Multi-level representation of terrain features on a contour map

Eric Guilbert

Abstract

Contour lines are important for quantitatively displaying relief and identifying morphometric features on a map. Contour trees are often used to represent spatial relationships between contours and assist the user in analysing the terrain. However, automatic analysis from the contour tree is still limited as features identified on a map by sets of contours are not only characterised by local relationships between contours but also by relationships with other features at different levels of representation. In this paper, a new method based on adjacency and inclusion relationships between regions defined by sets of contours is presented. The method extracts terrain features and stores them in a feature tree providing a description of the landscape at multiple levels of detail. The method is applied to terrain analysis and generalisation of a contour map by selecting the most relevant features according to the purpose of the map. Experimental results are presented and discussed.

Keywords: Contour graph, terrain analysis, object generalisation, feature tree

1 Introduction

The contour line is one of the most fundamental elements of a topographic map. Contours are one of the most appropriate method for displaying relief at large and medium scales. They also assist the user in identifying morphometric features such as valleys and ridges and in interpreting the map [11]. Indeed, it is the set of contour lines, not the individual lines themselves, which depict the major features of a landscape and retain its intrinsic character.

Most of the research on feature characterisation has been concerned with the analysis of digital terrain models [4, 8] as these provide more accurate local information (slope, aspect, curvature). Existing work on terrain feature identification from a contour map utilises contour trees representing spatial relationships between contour lines based on their inclusion. They can be built from the terrain model [27] or from the contours directly [13, 7, 5]. As shown in these papers, contour trees are useful for reasoning as they can assist the user in identifying and analysing terrain features. However, they do not fully take into account the different possible levels of description of the terrain since they focus on features located at the highest level of detail [22, p. 180].

The focus of this paper is to build up a hierarchical description of terrain features from the contours. At the end of the process, a feature tree is provided where each feature is defined by a set of contours. The tree allows a representation of the terrain at different levels of detail and is directly related to the complexity of the landscape, providing a qualitative description of the landscape. Geometric properties of the surface (slope, curvature) or the contours (distance between or length of contours) are not

involved and features are described based on topological relationships (adjacency and inclusion) ensuring a simple and robust process.

This paper also proposes a tool enabling qualitative terrain description useful for route planning and location (where landforms can be described according to different perspectives) and map generalisation. Currently, much work in contour line generalisation relates to cartographic generalisation where only aesthetics and legibility are considered [10, 15, 17, 19]. In some applications, features must be represented on the map in accordance with their meaning. For instance, in nautical chart generalisation, features are selected in accordance with the risk they represent (e.g. reef) or their relevance to navigation (e.g. fairway). Therefore, features must be identified and classified so that contours can be generalised with regard to the features they relate to.

The article is organised as follows. The next section reviews previous work related to contour trees and topological data structures for surfaces and to the description of landscapes at multiple levels. Section 3 describes an iterative process for extracting features at different levels. In section 4, the method is applied to terrain analysis and nautical chart generalisation and results are discussed. Conclusions and directions for future work are presented in the final section.

2 Related work

2.1 Contour trees and graphs

The first data structure addressing the description of surface topology was the Reeb graph as reported in [22, p. 8]. The Reeb graph is obtained from a surface as a topological quotient space where all the points having the same elevation and lying in the same connected component are equivalent [28]. Equivalence classes correspond to contours and nodes of the Reeb graph, which are the critical points (peaks, passes, pits) of the surface, represent the topological transitions between the contours.

If the surface is defined by a set of contours at arbitrary elevations, building up a topological graph is not based on the detection of critical points which can lead to numerical instability but on the spatial relationships between the contours. The resulting graph is called a contour tree. The structure is identical to a Reeb graph [26] since topological transitions between contours are also recorded (Figure 1). Contour trees are used for terrain representation and in computer visualisation for segmenting or rendering multi-dimensional data [3]. The most commonly used technique for building a contour tree is based on the containment relationship. If one contour is contained by another then that contour is a descendant. This simple approach has been applied in [7] where the map is partitioned into different areas, each containing a contour tree. The author uses this structure to interpolate terrain information such as elevation or gradient at a given point from neighbouring contour lines. Work taking a different approach based on containment relations is [13] where a single contour tree is built for the whole map. The method starts from the lowest elevation contour defining the root node recursively creating the contour tree in a depth-first fashion. The authors also consider the extraction of peaks and pits from the contour tree by considering the number of children of a node: a peak or a pit is defined by a starting node where all its descendants have only one child, excepted the last which has none (Figure 1).

The containment method is very simple but some limitations have motivated the development of different approaches. One difficulty for feature characterisation is that all contours are considered closed, thus limiting its application to specific cases. Open

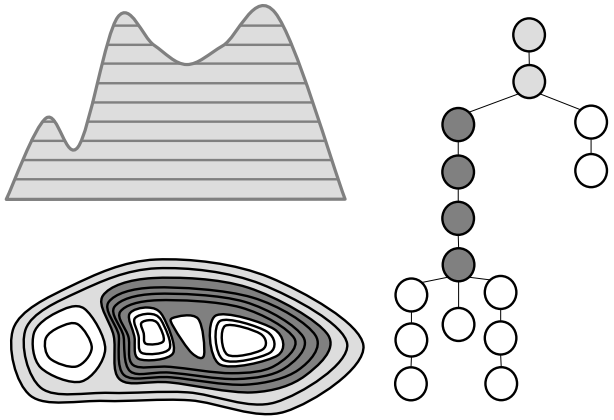


Figure 1: Contour tree corresponding to a set of closed contours. Branches with white nodes are peaks or pits. Nodes with several children are passes.

contours may be considered closed by the border of the map but it is not always possible to define which side is the interior or the exterior of a contour.

Other methods build the contour tree by considering contours in close proximity. Contours are considered adjacent if their growing regions [23] or their Voronoi regions [5] share a common boundary. The disadvantage of this approach is that the construction of the tree is based on the relationship between region boundaries, a relationship which does not necessarily correspond to the containment relationship. In [5], the algorithm detects and corrects such problems so that results are always consistent.

The Voronoi region based method can provide a contour tree for any kind of contour set, including open contours, but the changes in slope cannot be identified from the tree as elevations are not considered. Contours characterising one feature can be classified as descendants of a contour that is part of an adjacent feature. This occurs, for example, when a change of slope is represented by a series of open contours. Indeed, the contour tree representation is a limitation as it creates an artificial hierarchical structure of the landscape. It limits relationships between contours to those of containment only and fails to represent adjacency between open contours.

Another approach to the storing of relationships between contours is the use of a contour graph. Nodes are contours and edges represent adjacency relationships. Two contours are adjacent if a line can be drawn that connects the two contours and intersects no other contour [20]. Elevation is not required to build the graph. The structure applies to any kind of contour map containing open and closed lines (Figure 2c) and has been used for plane navigation [21] and vectorisation of map images [16]. The dual of the contour graph is the inter-contour region graph where contours are edges and inter-contour regions are nodes of the graph (Figure 2d). Both structures are equivalent since one is the dual graph of the other but the contour graph is easier to implement [16].

