INT. J. GEOGRAPHICAL INFORMATION SCIENCE, 2002
VOL. 16, NO. 8, 773–794

Taylor & Francis
Taylor & Francis Group

Research Article

Characterisation and generalisation of cartographic lines using Delaunay triangulation

P. M. VAN DER POORTEN
Department of Computer Science, Cardiff University, Newport Road, PO Box 916, Cardiff CF24 3XF, Wales, UK; e-mail: scmpv1@cardiff.ac.uk

and CHRISTOPHER B. JONES
Department of Computer Science, Cardiff University, Newport Road, PO Box 916, Cardiff CF24 3XF, Wales, UK; e-mail: c.b.jones@cs.cf.ac.uk

(*Received 10 March 2001; accepted 13 January 2002*)

Abstract. A method is presented for generalising cartographic lines using an approach based on determination of their structure. Constrained Delaunay triangulation is used to construct a skeleton of the space surrounding the lines and hence represent line features in terms of skeleton branches. Several statistical measures are used to characterise the triangulation branches. The measures enable selective generalisation of different types of line feature, leading to the possibility of user-specification of the style of generalisation. In our implementation of the approach, the triangulation is updated dynamically to allow both sides of multiple lines to be processed, while guaranteeing topological consistency between the resulting generalised lines.

1. Introduction

Map generalisation is the process of creating a legible map at a given scale from a more detailed geographical dataset. The art of map making lies in selecting both what features to include and how to represent them (Buttenfield 1985). Typically the effective representation of selected generalised features involves one or more of a variety of processes. These include reduction in detail of lines, areas and surfaces; caricature or typification of shape and pattern; amalgamation of neighbouring features; exaggeration of features that would otherwise be too small; collapse in dimensionality from areas to lines or points; and displacement to ensure adequate separation distances. Many procedures have been developed to automate these processes (Weibel and Dutton 1999) though the majority of these are concerned with generalisation of isolated lines (McMaster 1987, Thapa 1988, Visvalingam and Whyatt 1993, Wang and Muller 1988, de Berg *et al.* 1998).

Despite the attention paid to generalisation of lines, existing automated procedures have serious shortcomings. A failing common to many of the line generalisation methods so far proposed is that they treat the cartographic line as an abstract geometric entity. In so doing they do not take into account the line's geographical

International Journal of Geographical Information Science
ISSN 1365-8816 print/ISSN 1362-3087 online © 2002 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/13658810210149434

nature—the fact that it represents a feature of the physical, political, social or economic world. The line might represent a road, a coastline, or a river, and its curves and indentations may represent significant sub-features, such as a hairpin bend, a peninsula, or a delta. Many existing algorithms generally do not 'see' such sub-features, and may remove them or distort them inappropriately. Consequently, while they perform very well as point reduction techniques, they are not designed to achieve effective cartographic generalisation, particularly in regard to the retention of original shape characteristics.

Some recent work has attempted to identify sub-features, essentially by looking for so-called 'critical points' at different scales (points of maximum curvature and points of inflection) (Thapa 1988, Wang and Muller 1988, Plazanet *et al.* 1995). Using critical points is, however, a somewhat indirect means of finding sub-features. It requires additional processing to move from the points to the features, as the points detected by some critical point methods do not correspond exactly to those that would be identified as such by a human observer (Wang and Muller 1988). Furthermore the simpler of these methods can produce results which are less than ideal (Visvalingam and Herbert 1999). More refined approaches generally require some form of smoothing in order to obtain a hierarchy of features at different scales (Plazanet *et al.* 1995, Thapa 1988).

A further problem with the majority of existing methods is that they fail to respect topological relations between different lines, or even different parts of the same line. The generalisation process may create artefactual intersections between lines or parts of the same line, and this usually has to be cleared up with *ad hoc* post-processing (Muller 1990).

The procedure of de Berg *et al.* (1998) avoids crossovers within a line and between a line and neighbouring points (but not other lines). The procedure of Saalfield (1999) avoids self-intersections and crossovers with neighbouring features but at the expense of the possibility of no generalisation being performed, as it is based on 'unwinding' the Douglas and Peucker(1973) algorithm. Neither of these procedures is concerned with shape characteristics of the lines.

Visvalingam and Whyatt (1993), and Visvalingam and Herbert (1999) describe an approach, based on examining the area of triangles formed by consecutive triplets of line vertices. This appears to make good progress towards identifying features, which it tends to remove selectively, in an order determined by their area. This approach is perhaps closest in its aims to the method outlined in this paper.

This paper explores an alternative approach to the problem of identifying sub-features, expanding the ideas outlined in van der Poorten and Jones (1999) and Ai *et al.* (2000). This approach has some potential advantages over existing methods, notably its ability to eliminate sub-features on the basis of a set of shape parameters. In addition it is guaranteed to preserve topological relations between linear objects generalised as a group.

The method is based on a *dynamically updated* Delaunay triangulation, and is optimised for efficiency. This allows multiple lines to be generalised with *both sides* of the lines being considered equally, while attempting to keep processing times down to acceptable levels (currently around 10 minutes for a 30 000 points dataset, using a P3 600 Mhz). For the sample data used here the use of this more efficient approach (as compared to one involving repeated retriangulation and reprocessing of the entire map area) reduces typical processing times from hours to seconds.

An outline of the paper is as follows:

- The basic principles of the approach are first described, and a number of definitions given.
- The method of analysing the data set prior to processing is then described in detail and the various 'metrics' used to determine the style of generalisation are defined.
- A more detailed description is given of the implementation of the method, with emphasis on the complications implied by the need to optimise the algorithm.
- The benefits of the method are then described and a number of sample results are given to demonstrate this and to compare the approach with others.
- Finally some possibilities for further development of this approach are discussed.

