for the program resulted in the point topology that represents the TIN being stored in main memory rather than in a disk file that must be continually accessed. Because main memory can be accessed faster than a disk file the topology can be quickly retrieved and the tracking routine run at an optimum.
A straightforward user interface has been written which prompts the user to enter values and answer queries, or use the defaults values provided.

Graphics routines have also been written to display the contours and TIN on a monitor based on the Graphical Kernel System (GKS) concepts of windows, viewports, and device-independent routines.

Device-dependent graphics device drivers have been written for the IBM PC and Silicon Graphics Personal Iris which are external to the calling display routines and can be easily modified to accommodate a new machine.

If the machine running Falcon Contour Map does not have graphics capabilities, the display routines can be easily removed.

5.3 Topology

The TIN point topology is implemented as an array of points with each point having an \((x,y,z)\) coordinate, a linked list of neighbours and a flag for whether it has been visited (see Figure 7).

A TIN point is stored in the points array at the position determined by its identification number (i.e. the id number is used as the array subscript). I decided to use a static array for the TIN points because they can be directly accessed using the array subscript and the size of the TIN used will probably be known by the user.

Dynamic memory was chosen for the list of point neighbours because a point will have on average six neighbours, but it may have more in some situations. The point neighbour list is only singly linked (i.e. the list can only be traversed in one direction) because searching mostly occurs in one direction. By keeping pointers to the start and end of a neighbour’s list a doubly-linked list can be simulated (see Figure 7).

![Figure 7. A TIN point and neighbour list.](image-url)
A TIN edge is stored as two vertices \((v_1,v_2)\) and a flag for whether it is active (see Figure 8). An array of sorted TIN edges is used by the tracking procedure for two reasons:

(i) To determine a starting edge for a given contour
(ii) To ensure that only one contour passes through a triangle edge at a given contour height (level)

This has the disadvantage of requiring additional memory, but was chosen because the program is intended to run on a virtual machine.

A maximum possible size for the TIN point and TIN edge arrays must be determined by the user, the relevant constants changed and the program recompiled.

5.4 Algorithms

5.4.1 Tracking Contours

Contour tracking is the process by which a contour is generated by following it through the TIN until it forms an open or closed line. If a contour is closed it will have only one line segment which starts and finishes at the same point (see Figure 9a). An open contour will have two line segments originating at the same point, one from tracking forwards (clockwise) and one from tracking backwards (anti-clockwise) (see Figure 9b).


Before tracking can begin an array of TIN edges must be created. The TIN edges are sorted into ascending order based on the lowest vertex in the edge (i.e. \(v_1\)). The tracking only requires edges which have the second vertex \((v_2)\) above or equal in height to the first vertex \((v_1)\). This reduces the total number of edges stored and the search time to locate an edge.
The edge being investigated is called the present edge. The vertex above the contour is the reference-point and the vertex below is the sub-point. The lowest vertex on the edge is \( v_1 \) and the highest is \( v_2 \), thus \( v_1 \) is the sub-point and \( v_2 \) is the reference-point.

To track more than one contour through the TIN a loop is used which increments the contour height by the contour interval between the lower and upper contour bounds. A simplified version of the sub-point, reference-point tracking scheme is shown below:

* Search for a starting edge in the sorted edges array (i.e. \( v_1 <= \text{contour} <= v_2 \))
* While the contour is not complete
  * Let the present edge equal the starting edge
  * While the contour has not reached a boundary or formed a loop (i.e. returned to its start)
    * Interpolate point along present edge where contour height crosses and add it to contour points
    * Flag present edge as used so it cannot be used again
    * Find the vertex completing the triangle (i.e. \( v_3 \)).
    * If the \( v_3 \) is above the contour then the reference-point (\( v_2 \)) in the present edge is replaced with \( v_3 \)
      Otherwise \( v_3 \) is below the contour so the sub-point (\( v_1 \)) is replaced with \( v_3 \).
      Note: The present edge which was formerly the entry edge for the triangle has now become the exit edge.
  * If the present edge is on the boundary then the contour has reached the TIN boundary
  * If the present edge is already used then exit loop
End While Loop
* If the contour has formed a loop then the contour is complete so exit loop (i.e. stop tracking)
* If the contour has reached the TIN boundary while tracking forwards then return to starting edge and track backwards
* If the contour has reached the TIN boundary while tracking backwards then the contour is complete so exit loop (i.e. stop tracking)
End While Loop
This algorithm is very efficient at accessing the point topology, caters for contours passing through a triangle’s vertex and eliminates degenerate contours (i.e. contours at the same height which pass through the same triangle edge).