Another data structure storing topological relationships between regions is the extended Reeb graph [1] where equivalence classes are regions bounded by contour lines. If the contour map is defined by an arbitrary set of contours, the extended Reeb graph is identical to the inter-contour region graph. The structure is more reliable than the con-

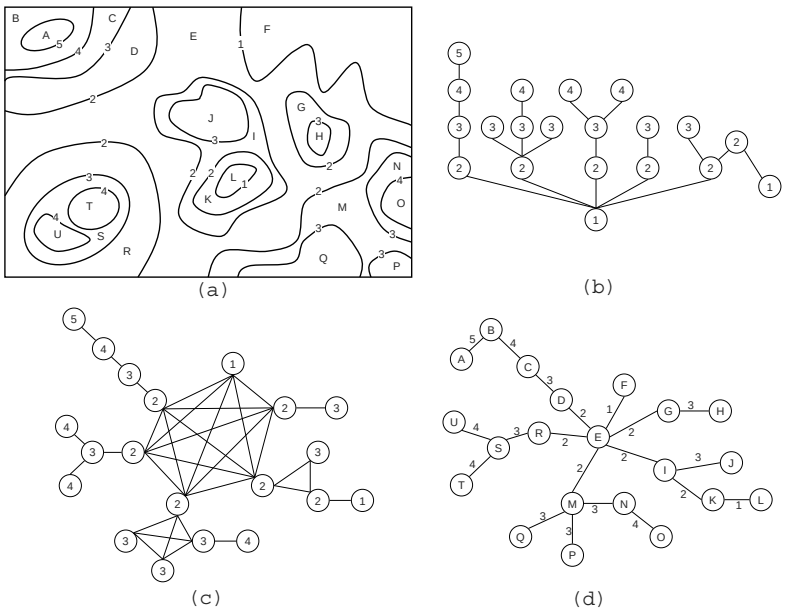


Figure 2: Contour map (a) with its corresponding contour tree (b), contour graph (c), region graph (d). Numbers indicate elevation. Letters are inter-contour regions.

tour tree or the Reeb graph as a region boundary can be composed of several contours and it better represents the topology between areas of the surface.

2.2 Multi level description of landscapes

Although describing a landscape at different levels has been tackled for Digital Elevation Models (DEM), topological structures such as contour graphs and Reeb graph do not address multiple representation of features [22, p. 180]. The definition of landforms depends on the scale of observation and the interpretation of the user [9]. Existing work relates to the identification of specific landforms [4], the definition of an ontology to describe these landforms [24], or the representation of terrain morphology of a DEM at multiple resolutions [8].

Six morphometric classes of terrain can be represented on a DEM [29] (Figure 3). Multi-level description depends on the scale of measurement or visualisation [9] and a point may belong to different morphometric classes depending on the size of the filter computing slope variations. In Figure 4, if the location is at the bottom of the pit marked G, use of a small size filter leads to its definition as a pit G. With a larger filter, the location is designated as belonging to peak B and with an even larger filter as belonging to peak A. Precision depends on the number of filters and their sizes set a priori.

Contour maps do not provide as much terrain information as a DEM but, although local variations of slope cannot be computed accurately, they can still provide qualitative information on terrain structure and features identified by sets of contours. Currently, contour trees can identify eminences and depressions at the highest level but not at lower levels. A less restrictive definition should be considered so that features at

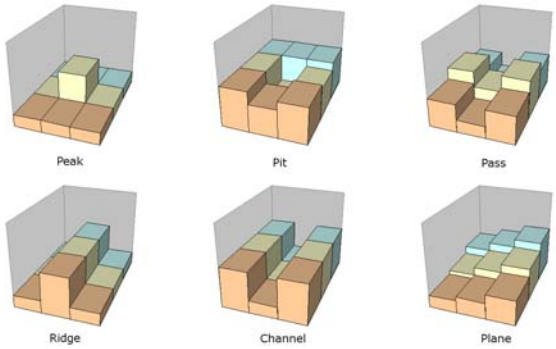


Figure 3: Morphometric classes of terrain.

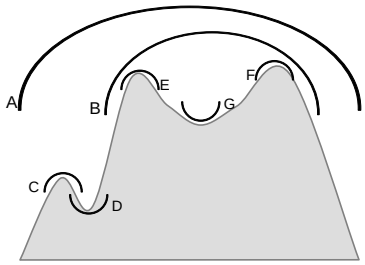


Figure 4: Morphometric description at different scales, using the example of Figure 1. At a small scale, feature A is defined as one structure. At a larger scale, feature B is identified. At the largest scale, C, D, E, F, G are detected.

different levels can be identified. One possibility would be to define a feature such that one contour delineates its boundary and all its descendants in the tree. For example, in Figures 1 and 4, contours contained in feature E also belong to features B and A. However, contour trees fail to represent correct relationships between contours. A feature boundary cannot be defined by two or more contours and larger features may not be correctly portrayed: a channel going across the map cannot be represented. Feature characterisation would be possible from the contour graph or the inter-contour region graph where relationships are correct, but feature at different levels cannot be shown on such a graph. Therefore, a new structure, the feature tree, defined from the contour graph is introduced in the next section.

3 Construction of the feature tree from a contour graph

3.1 Definition of features on a contour map

On a contour map, a region is defined by a set of contours and is the connected area bounded by these contours. The smallest regions that can be identified on a contour map are the inter-contour regions. Larger regions can be identified by the merging together of adjacent inter-contour regions.

The most obvious landforms identified on a contour map are eminences [18] and depressions. An **eminence** is defined as a region where all the inner contours are higher than the boundary contours. In a similar way, a **depression** is a region where all the inner contours are lower than the boundary contours. In order to allow comparison, the boundary contours must all be at the same elevation. Examples of eminences from Figure 2 are regions J, U and ABCD (the region formed by inter-contour regions A, B, C and D). Region EF is a depression as its edge elevations are all equal to 2 and it contains a lower contour. On the opposite, E is neither an eminence nor a depression as edges E-F and D-E are not at the same elevation.

Distinction between **peaks** and **ridges** and between **pits** and **channels** is made by checking the number of contours on the boundary. A region with only one boundary contour is a peak or a pit with its highest or lowest point at the centre. A region with two or more boundary contours is a channel or a ridge and its orientation is given by its medial axis. For instance, in Figure 2, KL is a pit and EF is a channel. In some cases, a region obtained by merging several regions may form a leaf of the graph and be neither an eminence nor a depression as its boundary contour is neither the highest nor the lowest. This feature is designated as a **mixed feature**. Region IJKL in Figure 2 is an example of a mixed feature.

In a DEM, a **pass** is a point defining a local minimum in one direction and a local maximum in the other. On a contour map, a pass would be located in a lower (respectively higher) region joining two higher (respectively lower) regions. In the graph, the pass is a region identified as connecting two or more features of the same type. Examples of passes from Figure 2 are regions S and M.

A **plane** is a type of terrain with a regular slope. On a contour map, it is a region delineated by two boundary contours of different elevations where all the inner contour elevations are within the boundary elevations. They correspond to regions representing hillsides such as BCD in Figure 2.

Eminences, depressions and mixed features should be as large as possible, i.e. if a feature is adjacent to a plane, both feature and plane can be merged into a larger feature of the same class. For example, in Figure 2, region A is not considered as a peak as it is part of the larger peak ABCD.

