Efficient Contour Line Labelling for Terrain Modelling

Xin Xie **Burkhard C. Wünsche**

Graphics Group, Department of Computer Science
University of Auckland,
Private Bag 92019, Auckland 1142, New Zealand,
Email: xxie021@aucklanduni.ac.nz, burkhard@cs.auckland.ac.nz

Abstract

Terrains are an essential part of outdoor environments. Terrain models are important for computer games and applications in architecture, urban design and archaeology. A popular and intuitive way to represent terrains is by contour maps. In order to render such representations in 3D the contours must be labelled with height values and converted to Digital Elevation Maps (DEM), which are regular grids of height values and are represented as gray scale images.

The labelling of contour lines is time intensive for large maps and prone to errors. In this paper we present an efficient and novel algorithm for semi-automatically labelling contour maps and for converting them to DEMs. The algorithm first identifies *point extrema* which must be labelled by the user. The point extrema are connected by a graph and the contour lines crossed by edges are labelled automatically. We show that ambiguities can exist for so-called *line extrema*. Our algorithms will resolve ambiguous regions requiring a minimal number of additional user inputs. We also present a more efficient graph representation, which requires about 10-20% more user inputs than the optimal case. After the contour lines are labelled, the contour map is triangulated and the height value at each point of the DEM is computed using a bilinear interpolation.

The presented algorithm is efficient, requires minimal user inputs, and produces good quality DEMs. We present several examples, discuss its suitability for different applications, and provide a complexity analysis.

Keywords: contour map, digital elevation map, shortest-path spanning tree, Delaunay triangulation, terrain visualization

1 Introduction

Terrain modelling is fundamental to a wide range of computer applications, such as games, flight simulators, urban design and planning, archaeology, and geography and engineering visualizations. The two most popular representations of terrains are contour maps and Digital Elevation Maps (DEM).

Digital elevation maps consist of a regular grid of height values and can be stored and represented by gray scale images where the gray level of each pixel represents the height value at that location. White/black represent the highest/lowest altitude, respectively. DEMs are popular for rendering since they can be compressed, are suitable for out-of-core rendering, and can be converted to progressive and multi-resolution representations, which are essential for real-time rendering of massive data sets (Duchaineau et al. 1997, de Boer 2000, Losasso & Hoppe 2004). Human users often have difficulties to perceive terrain details from DEMs since the human visual system's gray scale resolution is limited and colour/gray scale perception is non-linear. The ability to compare colours (gray scales) at different locations is even more limited since the perceived colour depends on the colour of surrounding regions (simultaneous contrast) (Schiffman 1996). Many of these disadvantages can be resolved by rendering the DEM in 3D. This however, requires complex rendering algorithms, more screen space and animations to fully perceive the 3D geometry.

A more effective and intuitive representation of terrains for human viewers are contour maps, also known as topographic maps. Contour maps consist of a series of non-intersecting curves or lines, so-called contour lines, which represent points of equal height in the terrain. Contour lines are always one-dimensional, i.e., they cannot represent planar regions of constant height, and are either closed or have endpoints on the boundary of the image. Contour maps are usually prepared using airborne and satellite altitude measurements, or photogrammetric interpretation of aerial photography (Wikipedia n.d.), which gives an accurate depiction of terrain features (U. S. Army Corps of Engineers and American Society of Civil Engineers 1996). In contrast to DEMs contour maps allow height comparisons of spatially distinct or neighbouring regions. Magnitude and direction of gradients can be perceived from the contour line density and orientation. Figure 1 shows an example of a contour map and the corresponding digital elevation map.

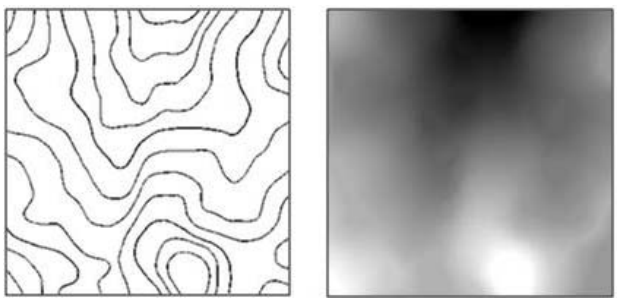

Figure 1: Example of a contour map (left) and the corresponding digital elevation map (right).

In order to display terrains represented by 2D contour maps in 3D they must be converted to DEMs. A precondition for this conversion is that height values

are given for all contour lines. This is usually the case when using Geographic Information Systems (GIS), but for many cases such height values must be input by hand which is cumbersome and prone to errors. Our research was motivated by two applications: one application was an archaeological visualization system where we were given hand drawn contour maps of a terrain (Kymer 2009). The second application is a sketched based terrain modelling tool, where users create 3D terrains by sketching contour lines, mountain peaks and rivers. Examples of both applications are used in the result section.

Section 2 reviews previous work on generating and converting contour maps. Section 3 and 4 present definitions, properties and theorems required for the development of an efficient algorithm for labelling contour lines. The resulting algorithm and implementation details are presented in section 5 and 6, respectively. Results are given in section 7 and are followed by our conclusions and suggestions for future work.

2 Literature Review

Contour maps are the most common representation of terrains in maps and the 3D visualization of map data improves realism and recognition of terrain features. Consequently there have been many projects concerned with the conversion of contour maps to DEMs. The majority of research concentrates on the image processing of map data and the interpolation of contour line data.

Image processing research is concerned with the recognition of contour lines from digitised map data, especially in the presence of textured backgrounds and noisy data. A typical procedure consists of the following four steps (Arrighi & Soille 1999):

1. digitalization of the topographic map, e.g., by using a scanner
2. thresholding
3. thinning of the black patterns by using some skeletonization procedure and
4. raster-to-vector conversion of the resulting thinned lines.

Arrighi and Soille extend this procedure based on advances in morphological image processing (Arrighi & Soille 1999). The most attractive aspect of their method is the reconnection of contour lines that are disconnected by coordinate grid lines or inscriptions. The authors extract the two ends of a line before skeletonising them by generalizing the hit-to-miss transform. Rivas and de La Fraga use a similar method, but also discuss an improved way to generate sample points from contour lines in order to create smoother terrain meshes (Rivas & de la Fraga 2005). Lalonde and Li perform contour line extraction with the use of colour information. They extract the basic colours of an image by switching from RGB to L*a*b colour spaces, projecting the image on its principal axes and using modified histogram splitting (Lalonde & Li 1997). A smoothness function assists with the reconnection of broken contour lines.