5.4.2 Finding a Contour’s Starting Edge

As seen in the algorithm described in Section 6.4.1 the first step of tracking a contour is to find a starting edge. The contouring problem thus centres on finding contour starting points and ensuring that the contours through these points have not already been determined (Mark 1978). Mark (1977,1978) uses a surface network to solve this problem.

I have opted for a somewhat simpler, but effective, approach suggested by Doug Hudson (pers. comm. 1991) which involves creating a sorted edge list based on the edge’s lowest vertex ($v_1$). By performing a linear search on the edges a starting edge can be found which contains the contour height (i.e. $v_1 \leq \text{contour height} \leq v_2$), and which is inactive (i.e. edge has not been used by another contour at that height).

5.4.3 Finding the Vertex Completing a Triangle

The tracking scheme is quite simple and operates on the principle that a contour entering through one edge of a triangle must exit through one of the remaining two edges (unless it passes through a triangle vertex). Thus, given the entry edge of a triangle which contains vertices $v_1$ and $v_2$ we need the third vertex ($v_3$) which completes the triangle so that an exit edge can be determined.

The clockwise ordering of neighbours around a point enables $v_3$ to be easily found. As mentioned earlier, $v_1$ is the sub-point (SP) and $v_2$ the reference-point (RP). When tracking forwards (clockwise) $v_3$ is the point after the SP in a RP’s neighbour list, and when tracking backwards (anti-clockwise) $v_3$ is the point before the SP in the RP’s neighbour list.

For example, refer to Figure 3 and its point topology in Table 2. If the entry edge has point 1 as $v_1$ and point 3 as $v_2$ then $v_3$ will be point 7 when tracking forwards and point 2 when tracking backwards.

5.4.4 Smoothing Contours

The best method for smoothing contours from a mathematical perspective is the local area surface patch which can be used to interpolate contours passing through a triangle.

This was not used for two reasons:

(i) It is computationally expensive
(ii) Help arrived too late!
I turned towards a spline-based technique to smooth the contours. Sabin (1985) and Miller (1988) have placed great caution on using splines such as B-splines to smooth contours. A B-Spline is inappropriate for smoothing contours because it does not pass through the contour points, only near them.

The main problem with using splines for smoothing contours is that they can cross one another where the terrain is steep because each contour is smoothed separately.

There were two possible solutions to this problem:

(i) Using an entire spline under tension (Cline 1974)
(ii) Using a piecewise spline under tension (Lauzzana and Penrose 1990)

Discussing the problem with Richard Hammer (pers. comm. 1991) revealed one major problem with an entire spline – a very large matrix must be solved. This may be feasible for a small set of points but not for a large set. He suggested an heuristic curve fitting technique developed by Akima (1970) as a way around this.

Even though Cline’s approach (Cline 1974) uses an entire spline, it only solves a tri-diagonal matrix. Joyce (1991) who suggested this approach also mentioned that it was quite efficient.

Lauzzana and Penrose (1990) described a piecewise curve fitting method using a bicubic spline with a tension factor that could be changed to vary the curvature on the spline.

The method of smoothing implemented in Falcon Contour Map is a combination of the piecewise bicubic spline (Lauzzana and Penrose 1990) and adjusting the contours edge cuts (Richard Hammer, pers. comm. 1991). This involves three steps:

(i) Interpolating the entry and exit points along a triangle’s entry and exit edges
(ii) Adjusting the entry and exit points
(iii) Interpolating points along a bicubic spline which passes through the adjusted exit point on one triangle and the adjusted entry point on its neighbouring triangle

Part (i) above is part of the linear tracking scheme. Parts (ii) and (iii) are the actual smoothing and have been included into the tracking scheme to make it efficient.

5.4.4.1 Interpolating Edge Cuts

To interpolate where the contour intersects an edge (i.e. the edge cut) the edge is treated as a vector which runs from the lowest vertex ($v_1$) to the highest ($v_2$) and is represented in parametric form.