Following these definitions, all features from the region graph can be identified and arranged in a feature tree based on their inclusion relationship. An example of a feature tree obtained from the contour map of Figure 1 is presented in Figure 5 right. Only eminences and depressions are shown. Figure 5 left represents the inter-region graph equivalent to the contour tree of Figure 1. Features A, B, C, E and F are obtained by merging together some inter-contour regions. D is not stored as it is a pass connecting A, C and B.

The next section presents a method for building the feature tree from a set of contours. As features are identified by grouping regions together, the method first requires the construction of the inter-contour region graph and proceeds by successively merging regions at one level to form larger regions at the next level.

3.2 Construction process

The hierarchical structure of the terrain is represented by a feature tree containing all the eminences, depressions and mixed features. Feature extraction from an initial region graph (Figure 6a) is an iterative process starting with features at the highest level of detail (Figure 6b) and moving towards larger features until no new feature can be

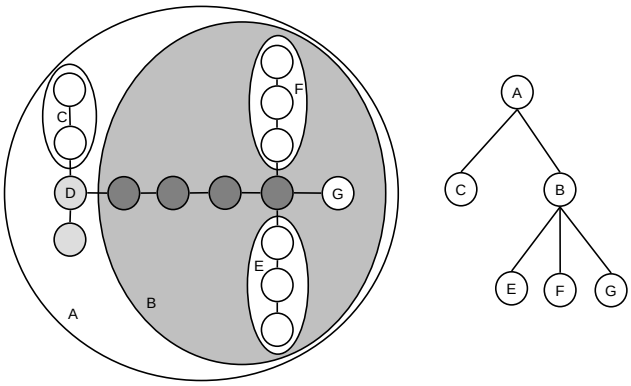


Figure 5: Left: Region graph from Figure 1 contour map with features at different levels. Right: Feature tree extracted from the region graph.

extracted (Figure 6g). At each step of the process three operations are performed in a sequence:

- obtain the greatest extent of each feature by merging it with adjacent planes;
- copy the new features into the feature tree;
- move to the next level by aggregating features with adjacent regions.

When a region is connected to several features, two cases can arise. First, the region is a pass connecting features of the same type and the pass is aggregated with the features to form a larger feature. For example, pass S and peaks T and U of Figure 6b are aggregated into a larger peak STU (Figure 6c). In the second case, the region connects features of different types (for example, region E of Figure 6d). The order in which the aggregations are performed leads to different features: E can be aggregated with F to form a depression or with other features to form an eminence. Aggregation of the region and its features involves several steps according to the different cases. This process may result in storing passes (such as IJ in Figure 6d) in the feature tree during the intermediate steps. These spurious features are removed in a last stage after the tree has been built as they can only be characterised as passes once features at the next level have been added to the tree.

The merging process applied to features and planes is described in the next section 3.2.1. The feature aggregation process, with the classification of regions in different cases is detailed in section 3.2.2. Third, spurious feature removal is described in section 3.2.3. Finally, the algorithm summarising the whole process is described (section 3.3).

3.2.1 Region merging

As mentioned in section 3.1, a feature should be of the largest possible extent and therefore should be merged with its adjacent planes. In the region graph, merging consists of collapsing the edge joining two adjacent regions. Two regions are merged if:

- they are adjacent planes oriented in the same direction. The merged region is another plane (regions B, C and D of Figure 6a).

- they consist of one feature and a plane and the new feature belongs to the same class (regions G and H or K and L of Figure 6a).

The merging process is repeated until no more merging can be performed. New features are copied to the feature tree. The result does not depend on the order in which the regions are merged. Figure 6b shows the region graph and feature tree after the first merging step. The features which are first identified correspond to the leaves of the feature tree. The merging algorithm is detailed in algorithm 1.

Algorithm 1 *The procedure **merge(R)** merges those neighbouring regions of the graph which have no more than two neighbours and have the same slope direction.*

```
Input: a region graph R
Output: the region graph R after merging
Begin
   While merging
      merging = false
      For each edge e of R
         rl = e.leftRegion
         rr = e.rightRegion
         If rl.neighbours ≤ 2 and rr.neighbours ≤ 2
         and rl.slope(e) == −rr.slope(e)
            merging = true
            e.collapse
         End If
      End For
   End while
End
```

3.2.2 Region aggregation

The process consists of aggregating features such that the aggregated region is a new leaf describing a feature at a lower level of detail. The operation is done by collapsing the edges between the features and the region. Candidate regions for aggregation are regions r from the graph for which all adjacent regions but one are leaves. The region which is not a leaf is the one connecting r to the rest of the graph by the edge which is the base of the region as it encloses the subset formed by r and its leaves. The procedure for candidate region detection is detailed in algorithm 2. In Figure 6b, I, M and S are candidate regions for aggregation; E is not.

Algorithm 2 *The function **M = candidate(R)** returns the list of regions of the graph for whom all but one neighbour are leaves.*

```
Input: a region graph R
Output: the list M of candidate regions
Begin
   Set M an empty list of regions
   For each region r of R
      counter = 0
      For each neighbour n of r
         If n is not a leaf
            r.base = r.edge(n)
```

```
            ++counter
         End If
      End For
      If counter == 1
         M.push(r)
      End if
   End For
End
```

To order the aggregation, three cases are considered. They are based on the elevation of the edges connecting r with its leaves and with the elevation b of the base. Indeed, as r is an inter-contour region, only two different elevations are possible. The elevation differing from b is denoted z. Algorithm 3 details the method for classification of candidate regions into three categories. A region r can be classified as:

- a *pass*: all the edges of the leaves are at the same elevation z. Region r is a pass connecting all the leaves to the base. Examples of passes are regions S and M of Figure 6b.
- *undefined*: edges of the leaves are at different elevations. There is no change of slope between the base and the leaves at elevation z but there is a change of slope between the base and features at the same elevation. Region I of Figure 6b and region E of 6d are undefined.
- a *level region*: All the edges are at the same elevation b as the base so that there is a change of slope between the base and the features. Regions EF and IJ of Figure 6e are examples of a level region.

Algorithm 3 *The procedure* ***classify(M, M_{pass}, M_{undef}, M_{level})*** *takes the list of candidate regions and builds three lists corresponding to the three cases.*

```
Input: a list of regions M
Output: three lists of regions M_pass , M_undef , M_level
Begin
   Set M_pass , M_undef , M_level three empty lists of regions
   For each region r of M
      count1 = 0
      count2 = 0
      For each edge e of r
         If e ≠ r.base
            If e.elevation == r.base.elevation
               ++count2
            Else
               ++count1
            End If
         End If
      End For
      If count2 == 0
         M_pass .push(r) ' r is a pass
      Else If count1 == 0
         M_level .push(r) ' r is a level region
      Else
```

```
            M_undef.push(r)  ' r is undefined
        End If
    End For
End
```

If r is a pass, features are aggregated to r into a new feature f. Feature f is their parent in the feature tree (passes M and S of Figure 6c).

A level region can be seen in two different ways. It can be a channel or a ridge connecting other features at the same level. In that case, r is a feature and must appear in the feature tree at the same level as all its adjacent features (feature EF in Figure 6g). It can also be a pass connecting the leaves at a higher level. In that case, r with its leaves, forms a larger region whose descendants are leaves of r. For example, the mixed feature IJKL in Figure 6g is obtained by aggregating level region IJ with its leaf KL.