Once contour lines have been identified they must be labelled. When reading map data this can be achieved using character recognition techniques and by finding the corresponding contour lines, usually by using a distance metric criteria. Surprisingly we found only one previous research project concerned with our research, the semiautomatic assignment of height values to contour lines. Maia and Xavier (Maia & Átila L. F. Xavier 1996) construct a nested tree by representing each contour line by a tree node. The tree reflects the relationship among all contours in the domain. Each node is then given an ambiguity interval which is initialised to [-∞, +∞] or a correct elevation value where possible. If a true elevation value is known the information is propagated through the tree in order to reduce each node's ambiguity interval. The authors claim that this method can effectively eliminate human interactions in the elevation assignment process such that human errors remain at the minimum.

The last step necessary for computing DEMs is to compute height values at its regular grid points. This requires interpolation of height values at irregular sample points, usually obtained by sampling the contour lines. Dakowicz and Gold introduce three interpolation methods based on a Delaunay triangulisation of the terrain (Dakowicz & Gold 2002): bilinear (barycentric) interpolation of triangle vertex values, and a so-called "Gravity interpolation" and "Sibson interpolation". The central idea of the "Gravity interpolation" is that the weighting of each data point used is inversely proportional to the square of the distance from the data point to the grid node being estimated, whereas the "Sibson interpolation" inserts each grid point temporarily into the Voronoi diagram of the data points and measures the area taken from each of a well-defined set of neighbours. Rognant et al. present a dual representation of contour lines and DEMs based on a Constrained Delaunay Triangulation (CDT) (Rognant et al. 2001). The data structure might be useful in interactive modelling applications using both contour and DEMs representations.

3 Definitions and Properties

3.1 Properties of Contour Lines

We did not find any precise definition of properties of contour lines. In many cases the actual properties depend on how the contour map was derived. We developed a set of properties which conforms to (often implicitly assumed!) properties used in the literature and with the properties of contour lines in widely used GIS software packages such as ArcInfo:

Equidistance: The height difference Δ_{height} between two neighbouring lines is constant (or zero if lines belong to the same iso-level).

Completeness: Contour lines contain no gaps and they are either closed or end at the map boundaries.

Orthogonality: The contour lines are orthogonal to the terrain gradient and the terrain gradient is orthogonal to the contour line. This also means that the contour lines must be smooth, i.e., contain no corners.

Continuously differentiable: The contour lines do not overlap, intersect or branch. The terrain gradient can be extracted from the distance and height difference between points on the contours.

Virtually all contour maps we examined fulfilled these properties. Exceptions were digitised contour maps where gaps could occur due to sampling errors and mathematically defined contour maps where branching can occur, e.g., for a function with a cross shaped extrema at an iso-level. Such cases are virtually non-existence for real terrains and would not be produced by software generating contour data.

3.2 Extrema in Contour Maps

A contour map can have four topographical types of extrema:

1. Closed contours, which have no other contours within it (e.g., mountain tops)
2. Contours connected to the boundary, which have no other contours on one side (e.g., slopes intersecting the boundary)
3. Closed contours containing additional closed contours and being higher/lower than the adjacent contours (e.g., the rim of a volcanic crater)
4. Contours connected to the boundary, with the contours on both sides being higher/lower (e.g., valleys/ridges)

For the purpose of our research we can differentiate these four topographical types of extrema into two topological types:

Definition 1 *A* Point Extremum *is a contour line which encloses, possibly together with part of the boundary, a region containing either no contours or all other contours.*

This definition includes case (1) and (2) of the topographical types of extrema. Note that all points in the region enclosed by a point extremum are considered to have identical height values. A point extremum can therefore be reduced to a single point without changing the topology of the terrain - hence the choice of the name. Note that if a contour line does not have neighbouring contour lines on both sides, then it is a point extremum. Several examples of point extrema are shown in red in the two images on the left hand side of figure 2.

Point extrema can be visually identified as closed contours or contours connected to the boundary, which contain no other contours. Section 6 will present an efficient algorithm for identifying point extrema which uses a Delaunay triangulation required for interpolating height values.

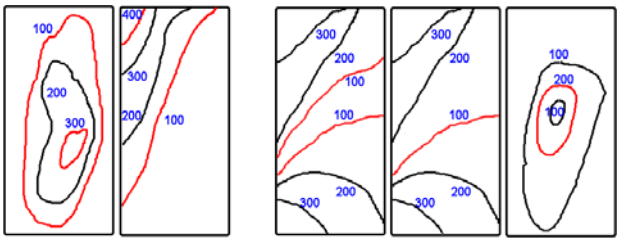

Figure 2: Left: Two images showing point extrema (red). Right: Three images showing line extrema (red).

Definition 2 *A* Line Extremum *is a contour line which is either not higher or not lower than the contour lines on either side of it.*

Several examples of line extrema are shown in red in the three images on the right hand side of figure 2. In contrast to point extrema, line extrema cannot be detected from the contour information alone. As an example consider a domain with three contours having the elevations:

Case 1: 400m, 300m and 200m
Case 2: 400m, 300m and 400m

In either case the contour lines on the left and right can be identified as point extrema. However, only in the second case the contour line in the middle is a line extrema.

Note that the line extremum in case (2) can be identified by providing height values for the point extrema on either side. However, even in this case the height value of the contour line in the middle cannot be determined since it could be lower (300m) or higher (500m) than its neighbouring contour lines. In contrast, in case (1) the height value of the centre contour line can be immediately determined from the heights of the neighbouring contours.

This observation suggests that an effective way to label all contour lines is to first label the point extrema and then deduce the height values in between automatically wherever possible. The following subsection proves several theorems required for the development of an efficient algorithm for labelling contour lines.