A point along the edge (vector) is determined using the following calculations:
\[
x = x_1 + a \cdot t \\
y = y_1 + b \cdot t \\
z = z_1 + c \cdot t
\]

where: 
- \(x_1\) is the x coordinate of the lowest vertex
- \(y_1\) is the y coordinate of the lowest vertex
- \(z_1\) is the z coordinate of the lowest vertex
- \(x_2\) is the x coordinate of the highest vertex
- \(y_2\) is the y coordinate of the highest vertex
- \(z_2\) is the z coordinate of the highest vertex
- \(a = x_2 - x_1\)
- \(b = y_2 - y_1\)
- \(c = z_2 - z_1\)
- \(t\) is the parametric term; ranges from 0 to 1

The value of 'z' (the contour height) is already known and can be used to calculate 't' (the distance along the vector) by rearranging equation (3) in terms of 't' to become:

\[
t = (z - z_1) / c
\]  

Substituting 't' back into equations (1) and (2) will give the values of 'x' and 'y' where the contour of height 'z' crosses the edge.

### 5.4.4.2 Adjusting Edge Cuts

When passing a straight line through linearly interpolated entry and exit points on a triangle's edges there is a sudden change in direction of the contour between neighbouring triangles (see Figure 10a).

By treating the line through the triangle as a vector in parametric form, the positions of the entry and exit points can be adjusted (or displaced) by a suitable amount (see Figure 10b). A value between ten and twenty percent of the vector's length produces acceptable results.

![Figure 10](image)

Figure 10. (a) Linearly interpolated contour showing edge cuts (b) The same contour with adjusted edge cuts
The coordinates of the adjusted entry and exit points can be determined using the following calculations:

\[
\begin{align*}
\text{adjusted entry } x &= x_1 + ax \cdot \text{adjust} \\
\text{adjusted entry } y &= y_1 + by \cdot \text{adjust} \\
\text{adjusted entry } z &= z_1 \\
\text{adjusted exit } x &= x_1 + a(1 - \text{adjust}) \\
\text{adjusted exit } y &= y_1 + b(1 - \text{adjust}) \\
\text{adjusted exit } z &= z_1
\end{align*}
\]

where:
- \(x_1\) is the x coordinate of the entry point
- \(y_1\) is the y coordinate of the entry point
- \(z_1\) is the z coordinate of the entry point
- \(x_2\) is the x coordinate of the exit point
- \(y_2\) is the y coordinate of the exit point
- \(z_2\) is the z coordinate of the exit point
- \(a = x_2 - x_1\)
- \(b = y_2 - y_1\)
- \(\text{adjust}\) factor by which entry point is displaced from start of vector
- \((1 - \text{adjust})\) factor by which exit point is displaced from start of vector

5.4.4.3 Interpolating Points Along a Bicubic Spline

When the edge cuts have been adjusted they can be used as the end points of a bicubic spline with the points either side as control points (see Figure 11).

![Figure 11. Bicubic spline across triangle boundaries.](image)

By interpolating points along a bicubic spline passing through the adjusted exit point of one triangle and the adjusted entry point on its neighbouring triangle a reasonably smooth contour is produced. A weighting factor of 0.4 was chosen for the curvature of the bicubic spline because it produces a smooth curve without excessive curvature (from Doak 1990).

The level of smoothing can be modified to meet the needs of the user by changing the amount of adjustment on the edge cuts and the number of points interpolated (sampled) along the bicubic spline.
The contours created using the above two methods do not produce cartographic quality contours, but they are reasonably smooth and have two advantages:

(i) They are fast to compute
(ii) They do not cross

It should be noted that as the level of smoothing is increased the accuracy of the contour map decreases because the smoothed contour points move further away from the original data.

It is suggested that if any accurate processing is required from the contours on a TIN, only a linear interpolation should be performed because this provides the most accurate set of contours that can be created from a given TIN.

### 5.5 Efficiency

Falcon Contour Map has been tested on several IBM compatible PC’s running between 8 and 33 MHz, and on a Silicon Graphics Personal Iris. The time taken to generate contours with no smoothing over different sized TIN’s is shown in Table 3 below. Due to memory management on the PC’s, the large data sets could not be tested.