An undefined region r cannot be a feature as its edges are at different elevations but it can be aggregated with other features at elevation z to form a level region c connecting those other features with the base region: level region IJ is obtained by aggregating I with its leaf J in Figure 6c and level region EF is obtained by aggregating E with its leaf F in Figure 6e. The z features are descendants of c which is at the same level as all the b features. At this stage, it is not possible to know if c is a pass as this depends on whether or not c is contained by a larger region. Therefore c is added to the feature tree and whether it should be removed is checked at a later stage once the whole tree has been built. Features which must be removed are denoted as spurious.

As mentioned above, the order in which candidate regions are aggregated with their leaves influences the feature tree. Passes are treated first as they create no ambiguity. Undefined regions are also aggregated with their leaves at elevation z in order to transform them into level regions. Only level regions need to be ordered so that bigger features are given more importance and are placed closer to the root of the tree.

The height of a region is defined by the difference between its lowest and highest contours. If a level region is connected to a higher leaf, it is given more importance than a region connected to a smaller leaf and it should be aggregated later so that the high leaf appears at a lower level in the tree. Therefore, level regions are sorted based on the height of the highest adjacent leaf and the level region with the smallest value is aggregated with its leaves first.

Algorithm 4 *The procedure* ***aggregate(R, M_{pass}, M_{undef}, M_{level})*** *aggregates the candidate regions with their neighbouring leaves according to the different cases.*

```
Input: the region graph R
       the lists of candidate regions M_pass, M_undef, M_level
Output: the region graph R after aggregation
Begin
   If M_pass ≠ ∅ or M_undef ≠ ∅
      For each region r of M_pass
         ' Aggregate pass r with its leaves
         For each neighbour n of r
            e = r.edge(n)
            If e ≠ r.base
               r = r.aggregate(n)
            End If
         End For
```

```
      End For
      For each region r of M_undef
         ' Aggregate r with leaves at elevation z
         For each neighbour n of r
            e = r.edge(n)
            If e.elevation ≠ r.base.elevation
               r = r.aggregate(n)
            End If
         End For
      End For
   Else
      For each r of M_level
         Set r.h height of highest neighbour of r
      End For
      Set r the region with smallest h
      ' Aggregate level region r with all its leaves
      For each neighbour n of r
         e = r.edge(n)
         If e ≠ r.base
            r = r.aggregate(n)
         End If
      End For
   End If
End
```

After aggregation, the merging process is repeated in order to extract new features and insert them into the feature tree. The whole process is iterated (Figures 6b-g) until features at the last level of the feature tree form a whole partition of the map (Figure 6g). The whole map includes one more feature which is the root of the tree. This feature cannot be characterised because features are defined by comparing their edges. With only one feature, characterisation depends only on the user's interpretation.

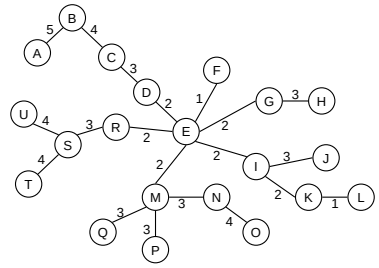
3.2.3 Removal of spurious features

During the construction stage, undefined regions are aggregated to form level regions. If the level region is a pass, it is spurious and should be removed from the feature tree. It corresponds to the case where the level region with its neighbouring features form a partition of their parent. For example, level region IJ and feature KL form a partition of feature IJKL in Figure 6g. After removal, feature IJKL contains only two features KL and J.

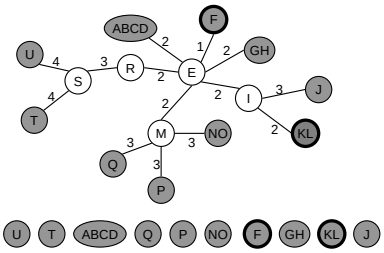
If the level region is a ridge or a channel, it is kept in the feature tree. If this region has only one descendant which is of the same type (eminence or depression), this descendant is spurious since its extent is not maximum. This occurs with features EF and F in Figure 6g. F has been identified as a depression inside region EF which is also itself a depression. The final feature tree after spurious feature removal is shown in Figure 6h.

3.3 Algorithm of the whole process

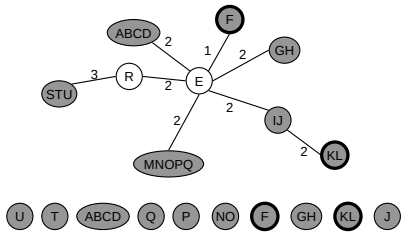
The whole process, from the initial contour graph to the final feature tree is summarised below. Algorithm 6 constructing the inter-region graph is presented in Appendix A.



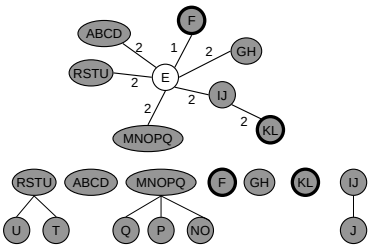
(a) Initial region graph.



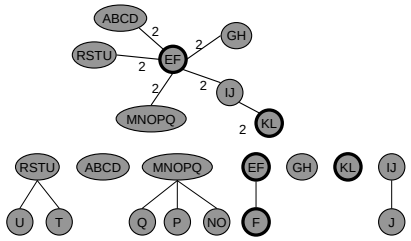
(b) ABCD, GH, KL and NO are obtained by merging. Candidate regions for aggregation are passes M and S and undefined region I.



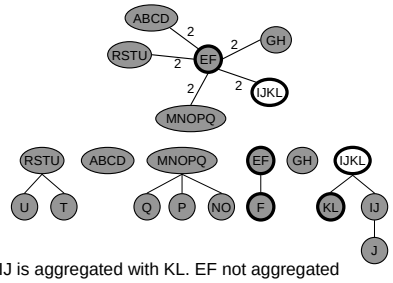
(c) Aggregation of S and M with their leaves. I is aggregated with the higher leaf J.



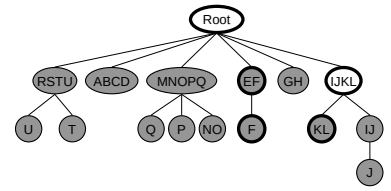
(d) RSTU is obtained by merging. RSTU, MNOPQ and IJ are added to the feature tree. Candidate regions for aggregation are crater IJ and undefined region E.



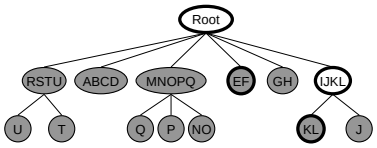
(e) Aggregation of E to form the crater EF. No merging needs to be done. EF is added to the feature tree. Candidate regions are craters EF and IJ.



(f) IJ is aggregated with KL. EF not aggregated because it connects higher features than IJ.



(g) Feature tree at the end of aggregation process. F and EF are both depressions so F is spurious. IJ is a pass between KL and J and is spurious.



(h) Feature tree after removal of spurious features.

Figure 6: Construction of the feature tree from the inter-contour region graph of Figure 2. Grey nodes with thin outline are eminences, grey nodes with thick outlines are depressions. White nodes with thick outline are mixed features. Figures from b to f show the evolving states of the region graph and the feature tree during the merging and aggregation process. Figures g and h represent the feature tree before and after removal of spurious features.