4 Graph-Theoretical Foundations

As indicated in the previous section our proposed solution will be based on labelling point extrema first. These extrema can be easily identified and, as we proof next, always exist:

Theorem 1 *If a domain contains one or more contour lines, then it has at least one point extremum.*

Proof. Suppose that a domain or its subset contains one or more contour lines and none of them are point extrema.

Case 1 If the contour line is closed and has no other contours inside, then by definition it is a point extremum, which contradicts the assumption.

Case 2 If the contour line is closed but contains one or more contours, then the contours must be closed (i.e., not connected to the boundary) since otherwise they would intersect the surrounding contour. In this case the innermost one is a point extremum (proof by induction), which contradicts the assumption.

Case 3 If the contour line is connected to the boundary and the region surrounded by it contains no other contours, then by definition it is a point extremum, which contradicts the assumption.

Case 4 If the contour line is connected to the boundary and the region surrounded by it contains N contours, then it either contains a closed contour (case 1 or 2), or a contour connected to the boundary forming a region with at most N-1 contours (case 3 or 4). By structural induction this case can be reduced to case 1-3, i.e., the region contains a point extremum, which contradicts the assumption.

Hence by proof of contradiction any domain containing contour lines contains at least one point extremum. □

Theorem 2 *The height value of a point extremum cannot be inferred from the heights of its neighbouring contour lines.*

Proof. By definition a point extremum is a contour line which encloses a region which contains no other contour lines. Hence it has neighbouring contour lines on at most one side. Even if the height value for the neighbouring contour line is known, the point extremum can be higher or lower than that contour line. Hence the height value cannot be inferred. □

Because of theorem 2 it is clear that any contour line labelling algorithm requires user input of height values for all point extrema. As mentioned previously

we will present in section 6 an algorithm for identifying these extrema. We will show next that knowledge of the point extrema is sufficient to identify all other contour lines:

Theorem 3 *If all point extrema in a domain are connected by a spanning tree, then all contour lines will be crossed by at least one edge of the tree.*

Proof. Theorem 3 is equivalent to

If there is a contour line which is not crossed by any edge of the spanning tree, then there must be a point extremum which is not connected to the spanning tree.

Assume there is a contour line that is not crossed by any edge of the spanning tree and all point extrema are part of the spanning tree.

Case 1 The not-crossed contour line is closed and contains no other or all other contours in the domain. In this case it is a point extremum by definition. This contradicts the assumption that all point extrema are part of the spanning tree.

Case 2 The not-crossed contour line is open, i.e., its endpoints lie on the boundary of the domain and the line divides the domain into two regions. If one of these regions contains no contour lines, then the contour line is a point extremum by definition. This contradicts the assumption that all point extrema are part of the spanning tree.
If both regions contain contour lines, then by theorem 1 both regions must contain at least one point extremum. Since we assumed that all point extrema are connected by a spanning tree this means that the spanning tree must contain at least one edge connecting one extremum from either side. Hence this edge must cross the contour line dividing the regions. This is a contradiction to the assumption that the contour line is not-crossed.

Hence by proof of contradiction all contour lines in a domain are crossed by the edges of a spanning tree connecting all point extrema in the domain. □

Theorem 4 *If none of the contours between two point extrema is an extremum then all of these contours have either strictly monotonically increasing height values or strictly monotonically decreasing height values.*

Proof. Assumed that not all of the contour lines between the point extrema have either strictly monotonically increasing height values or strictly monotonically decreasing height values. In that case there is at least one contour line which either has a neighbouring contour line with equal height value or where the neighbouring contour lines are either both higher or both lower. By definition 2 this contour line is a line extremum (or point extremum if closed), which contradicts the requirement that there are no other extrema in between these two point extrema. □

5 Algorithm Design

Our algorithm requires that the contour map fulfils the properties listed in subsection 3.1. We also assume that it contains no other information such as text or geographic features. A significant amount of research has been going into image processing tools extracting contour lines from real map data, but is outside the scope of this paper (Xin et al. 2006, TERRAINMAP.COM - Digital Elevation Modeling Journal n.d., Arrighi & Soille 1999).

Our algorithm is based on the theorems proven in the previous section. In particular theorem 2 shows that any algorithm for labelling contour lines requires user inputs for all point extrema. Theorem 3 shows that in order to label all contours it is sufficient to consider contours crossed by the edges of a spanning tree connecting all point extrema. Hence our algorithm contains the following steps:

Algorithm 1: Contour Line Labelling

Step 1: Identify point extrema and request appropriate height values from the user

Step 2: Create a *contour graph* whose nodes are point extrema and connect them by edges such that every node is connected to any other node. This means that the graph represents or contains a spanning tree and hence, with theorem 3 every contour line is intersected by at least one edge.

Step 3: Investigate the number of contour lines intersected by each edge. Let A and B be two nodes of the graph with heights h_A and h_B and n the number of contour lines intersected by the edge between those nodes and Δ_{height} the height step between two contour lines.
Step 3A: If

$$h_B - h_A = (n-1) * \Delta_{height} \tag{1}$$

then there is no extremum in between the nodes A and B and with theorem 4 we can label the intersected contour lines with strictly monotonically decreasing or increasing height values.
Step 3B: Otherwise all contour lines between the nodes are labelled as ambiguous and the user has to choose one of them and label them appropriately. The labelled contour line will become a new node of the graph. For all new edges it is checked whether

$$h_B - h_A > (n-1) * \Delta_{height} \tag{2}$$

If yes, then the user input was invalid since it violates the equidistance condition of the contour map. If no, apply step 3A to each section of the edge subdivided by the new node.
Repeat step 3 until all edges of the contour graph have been processed.

Step 4: If all edges of the graph have been investigated, but not all contours have been labelled than the input contour map has an invalid topology, e.g., gaps between contour lines.

The algorithm has three critical aspects:

- The choice of the edges of the contour graph influences the efficiency and effectiveness of the labelling process. A full graph grows quadratically in size, but minimises user inputs in the sense that contour lines can be automatically labelled wherever possible. A spanning tree grows linearly, i.e., less contour lines must be investigated by the algorithm, and it still guarantees that all contour lines are labelled. However, in some cases it might not be optimal, i.e., its edges do not cover all cases where an automatic labelling is theoretically possible.
- If a shortest spanning tree is used edge weights must be defined. Possible choices are the Euclidean distance or the number of contours between nodes (point extrema). Different weighting functions result in different SSTs and hence can result in different numbers of user inputs.