<table>
<thead>
<tr>
<th># PTS</th>
<th>PC (8 MHz)</th>
<th>PC (12 MHz)</th>
<th>PC (33 MHz)</th>
<th>Iris</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4 s</td>
<td>1 s</td>
<td>1 s</td>
<td>&lt; 1 s</td>
</tr>
<tr>
<td>80</td>
<td>34 s</td>
<td>4 s</td>
<td>1 s</td>
<td>&lt; 1 s</td>
</tr>
<tr>
<td>100</td>
<td>44 s</td>
<td>9 s</td>
<td>2 s</td>
<td>&lt; 1 s</td>
</tr>
<tr>
<td>115</td>
<td>54 s</td>
<td>12 s</td>
<td>2 s</td>
<td>&lt; 1 s</td>
</tr>
<tr>
<td>519</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>3 s</td>
</tr>
<tr>
<td>1061</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>10 s</td>
</tr>
</tbody>
</table>

Table 3. Timings for contours generated without smoothing.

Note: - PC timings based on internal clock
- Iris times vary according to system load

### 5.6 Problems

The major problem with Falcon Contour Map is the large amount of main memory required for storing the point topology and related edges. Thus, for a real-world data set (i.e. over 20,000 points), a virtual based machine is required to provide the necessary memory during run-time.

As the edge cut adjustment factor is increased and the spline sampling interval decreased the number of points representing a contour will increase. Because the contour is stored as a static array the number of contour points generated may overflow the contour array and cause the program to crash. As with the point and edge arrays, the size of the contour array may need modifying to meet the level of smoothness required and the size of the data set to be used.
At present the sorted edge list is searched using a combination of binary and linear searches. As the data size increases so will the number of edges and the time taken to search for a given edge. It is expected that the machine in use will slow down with a large data set, but if it grinds to a halt then one of two options may need considering:

(i) Using a different search strategy for the edges  
(ii) Taking a different approach for finding the starting edge and flagging used edges

5.7 Test Runs

Figure 12 shows the contours generated over a small TIN of only 10 data points without smoothing. Figure 13 shows the same TIN which has been smoothed using an edge cut adjustment (ECA) factor of 20 percent and a bicubic spline (BCS) sampling interval of 0.3 unit.

Figure 14 depicts the contours generated over an 80 data point TIN without smoothing. Figure 15 shows the same TIN smoothed using an ECA factor of 20 percent and a BCS sampling interval of 1 unit.

Figure 16 illustrates contours generated over a 100 data point TIN without smoothing. Figure 17 is the same TIN that has been smoothed using an ECA factor of 20 percent and a BCS sampling interval of 1 unit.
Figure 12. Unsmoothed contours over a 10 point TIN.

Figure 13. Smoothed contours over a 10 point TIN (adjust = 20%, sample = 0.1 unit).
Figure 14. Unsmoothed contours over an 80 point TIN.

Figure 15. Smoothed contours over an 80 point TIN (adjust = 20%, sample = 1 unit).
Figure 16. Unsmoothed contours over a 100 point TIN.

Figure 17. Smoothed contours over a 100 point TIN (sample = 20%, adjust = 1 unit).
6.0 CONCLUSION

Automated cartography is a field which has expanded enormously in the short time of its existence. As could be expected a number of problems have also appeared, mostly related to programming a computer to generate contour maps that are comparable to those of a cartographer. At the same time there has been a great deal of work on how spatial data can be represented within a computer and the operations that can be performed on it.

Many different spatial data structures have been created for representing a terrain. Two of the more important tessellations are the gridded and triangulated networks which have been tested extensively.

The Triangulated Irregular Network (TIN) has been proven as an efficient data model for representing a terrain and a tool for performing spatial analysis by many researchers including the original developers Peucker et al. (1978).

A program called Falcon Contour Map (FCM) has been written to create smooth contours over a TIN. Many of the aesthetic aspects of commercial contouring packages, such as contour labelling, have not been included because FCM was never intended to be a stand-alone program. Instead, FCM is a module which will contour a TIN and export contour data to other mapping packages, such as MicroStation, for further processing.

Falcon Contour Map is efficient for a data set with 80 - 1000 elevation points in the TIN. At the time of writing this report the program had not been tested on a large data set (eg. 20,000 points) but it is expected to perform comparably with the commercial contouring programs currently in use by the Surface Water Branch at the Water Authority of Western Australia.

Falcon Contour Map has addressed the need to efficiently store and access TIN topology and produce reasonably smooth contours which do not cross. However, automating cartography to meet the needs of cartographers and scientists is an on-going research problem which has not been met by this report.
7.0 BIBLIOGRAPHY


MetaWare Inc. (1990) MetaWare High C. Vers. 2.32 (revised). Computer software. MetaWare Incorporated. IBM PC with Intel 80386 CPU.


n.b. T.K. Peucker and T.K. Poiker are the same author