An alternative algorithm constructing the region graph from a DEM is presented in [1]. Output is the feature tree with features classified as peaks, ridges, pits, channels and mixed features.

Algorithm 5 *Feature tree construction*

```
Input: a set of contour lines
Output: feature tree F
Begin
   Construction of the contour graph
   Construction of the inter-contour region graph R ' algorithm 6
   Set F an empty feature tree

   ' Construction of the feature tree
   merge(R) ' algorithm 1
   Add new features to F

   While regions at last level of F do not form
   a complete partition of the map

      M = candidate(R) ' algorithm 2
      classify(M, M_pass, M_undef, M_level) ' algorithm 3
      aggregate(R, M_pass, M_undef, M_level) ' algorithm 4

      merge(R) ' algorithm 1
      Copy new features of R to F
   End While

   ' Removal of spurious features
   For each feature f of F
      If f is adjacent to all its sister features
         Remove f from F
      End If
      If f is the only descendant
      and belongs to the same class as its parent
         Remove f from F
      End If
   End For
End
```

3.4 Feature tree simplification

Apart from the analysis of terrain morphology and its complexity, the feature tree can also be used to perform simplification operations on the contour map, mainly deleting features which are considered not relevant. Feature deletion may be performed based on semantic meaning (type of feature), spatial relationships in the feature tree or geometric criteria (elevation, height or spatial extent of a feature). If a feature is deleted, all its contour lines are removed from the map. A feature can be deleted in different ways; for example by simply deleting all its contours or by aggregating the feature with a neighbouring feature sharing the same parent.

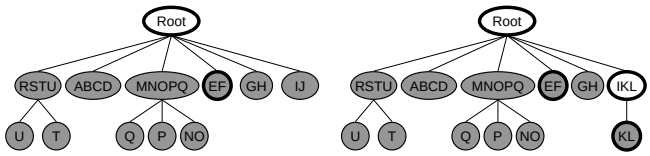


Figure 7: Left: Feature tree after removing feature KL. Right: Feature tree after removing feature J.

Removal can trigger two kinds of change: a change of class for ascendant features and removal of adjacent features. Firstly, an ascendant feature which was a mixed feature may become an eminence or a depression. This is checked by comparing the edges of the ascendant feature with its inner contours. Secondly, if after removal the ascendant feature contains only one feature of the same class, the smaller feature becomes spurious and is removed from the feature tree. Its contours are not deleted but become inner contours of the larger feature. For example, in Figure 7 left, feature KL was removed. Feature IJKL, which was a mixed feature, becomes a peak containing another peak J so that J is removed from the tree.

4 Results

The algorithm was implemented in C++ with CGAL [6] used for the construction of the contour graph. Results are presented for a set of contour lines representing a coastal area with relatively smooth terrain below sea level and more variations for the terrain above sea level. In Figure 8, contour lines are drawn at vertical intervals of five metres. The digital elevation model was provided by the Hydrographic and Oceanographic Service of the French Navy (SHOM[1]). The algorithm was also applied to other sets of contours of different sizes up to a maximum of 1500 contours and 500 features. Construction of the contour graph and the region graph are obtained immediately. Feature tree construction is directly related to the number of features. As no geometric operation is performed, the data resolution or the number of points have no influence on the result but the number of features does depend on the vertical interval between the contours: the smaller the interval, the more precise the description of the terrain. In the worst case tested, feature tree generation took a few seconds. The algorithm was also tested on singular cases and is robust when adjacent contours touch at one point: since polygons bounding the regions are not explicitly computed, occurrence of non simple polygons does not affect the process.

4.1 Identification of morphometric features

The corresponding feature tree is presented in Figure 9. Features can be displayed at different levels of detail. At level 1, three features are identified. These consist of the channel in the middle of the map and the peaks on each side (Figure 10). The depth of the feature tree is directly related to the complexity of the terrain. The node on the left of the tree corresponds to the peak at the right hand side of the map where the terrain is much more complicated. The tree on this side goes deeper (down to level 10) than it does for the left hand peak, which is described in five levels of detail.

[1] http://www.shom.fr

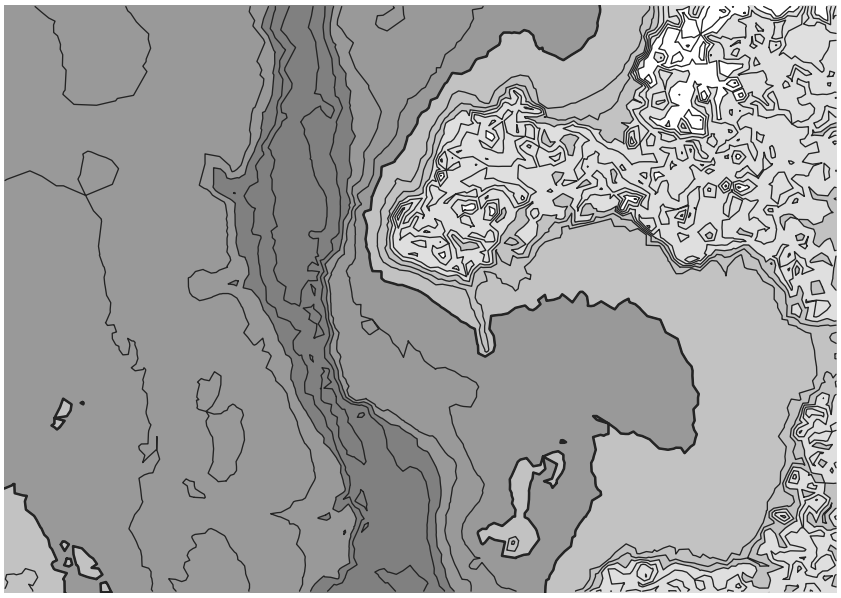


Figure 8: Contour map of a coastal area. Vertical interval is equal to five metres, the deeper the area, the darker the colour. Thick contour line is the zero metre isobath.

During the aggregation process, features inside the left hand peak and inside the channel are quickly aggregated together but the peak and the channel still remain as separate features. Indeed, the channel is a candidate region for aggregation corresponding to a level region but as the left peak is too big a feature, in comparison with other features in the right hand part of the map, it cannot be aggregated with the channel before all the features inside the right hand peak are aggregated. It is also worth noticing that some features in the right hand peak can include eminences and depressions and not be classified as mixed features. This is because the base of the larger feature is below or above all other inner contours.

Features at the highest level of detail correspond to features which do not contain any smaller details and are the leaves of the tree (Figure 11). This level of representation extracts the same features as does the contour tree presented in [13] where multi-level representation is not considered.

4.2 Application to contour map generalisation

As mentioned in section 3.4, the feature tree can be used to perform generalisation. Generalisation operators can be applied depending on characteristics highlighted by the terrain features. Here, only selective omission has been applied to remove features with their contours.