- If an automatic labelling in step 3 is not possible the user must choose to label a line extrema in order to minimise user inputs. As explained in subsection 3.2 it is not possible to identify the exact location of a line extrema or to deduce its height value. The user must make this choice from additional information not available in the contour map. This could be a real map from which the contour map was derived or it could be design choices when modelling a contour map from scratch.

These issues will be discussed in more detail in section 7.

6 Implementation

6.1 Contour Line Indexation

We represent contour maps by gray scale images that only contain contour curves and, as mentioned previously, exclude other geographic data such as background details and inscriptions. In order to label contour lines they must be first identified and then indexed. The indexing makes it possible to determine whether two points belong to the same contour line. The contour lines are first skeletonised such that they form a simple 8-connected line, i.e., all pixels of the line are connected via horizontal, vertical or diagonal neighbouring pixels. We use an extension of a simple border tracing algorithm (Marita n.d.). A pixel is set to zero if it does not belong to a contour line and otherwise its gray scale value represents the index of the contour line to which it belongs. Note that if a map contains more than 256 contour lines other image formats could be used. For example, 3 byte RGB colours provide more than 16 million indices. More details of our implementation of the border tracing algorithm are given in (Xie 2008).

6.2 Sampling and Triangulation

In order to obtain a continuous height field for the domain we sample the contour lines and triangulate the sample points. We currently sample the contour lines at regular intervals (e.g., 10 pixels). Very thin triangles are avoided by adjusting sample points at the boundaries accordingly (Xie 2008). An improved solution would take into account contour density and curvature in order to avoid very small triangles and increase sample point density in regions of high curvature.

In the next step the sample points are triangulated. By interpolating height values at the triangle vertices this results in a continuous representation of the height field. Desirable conditions for a triangulation are:

1. The solution should be unique for a given set of points, regardless the sequence of points inserted.
2. The shapes of each resultant triangle should be equilateral or close to equilateral, if no other constraints are specified.
3. The vertices of each triangle are the nearest neighbour points, such that the perimeter of the triangle is minimum.

These conditions reduce numerical errors and gradient discontinuities in the resulting interpolation. The above conditions define the well-known and popular Delaunay triangulation. We use an implementation of the Bowyer-Watson algorithm (Arens 2002) which has a run-time of $n \log(n)$ where n is the number of sample points.

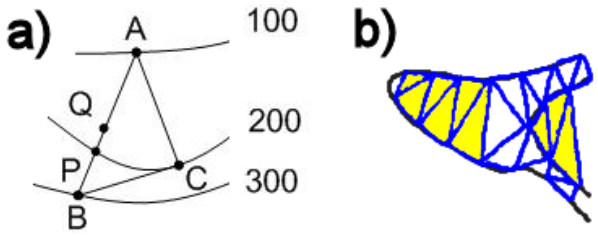

Figure 3: (a) Contour intersected by an edge of the triangulation (b) Flat area in a triangulation.

The basic Delaunay triangulation has two disadvantages. The first one is that the semantic information given by the contour lines is ignored. Figure 3 (a) shows an example: Assume the points A, B and C are sample points and form a triangle in the Delaunay triangulation. If this triangle is then used to reconstruct height values using a bilinear (barycentric) interpolation, then for the line $\overline{AB}$ the height value of $200m$ would be at the point Q, whereas according to the contour lines it should be at the point P. The problem can be avoided by ensuring that the sample point distance is smaller than the contour line distance. Another solution is to force triangle edges to lie on the contour lines by using a Constraint Delaunay Triangulation (CDT) (Peterson 1998). If the sample point density is low and contour line curvature is high this can lead to triangles "cutting off" part of the contour line. In our examples we found that a standard Delaunay triangulation is sufficient. We suggest that the best solution would be a hybrid approach where constraint edges are only used in regions of high contour line density.

The second disadvantage of the Delaunay Triangulation is that it leads to unnatural flat regions as illustrated in figure 3. Note that this effect also occurs for a CDT and can even be stronger in that case. The best ways to avoid this effect are to force the triangulation to use sample points from different contours where possible or to use a higher order interpolation with a larger support (i.e., the weighting factors of the base functions also take into account height values of sample points outside the interpolated triangle).

6.3 Detection of Point Extrema

Subsection 3.2 showed that point extrema represent regions of equal height value. We can therefore determine point extrema from the triangulation as follows: During the triangulation give each triangle a marker "Y" if all its vertices lie on the same contour (i.e., same index) and otherwise "N". In a second step trace all contours. If for one contour all triangles on (at least) one side have the marker "Y" then the contour is a point extrema. An example is given in figure 4. More details are found in (Xie 2008).

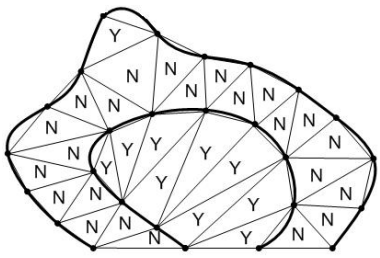

Figure 4: Examples of a point extremum (inner curve) and a normal contour line (outer curve).

6.4 Contour Graph

As explained in section 5 different types of contour graphs can be used for our algorithm. A *full graph* is constructed by connecting all point extrema with each other. If there are n point extrema the graph has $n(n-1)/2$ edges. Point extrema are represented by using a random sample point on the contour. The choice of a sample point does not influence the results of the labelling process and is only visible when rendering the contour graph for illustration purposes.

The alternative is to use a spanning tree. We use a *shortest spanning tree* constructed with Prim's algorithm (Xie 2008). A more efficient implementation would be Kruskal's algorithm which is $O(n \log n)$. As weighting function we use the Euclidean distance between point extrema. The motivation for this is the hypothesis that the shorter the distance between two point extrema the fewer line extrema (ridges and valleys) are in-between. This would make it easier to identify line extrema.

The crossings between graph edges and contour lines are computed by drawing the graph edges as lines into an off screen buffer and checking their pixel values. If the value of a pixel is non-zero then it is intersecting a contour line and the pixel value is the contour line index. Since contour lines can have loops multiple intersections are possible. Hence new intersections are only counted if the index of the intersected contour line changes.