4.2.1 Reduction of complexity

The first example of simplification consists in reducing the complexity of the map by fixing a maximum level of detail. Features located close to the root are the most relevant as they represent the biggest features. In Figure 12, features of level 6 and

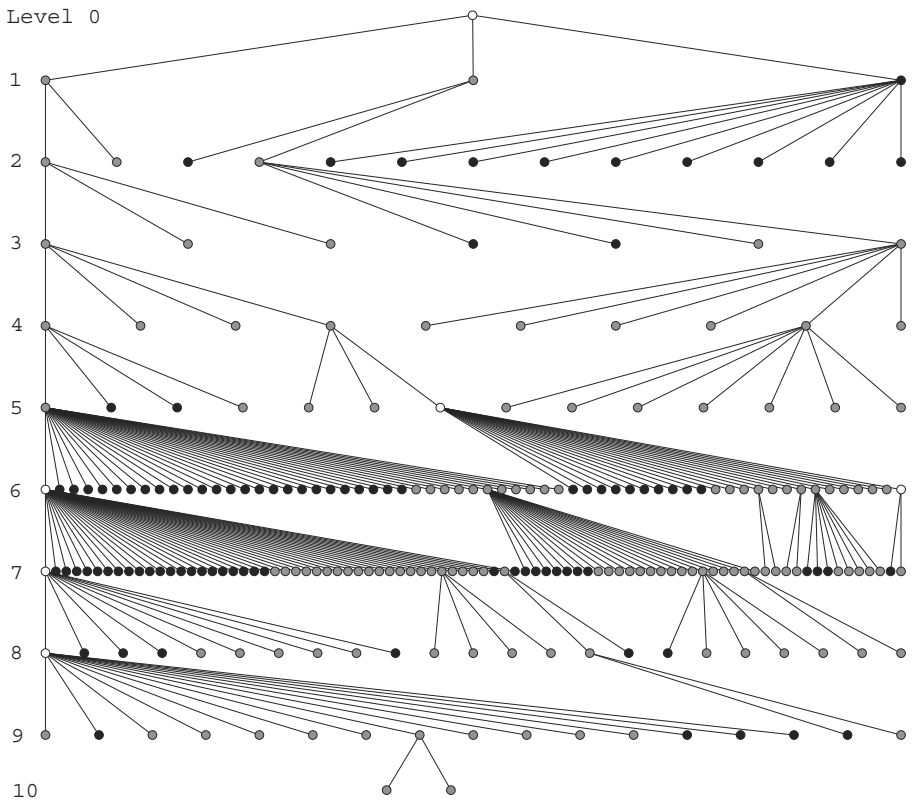


Figure 9: Feature tree of the contour map. Black nodes: depressions, Grey nodes: eminences, White nodes: mixed features. Numbers indicate levels.

above have been removed with their contours. The feature that was mixed in level 5 of the feature tree now appears as an eminence. Such an operation is useful to show the spatial extent of the main features composing the landscape and provides a more uniform distribution of features over the map.

4.2.2 Generalisation of isobathymetric lines for nautical charts

A more common operation is the removal of features which do not fit some given criteria related to the purposes of the map. On a nautical chart, in order to ensure safety of navigation, isobaths featuring underwater peaks or ridges such as reefs must be maintained while isobaths featuring pits may be removed from the chart. The importance of a feature is therefore defined by its class (eminences are more important than depressions), its area (features which are too small and not hazards may be removed) and its location (underwater features which are useful for navigation must be preserved; on land, only salient features that are visible from afar may be kept). Figures 13 and 14 illustrate this kind of generalisation where pits below sea level are removed if their areas are smaller than a given threshold and, on land, only peaks of sufficient height and area are kept.

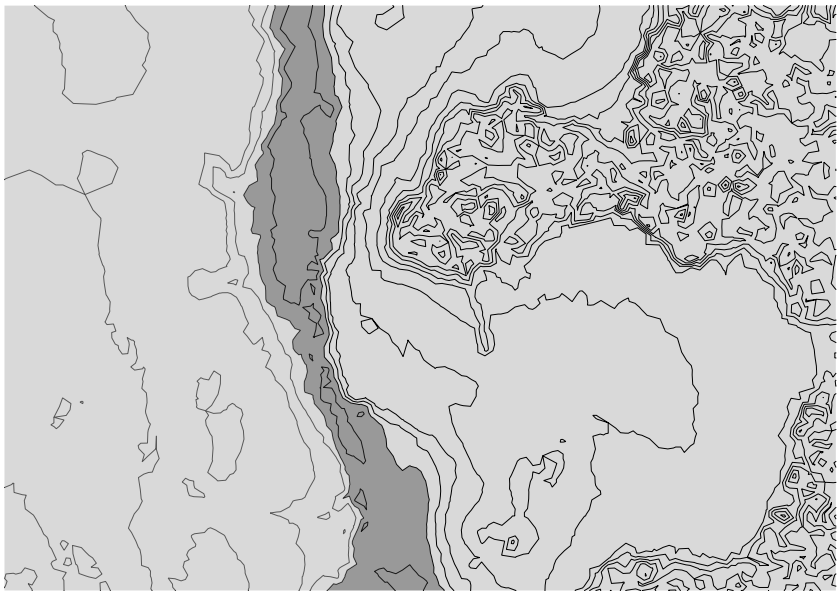


Figure 10: Features at the lowest level of details. Channel in dark grey, peaks in light grey.

5 Conclusions and perspectives

In this paper, a new method for extracting and analysing terrain features from a contour map is presented. This method classifies the features into eminences (peaks and ridges), depressions (pits and channels) and mixed features. The main contribution of this work is that the features are extracted at different levels of detail based on their spatial relationships. Other information, such as the locations of passes, can also be extracted from the feature tree. The method is based on the construction of a region graph where both open and closed contours are represented and features can be delineated by several contours.

The method is based on an iterative process which identifies features at the highest level of detail first and moves to lower levels by aggregating regions into larger features. Once the process is complete, a final stage is still required in order to remove any spurious features that may appear in the tree. Feature classification is based only on adjacency relationships from the region graph and on comparison of their heights. No geometric criteria such as the distance between contours, the area of a region or the steepness of a slope are used in the process, avoiding any kinds of error due to numerical approximation or the use of any threshold parameter.

The feature tree allows easy identification of terrain features and terrain analysis from the contour map. The number of features depends on the vertical interval between the contours. The smaller it is, the more detailed the terrain description. However, in rugged areas, characterisation of features at high levels may be more difficult if the interval is too small resulting in a large number of mixed features which are noise.

Information stored in the tree can also be applied to map generalisation. An example is presented for isobathymetric line generalisation. The interest is that some model generalisation operations can be performed automatically. This issue is quite impor-

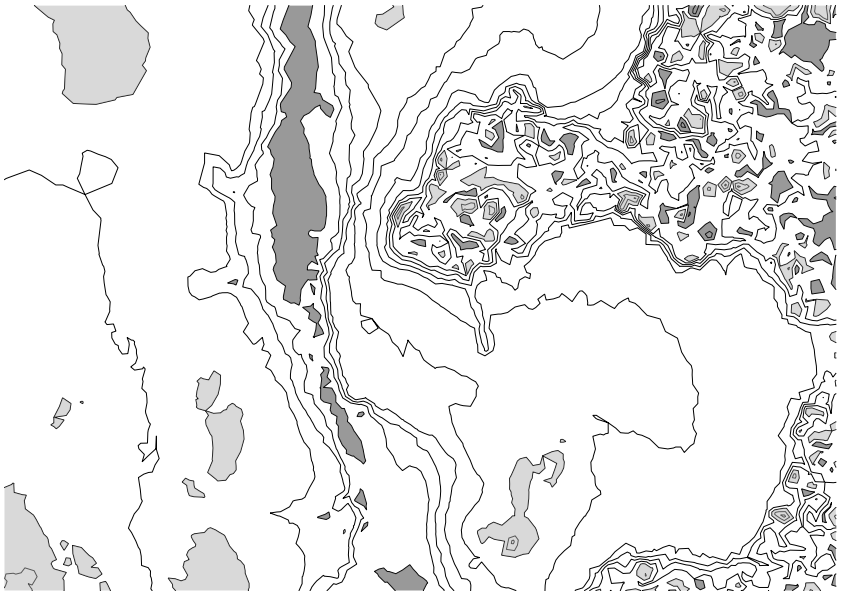


Figure 11: Features at the highest level of details. Pits in dark grey, peaks in light grey.

tant in cartography as most of the existing work on contour generalisation focuses on cartographic generalisation where only legibility and aesthetics are considered.