6.5 Construction of DEM

A Digital Elevation Map with $k \times l$ pixels is constructed by computing the coordinates of each sample point with respect to the contour map and by determining for each sample point the triangle in which it lies. In order to obtain a smooth 3D rendering of the terrain it is recommended to make the sample point distance smaller than the average triangle size. Hence we loop through the triangles and determine for each triangle which sample point it contains. This can be computed in constant time since the sample points form a grid of lines parallel to the coordinate system: From the axis-aligned bounding box (AABB) and the sample distance we compute the sample points lying inside the AABB and then perform a 2D inside-outside test with the triangle edges. For each sample point we then compute a height value by performing a bilinear (barycentric) interpolation of the triangle vertex heights. Efficient formulas for this calculation can be derived by solving a linear system of equations (Xie 2008).

In order to render the DEM with common graphics and game engines we store it as a gray scale image by scaling the height values to the range $[0, 255]$. A smoother interpolation can be obtained by applying a smoothing filter to the image. Note that in this case the original contour height values would be lost and extrema would be flattened. If both precision and smoothness is important higher-order interpolation functions such as thin-plate splines and Catmull-Rom splines can be employed (Gousie & Franklin 1998).

7 Results

7.1 Examples

7.1.1 Simple Contour Map Sketched from a Real Terrain

Figure 5 (a) shows a simple contour map representing a small section of a real terrain. The contour map was sketched by architects to represent the terrain of Selinus, an extinct city founded by the ancient Greek civilisation in the southwest of Sicily. It has become the subject of considerable archaeological attention since the 19th century due to its remarkable architectural and artistic features, especially those of the Doric temples on the site (De Angelis 2003). The height labels have been added for illustration. The map has 7 point extrema, no line extrema and 21 contour lines.

Part (b) of the figure shows the results after sampling the contour lines and triangulating the sample points. Part (c) and (d) of the figure show a full graph and the shortest spanning tree connecting the point extrema. When using the full graph all contour lines can be labelled automatically after inputting the heights of the point extrema. When using the shortest spanning tree an ambiguity exist as illustrated in part (e) of the figure - one additional user input is necessary to resolve it. The ambiguity is caused because the contours along the bottom left edge of the SST form a "local valley". The valley does not continue over the entire map and hence the ambiguity can be resolved when using the full contour graph, but cannot be resolved when using the SST. The resulting DEM is shown in part (f) and its 3D visualization using two different view points is displayed in part (g) and (h) of figure 5.

7.1.2 Example 2 - Sketched Contour Map of an Imaginary Terrain

Figure 6 (a) shows a sketched contour map which was created using free hand drawings without any terrain or map information as model. Finding appropriate height values for such free hand drawings is not trivial since they must fulfil the requirements listed in subsection 3.1. Part (b) of the figure shows the height values of all point and line extrema in red colour. The contour map has 15 point extrema, 6 line extrema, and 59 contours. Note that in the centre of the contour map there are three neighbouring contour lines with a height of 300.

Figure 6 (c) and (d) show the full contour graph (105 edges) and SST (14 edges), respectively. When using the full graph the entire map is labelled using 21 user inputs (for the 15 point and 6 line extrema). When using the SST two additional user inputs are necessary which are indicated in part (a) by yellow height values. Part (e) of the figure shows the algorithm after most of the point extrema have been labelled (shown in red). The blue lines indicate not yet labelled contours. The resulting DEM is shown in part (f) and its 3D visualization using two different view points is displayed in part (g) and (h) of figure 6.

7.1.3 Example 3 - Topographic Map Data

Figure 7 (a) shows a section of a topographic map. The map was scanned in and contour lines were extracted using various image processing operators provided by the program PaintShop. Because of noise the resulting map had about 20 gaps in the contour lines which had to be fixed by hand. More advanced techniques for extracting contour lines are described in the literature (see section 2). The contour height values in part (a) of the figure were obtained by converting the contour labels in the original map from feet to meter and scaling them in order to simplify labelling and user input. Since the DEM is scaled for rendering these changes have no effect on the resulting visualization. The contour map has 24 point extrema, no line extrema, and 72 contours.

Figure 7 (c) and (d) show the full contour graph (276 edges) and SST (23 edges), respectively. When using the full graph the entire map is labelled using 24 user inputs. When using the SST six additional user

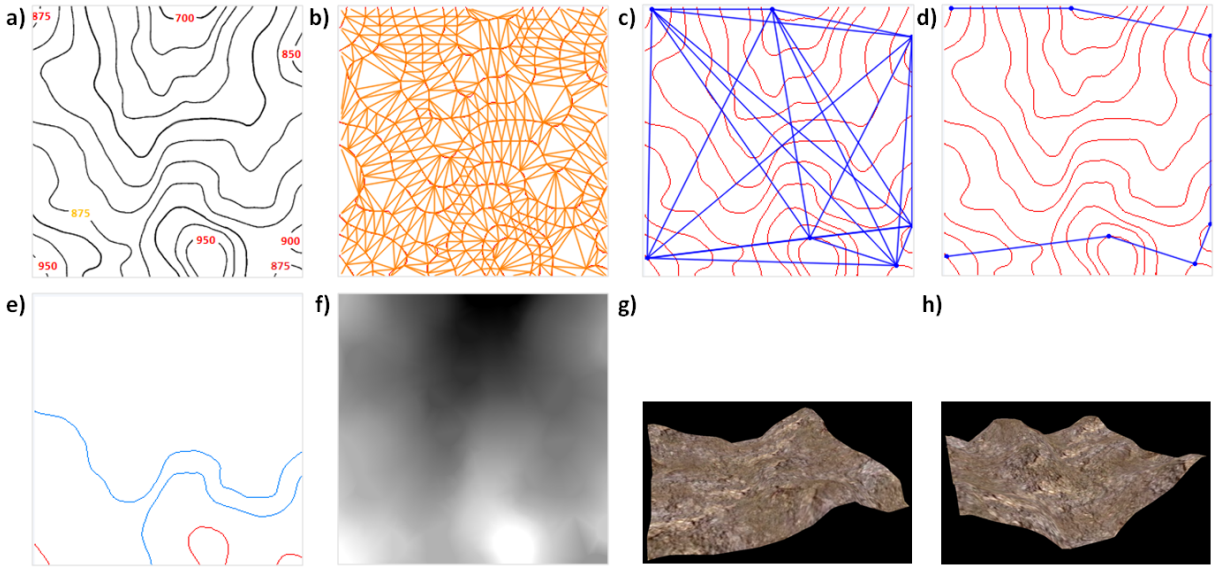

Figure 5: A sketched contour map of an archaeological site with height values (a) and the resulting Delaunay triangulation (b), full contour graph (c) and SST (d). Part (e) demonstrates that labelling using the SST results in ambiguous contours requiring an extra user input. A 3D visualization of the resulting DEM (f) using two different view points is shown in (g) and (h).