Other limitations are due to contour representation and the lack of information about terrain morphology between the contour lines. Point features such as saddle points and local terrain characteristics such as slopes and curvatures cannot be computed from the contour set and represented in the feature tree. Such computation requires the definition of a DEM from the contours with a sufficiently small vertical interval. Nonetheless, contours provide a structure, the feature tree, wherein features are organised according to their spatial relationships, and may be combined with information extracted from the DEM. For instance, in [4], the authors make use of contours to delineate the spatial extent of morphometric features computed on a DEM.

A first direction for future work is to achieve a more thorough feature description for chart generalisation. Feature points extracted from a DEM or spot heights from a topographic map can be considered to define geometric parameters (feature shape, slope) for this purpose. Based on the classification of underwater features provided in [12], a taxonomy of features based on their geometric, topological and contextual information may be established. Specific generalisation constraints and operators can be defined with consideration of terrain features. These sets of constraints and operators can be worked out at different levels: operators that apply either to the contours or to the features [30]. Analysis of terrain features can also be undertaken to estimate the quality of a generalisation by measuring the amount of information preserved on different maps [2].

Second, feature trees provide a qualitative description of a landscape. They can be used for different types of applications that relate to qualitative description of landforms such as valleys [25], or to positioning and wayfinding such as describing or locating a position or an itinerary from users' representations [14]. The feature tree may be used in both circumstances to translate numerical positions into qualitative descriptions or

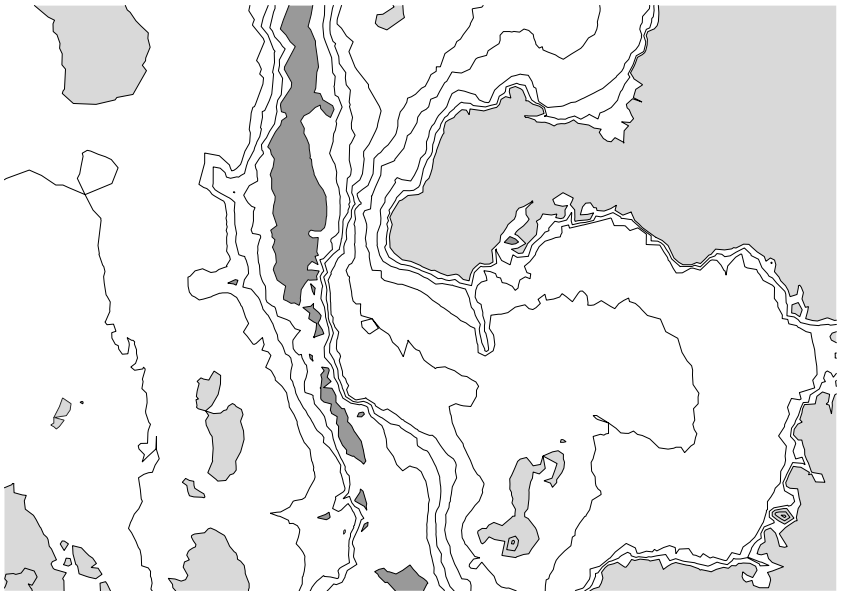


Figure 12: Reduction of map complexity. Only features from levels 1 to 5 are kept.

to locate a position from a qualitative description.

A Construction of the inter-contour region graph

This section presents the method used for constructing the inter-contour region graph from the contour graph. A contour is defined by a polygonal line and an elevation (Figure 15). Its neighbours are stored into two lists, one for each side of the contour referred to as the positive and negative sides following definitions in CGAL [6]. A region is defined by a set of contours forming its boundaries (Figure 16). The feature type indicates if the region is a feature (peak, pit, ridge, channel) or not. Region height is the difference in elevation between the highest and lowest of all contours in the region. Regions are the nodes of the region graph. Each edge of the graph is stored as a triplet formed by a contour and the two regions located on either sides of the contour (Figure 16).

Construction of the inter-contour region graph is done recursively by visiting all the contours of the graph. Each time a new contour is visited, the region formed by the contour and all its contours on the side not yet visited is created (Algorithm 6).

Algorithm 6 *Construction of the inter-contour region graph.*

```
Constructor of the region graph class
Input: one contour c of the contour graph
Begin
   createRegion(c, positive)
   createRegion(c, negative)
End
```

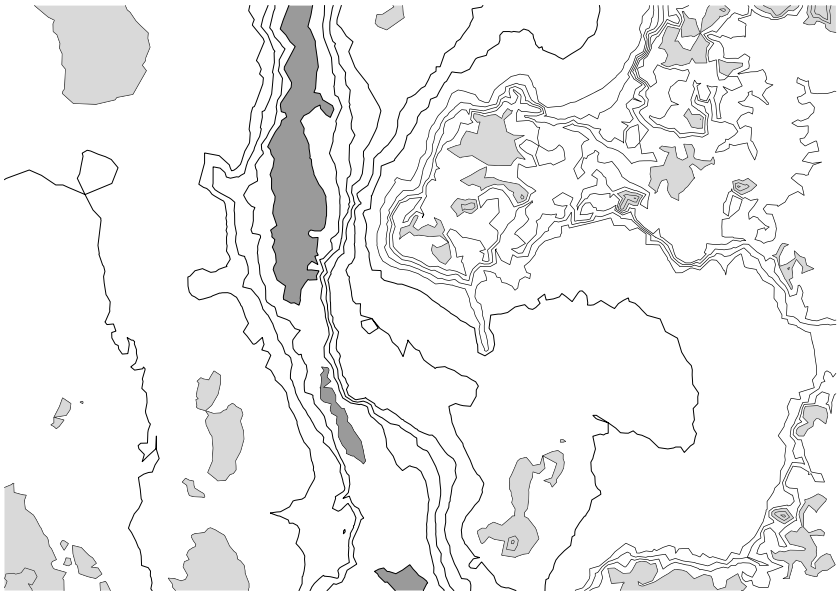


Figure 13: Feature selection for isobathymetric line generalisation.

```
Procedure createRegion(Contour, Side)
Input: one contour c and one side s (positive or negative)
Begin
   Region r = New Region(c, s)
   node.push(r)
   For each b of r.boundary
      If b ≠ c
         z = b.getSideOf(c)
         createRegion(b, −z)
      End If
   End For
End
```


Acknowledgements

The work described in this paper was supported by the Research Grants Council of Hong Kong under the General Research Fund grant PolyU 5172/08E. The author would also like to acknowledge the Hydrographic and Oceanographic Service of French Navy (SHOM) who provided the bathymetric data.