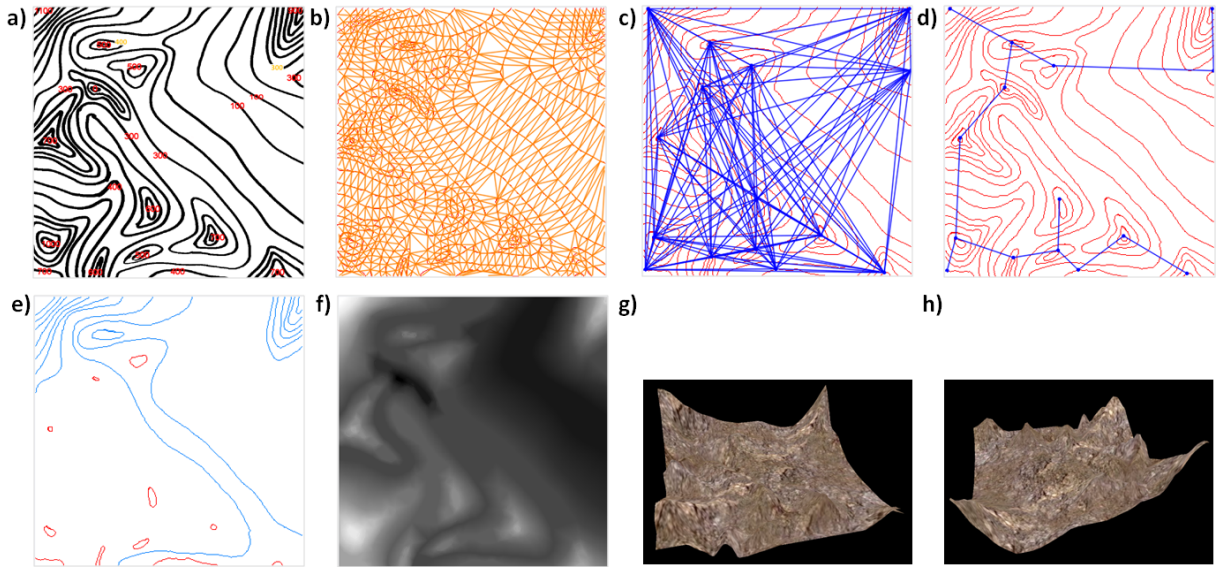

Figure 6: A sketched contour map of an imaginary terrain with height values (a) and the resulting Delaunay triangulation (b), full contour graph (c) and SST (d). Part (e) of the figure shows the algorithm after most of the point extrema have been labelled (red). The blue lines indicate not yet labelled contours. A 3D visualization of the resulting DEM (f) using two different view points is shown in (g) and (h).

inputs are necessary which are indicated in part (a) by yellow height values. Part (e) of the figure shows the algorithm after most of the point extrema have been labelled (shown in red). The blue lines indicate not yet labelled contours. The resulting DEM is shown in part (f) and its 3D visualization using two different view points is displayed in part (g) and (h) of figure 7.

7.2 Applications

The presented algorithm is useful for all applications where a contour map must be converted into a DEM and where the contour labels are not available in electronic form. We found three applications which are common in practice.

7.2.1 Sketching a Real Terrain

The first application is sketching a real terrain using contour lines as demonstrated in the first example above. While in practice many real terrains are available as GIS data, there are cases where handmade sketches are more practical and useful. Examples are terrains which have changed over time, e.g., an archaeological site which is thousands of years old, or applications where the user wants to simplify or modify the actual terrain, e.g., by adding new terrain features such as excavations representing roads or open pit mining.

Preliminary tests suggest that the interface is very intuitive if consistent height values for the overall terrain shape are available and the user only adds, deletes and modifies contours in order to model the desired landscape features. However, problems exist if no height values are given. Many users find it diffi-

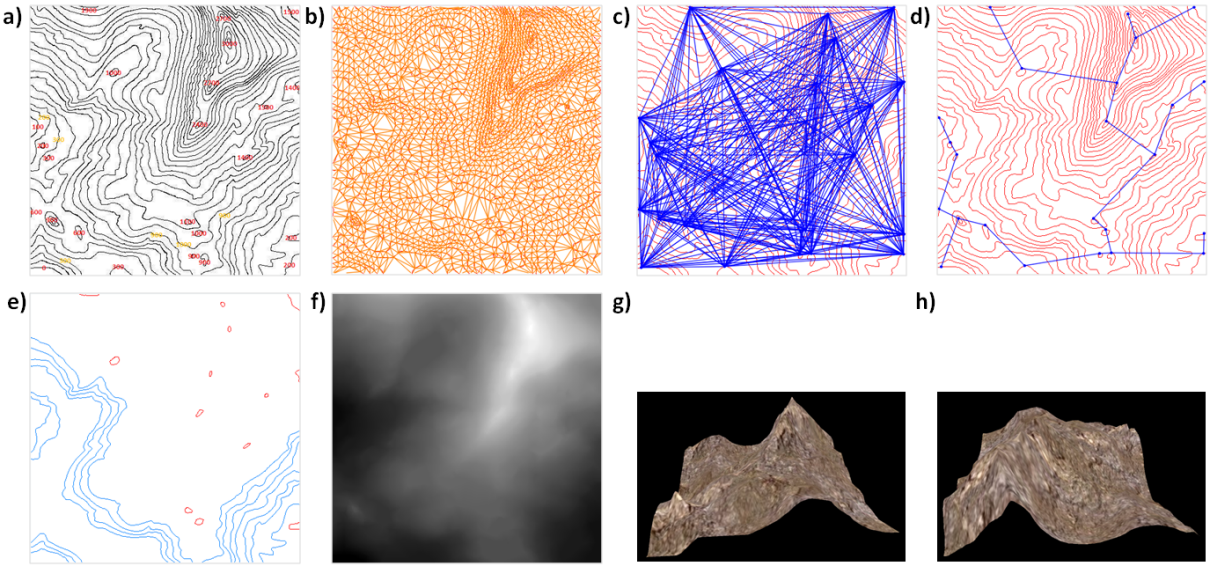

Figure 7: Section of a topographic map of a mountainous terrain. The height values have been converted from feet to meter and scaled in order to simplify labelling and user input (a). The remaining images show resulting Delaunay triangulation (b), full contour graph (c) and SST (d). Part (e) Performs the labelling using the SST results in ambiguous contours requiring an extra user input. A 3D visualization of the resulting DEM (f) using two different view points is shown in (g) and (h).

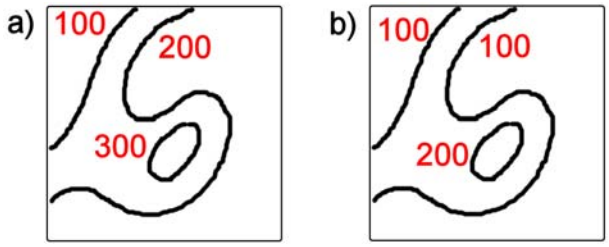

Figure 8: (a) Invalid contour labelling resulting from considering only local features. (b) Valid labelling.

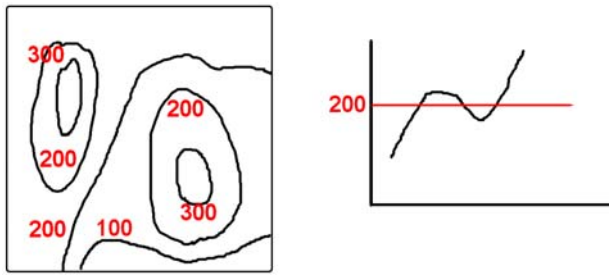

Figure 9: The labelling on the left represents an impossible configuration. The cross section on the right demonstrates that a terrain rising from 100 to 300 must intersect the 200 contour an odd number of times.

cult to determine consistent and correct height labels when using a single view of a 3D terrain.

This observation resulted from a simple user study with 6 male and 3 female participants with various levels of computer graphics experience. We showed the participants the 3D terrains from figure 5 and 6 and asked them to label the corresponding contour map. All of the nine users managed to recreate the peaks of the terrain, but often didn't notice valleys and had problems estimating relative height values correctly. This was largely due to the lack of perception of terrain features rather than the inability to understand the contour label process.

Several users created impossible sequences of height labels. The main cause were consecutive contours with the same height value. The users assigned different height values which had the effect that in other parts of the map two neighbouring contours had double the allowed height difference between them. The problem is illustrated in figure 8. The results confirm the importance of the consistency test in equation 2.

Another interesting observation was that most users performed the labelling by proceeding from the lowest to the highest contour (or vice versa). This was rather surprising since the silhouettes provided a much better indication of relative height values and hence we would have expected users to first label the contours intersecting the boundary of the map. The results suggest that in applications where the point extrema are unknown a more incremental approach might be preferable.

7.2.2 Sketching an Imaginary Terrain

The second application is the sketching of an imaginary terrain as demonstrated in figure 6. While the interface is very easy to use, problems exist with the labelling of contour lines. Without any guidelines finding a consistent labelling as in part (a) of the figure is challenging. If the point extrema are labelled first then frequently there are either too few or too many contour lines between them. While in the latter case the user can create line extrema this can lead to impossible configurations as described above. The problem is illustrated in figure 9.

Similar to the previously discussed application our experiments suggest that it is more intuitive to perform the labelling incremental. This means each time the user inputs the height value of an extremum the application performs the consistency check (equation 2) of algorithm 1 for all edges of the contour graph. This, however, is not very efficient. We are currently implementing an improved interface where the user indicates the terrain gradient by sketching arrows. Using this interface one single sketch can label all contour lines between two point extrema or line extrema. The number of necessary sketches depends

on the number of gradient changes, i.e., is equivalent to the number of edges of a subtree of the full graph connecting the extrema in the terrain.

7.2.3 Converting Topographic Map Data

The third application is the labelling of contour lines of a topographic map as demonstrated in figure 7. If the original map is consistent then a consistent labelling is always achieved as long as the correct height values for the extrema are input. Since the program automatically identifies point extrema the user only has to match their positions to the map and input the corresponding height values. Labelling line extrema is potentially more difficult since the program might display several potential line extrema as indicated in part (e) of the figure. If the user chooses a contour which is not a line extrema then one additional input will be necessary. The biggest problem is that the user might read the wrong height value from the topographic map in which case the resulting contour map will be inconsistent.

7.3 Complexity Analysis

Let the input contour image be of size $N \times N$. Each individual contour line has $O(N)$ pixels. Preliminary studies performed by us suggest that most contour images use screen space efficiently, i.e., the height steps are defined such that the contour lines are neither too sparse nor too close. That means that the number C of contour lines lies between $O(1)$ (e.g., if using a constant height step independent of image size) and $O(N)$ (e.g., height step is adjusted such that the distance between contour lines is constant independent of the image size). In practice we hypothesize that the dimension of a contour map is similar to a fractal, i.e., $O(N^k)$ with $1 \leq k \leq 2$. More studies are necessary to verify this hypothesis.

The contour tracing, sampling and number of sample points is constant in the number of contour pixels, i.e., $O(N^k)$. The Delaunay triangulation is $O(m \log m)$ in the number of sample points m, i.e., $O(N^k \log N)$.