References

[1] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Surface shape understanding based on the extended reeb graphs. In S. Rana, editor, *Topological data structures for surfaces. An introduction to geographical information science*, pages 87–102. Wiley, 2004.

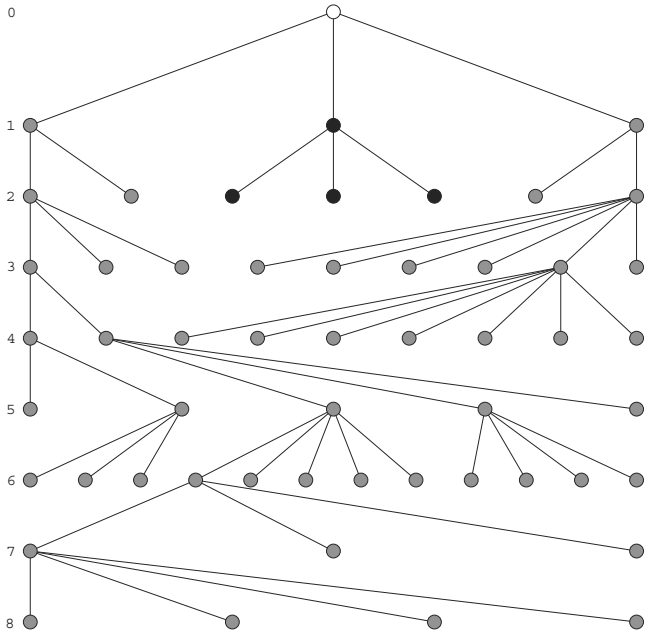


Figure 14: Feature tree after generalisation. Black nodes: depressions, Grey nodes: eminences, White nodes: mixed features.

Contour
list<Point> polyline # Number elevation # list<Contour> positiveside # list<Contour> negativeside
+ list<Contour> getAdjacentContours(Side) + Side getSideOf(Contour)

Figure 15: The Contour data structure

[2] J. T. Bjørke. Framework for entropy-based map evaluation. *Cartography and Geographic Information Systems*, 23(2):78–95, 1996.

[3] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75 – 94, 2003.

[4] O. Chaudhry and W. Mackaness. Creating mountains out of mole hills: Automatic identification of hills and ranges using morphometric analysis. *Transactions in GIS*, 12(5):567–589, 2008.

[5] J. Chen, C. Qiao, and R. Zhao. A Voronoi interior adjacency-based approach for generating a contour tree. *Computers and geosciences*, 30:355–367, 2004.

[6] Computational Geometry Algorithms Library. http://www.cgal.org.

[7] T. Cronin. Automated reasoning with contour maps. *Computers and geosciences*, 21(5):609–618, 1995.

Region
MorphometricClass featuretype # list<Contour> boundary
+ Number getRegionHeight() + Number getBaseElevation()

RegionGraph
list<Region> node # list<(Contour, Region, Region)> edge
+ createRegion(Contour, Side)

Figure 16: The Region and RegionGraph data structures

[8] E. Danovaro, L. De Floriani, L. Papaleo, and M. Vitali. A multi-resolution representation for terrain morphology. In M. Raubal et al., editors, *GIScience 2006, LNCS 4197*, pages 33–46. Springer-Verlag Berlin Heidelberg, 2006.

[9] P. Fisher, J. Wood, and T. Cheng. Where is Helvellyn? Fuzziness of multi-scale landscape morphometry. *Transaction of the Institute of British Geographers*, 29(4):106–128, 2004.

[10] T. Gökgöz. Generalization of contours using deviation angles and error bands. *The Cartographic Journal*, 42(2):145–156, 2005.

[11] E. Imhof. *Cartographic Relief Representation*. de Gruyter, Berlin, 1982.

[12] International Hydrographic Organization. *Standardization of undersea feature names*. International Hydrographic Bureau, Monaco, 4^{th} edition, 2008.

[13] I. S. Kweon and T. Kanade. Extracting topographic terrain features from elevation maps. *CVGIP: Image Understanding*, 59(2):171–182, 1994.

[14] J.-M. Le Yaouanc, É. Saux, and C. Claramunt. A semantic and language-based representation of an environmental scene. *Geoinformatica*, 14(3):333–352, 2010.

[15] Z. Li and H. Sui. An integrated technique for automated generalization of contour maps. *The Cartographic Journal*, 37(1):29–37, 2000.

[16] X. Liu and J. R. Ramirez. Automated vectorization and labeling of very large raster hypsographic map images using contour graph. *Surveying and Land Information Systems*, 57(1):5–10, 1997.

[17] W. Mackaness and M. Steven. An algorithm for localised contour removal over steep terrain. *The Cartographic Journal*, 43(2):144–156, 2006.

[18] D. Mark and G. Sinha. Ontology of landforms: Delimitation and classification of topographic eminences. In M. Raubal, H. J. Miller, A. U. Frank, and M. F. Goodchild, editors, *Proceedings of the 4^{th} International Conference on Geographic Information Science, extended abstracts, Münster, Germany*, pages 129–132, September 2006.

[19] K. Matuk, C. Gold, and Z. Li. Skeleton based contour line generalization. In A. Riedl, W. Kainz, and G. A. Elmes, editors, *Progress in Spatial Data Handling*, pages 643–658. Springer Berlin Heidelberg, 2006.

[20] S. P. Morse. A mathematical model for the analysis of contour line data. *Journal of the Association for Computing Machinery*, 15(2):205–220, April 1968.

[21] S. P. Morse. Concepts of use in contour map processing. *Commun. ACM*, 12(3):147–152, 1969.

[22] S. Rana, editor. *Topological data structures for surfaces. An introduction to geographical information science*. Wiley, 2004.

[23] J. Roubal and T. Poiker. Automated contour labelling and the contour tree. In *Proceedings of the Auto-Carto 7. Digital representations of spatial knowledge*, pages 472–481, 1985.

[24] R. K. Straumann. Experiences in developing landform ontologies. In *Proceedings of Geomorphometry 2009*, pages 17–21, 2009.

[25] R. K. Straumann and R. S. Purves. Computation and elicitation of valleyness. *Spatial cognition and computation*, 11(2):178–204, 2011.

[26] S. Takahashi. Algorithms for extracting surface topology from digital elevation models. In S. Rana, editor, *Topological data structures for surfaces. An introduction to geographical information science*, pages 31–51. Wiley, 2004.

[27] M. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. L. Schikore. Contour trees and small seed sets for isosurface transversal. In *Proceedings of the 13th ACM symposium on computational geometry*, pages 212–220, 1997.

[28] G. W. Wolf. Topographic surfaces and surface networks. In S. Rana, editor, *Topological data structures for surfaces. An introduction to geographical information science*, pages 15–29. Wiley, 2004.

[29] J. Wood. *The geomorphological characterisation of digital elevation models*. unpublished PhD thesis, University of Leicester, 1996.

[30] X. Zhang and E. Guilbert. A multi-agent system approach for feature-driven generalization of isobathymetric line. In A. Ruas, editor, *Advances in Cartography and GIScience. Selection from ICC 2011, Paris*, volume 1, pages 477–495. Springer, 2011.