Let n and e be the number of nodes (point extrema) and number of edges of the contour graph. For a full graph $e = O(n^2)$ and for a SST $e = O(n)$ where the computation of the SSP can be achieved in $O(n \log n)$ (Kruskal's algorithm). The number of point extrema n is bounded by the number of contour lines and the average number of pixels of an edge is $O(N)$ for a full graph, but might be smaller for a SST. Consequently an upper bound for the labelling step is $O(N^{(k+1)})$ for a full contour graph and $O(N^k \log N)$ if a SST created with Kruskal's algorithm is used. In the latter case the complexity of the total algorithm is bounded by $O(N^k \log N)$ where $1 \leq k \leq 2$ depends on the number of contour lines.

8 Conclusion

We have presented an efficient algorithm based on a graph-theoretical approach for contour line labelling and for converting contour maps into digital elevation maps. The algorithm automatically determines point extrema and uses them to create a contour graph. When using a full graph the number of required user inputs is optimal, i.e., the user only has to input height values for the extrema. All other contour lines are labelled automatically. When using a shortest spanning tree additional user inputs might be required, which in our tests were always around 10%–20% higher than the minimum number of user inputs. The advantage of using an SST contour graph is that the total algorithm complexity can be bounded by $O(N^k \log N)$ where $1 \leq k \leq 2$ depends on the number of contour lines, which can vary between $O(1)$ and $O(N)$.

The labelling algorithm can be used for contour maps obtained from digitized maps or for sketch-based terrain modelling. In the latter case it is often difficult to find consistent extrema height values and an incremental approach might be more appropriate. We have presented a simple test which the algorithm uses to verify the consistency of the contour labelling.

In future work we want to get more insight into the complexity of the algorithm, research alternative representations for the contour graph, and adapt the algorithm to make it more suitable for interactive sketch-based terrain modelling. In particular we are currently working on an application where relative heights of contours are indicated by local gradients (arrows) and classified contours have an immediate effect on the 3D terrain rendering.

References

Arens, C. A. (2002), The bowyer-watson algorithm: An efficient implementation in a database environment, Technical report, Department of Geodesy, Section GIS Technology, Delft University of Technology, Netherlands. URL: `http://www.gdmc.nl/publications/2002/Bowyer_Watson_algorithm.pdf`.

Arrighi, P. & Soille, P. (1999), From scanned topographic maps to digital elevation models, *in* D. Jongmans, E. Pirard & P. Trefois, eds, 'Proceedings of Geovision'99: International Symposium on Imaging Applications in Geology', pp. 1–4.

Dakowicz, M. & Gold, C. M. (2002), Visualizing terrain models from contours - plausible ridge, valley and slope estimation, *in* 'Proceedings of the International Workshop on Visualization and Animation of Landscape', Kunming, China.

De Angelis, F. (2003), *Megara Hyblaia and Selinous : two Greek city-states in archaic Sicily*, University School of Archaeology monographs, 55, Oxford University Press, New York.

de Boer, W. H. (2000), 'Fast terrain rendering using geometrical mipmapping'. URL: `http://www.flipcode.com/archives/article_geomipmaps.pdf`.

Duchaineau, M., Wolinsky, M., Sigeti, D. E., Miller, M. C., Aldrich, C. & Mineev-Weinstein, M. B. (1997), ROAMing terrain: Real-time optimally adapting meshes, *in* 'Proceedings of Visualization '97', pp. 81–88.

Gousie, M. & Franklin, R. (1998), Converting elevation contours to a grid, *in* 'Proceedings of the 8th International Symposium on Spatial Data Handling', pp. 647–656. URL: `http://citeseer.ist.psu.edu/article/gousie98converting.html`.

Kymer, D. J. (2009), User interfaces for the effective exploration and presentation of virtual archaeological sites, Master's thesis, Department of Computer Science, University of Auckland, Auckland, New Zealand. (to be published).

Lalonde, M. & Li, Y. (1997), Contour line extraction from color images of scanned maps, *in* 'ICIAP '97: Proceedings of the 9th International Conference on Image Analysis and Processing-Volume I', Springer-Verlag, London, UK, pp. 111–118.

Losasso, F. & Hoppe, H. (2004), 'Geometry clipmaps: terrain rendering using nested regular grids', *ACM Transactions on Graphics* **23**, 769–776.

Maia, M. A. G. M. & Átila L. F. Xavier (1996), 'A semiautomatic method for assigning elevation in contour maps', *IEEE Transactions on Knowledge and Data Engineering* **8**(4), 596–603.

Marita, T. (n.d.), 'Border tracing algorithm'. URL: `http://users.utcluj.ro/~tmarita/IPL/L6/PI-L6e.pdf`.

Peterson, S. (1998), 'Computing constrained delaunay triangulations in the plane'. URL: `http://www.geom.uiuc.edu/~samuelp/del_project.html`.

Rivas, A. M. & de la Fraga, L. G. (2005), Terrain reconstruction from contour maps, *in* 'Proceedings of the International Workshop on Visualization and Animation of Landscape', Mexico City, pp. 167–175.

Rognant, L., Planes, J. G., Memier, M. & Chassery, J. M. (2001), Contour lines and DEM: Generation and extraction, *in* 'Proceddings of First International Symposium on Digital Earth Moving (DEM 2001)', Lecture Notes in Computer Science, Springer-Verlag. URL: `http://www.springerlink.com/content/q962mm5bf2c43byd/fulltext.pdf`.

Schiffman, H. R. (1996), *Sensation and Perception: An Integrated Approach*, 4^{th} edn, John Wiley & Sons.

TERRAINMAP.COM - Digital Elevation Modeling Journal (n.d.), 'Contour line extraction from a difficult topo map using BLACKART'. URL: `http://www.terrainmap.com/rm36.html`.

U. S. Army Corps of Engineers and American Society of Civil Engineers (1996), *Photogrammetric Mapping*, ASCE Publications, New York.

Wikipedia (n.d.), 'Contour map'. URL: `http://en.wikipedia.org/wiki/Contour_line`.

Xie, X. (2008), Automatic 3d terrain modelling from topographic map drawings, 780 project report, Dept. of Computer Science, University of Auckland, New Zealand.

Xin, D., Zhou, X. & Zheng, H. (2006), 'Contour line extraction from paper-based topographic maps', *Journal of Information and Computing Science* **1**(5), 275–283.