2. A triangulation based approach

Critical point methods approach the problem of segmenting the line into distinct 'features' by examining the line itself. An alternative approach is to identify such features by examination of the space surrounding the line. The hope is that such an approach would allow sub-features to be identified in a more direct fashion than in the former method. Additionally it should be possible to calculate a variety of descriptive statistics about the sub-features so identified.

It was decided to use the method of Delaunay triangulation for the purposes of investigating the space surrounding the line. Such an approach, strictly speaking *constrained* Delaunay triangulation (CDT), has proved fruitful in exploring other aspects of cartographic generalisation. For example, the use of triangulated networks is helpful for handling the various operations (e.g. amalgamation, collapse and displacement) necessary for generalising areal objects (Jones *et al.* 1995). The benefits of triangulation derive particularly from the rich neighbourhood relationships that are encoded in the triangulation. This leads, for example, to very efficient search procedures, as well as the identification of local proximal relations that can be exploited in triangle transformations such as collapse and re-attribution.

The line or set of lines to be generalised is enclosed in a containing box and a constrained Delaunay triangulation is created for the resulting area (figure 1). The constraint is that the line segments making up the given lines and the bounding box must be retained as edges within the triangulation. The triangulation so obtained is then examined in an attempt to gain an idea of its structure.

In generating a CDT of a line, or a set of lines, we hope to identify *geometric features* of the line that may become candidates for elimination for purposes of line generalisation. A *geometric feature* is a part of the line that a human observer recognises as a distinct entity or sub-entity corresponding to a characteristic form in the real world. In practice this will be in the form of a bend, an embayment or a protuberance. Such features occur at different levels of detail, a feature may have a sub-feature. The CDT helps to find these features because in a digitised line they will be associated with sets of triangles that fill the space that the feature contains. An analogy may be made with the medial axis transformation or skeleton which has a history of assisting in identifying features associated with curves (Lee 1982, Ferley *et al.* 1997, Gold 2000). In a CDT, sequences of neighbouring triangles form paths that *approximate* the location of the skeleton. The sum of these paths constitutes a hierarchy of branches and sub-branches that we regard as features of the line.

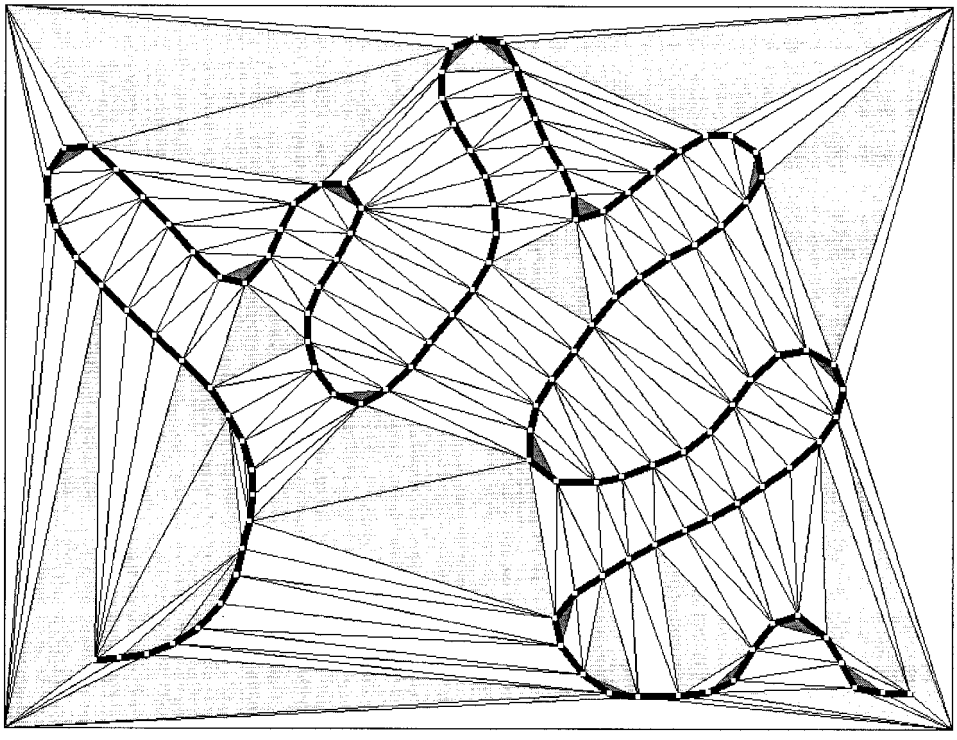


Figure 1. A sample line and its corresponding constrained Delaunay triangulation. Leaf triangles are shaded dark grey, trunk triangles are white and branching triangles are light grey. (Note: artificial data, regularity of point spacing is a result of the way data were generated).

Having identified features in terms of sets of triangles, we can calculate metrics that may be used to recognise particular types of feature, and hence make decisions on the selective elimination of features. An important characteristic of features composed of sets of triangles is that, provided the vertices of the respective triangles constitute a continuous sequence of points belonging to a single line, their elimination is guaranteed to avoid topological inconsistencies. This is because by definition a triangulation of all the geometry in an area cannot contain any other geometry and hence its controlled collapse cannot result in overlap of other features.

2.1. *Triangulation components*

We now define the components of a CDT that lead to the identification of a hierarchy of branches.

Edges of the triangulation are described as *real* if they belong to an original line, and therefore constrain the triangulation, *external* if they belong to the bounding box, and otherwise as *virtual*.

Two triangles that share a common edge are described as *internal neighbours* if the edge is virtual and *external neighbours* if the common edge is real. Triangles that are internal neighbours are also described as being *connected*.

A triangle with two real edges is a *leaf* triangle. A triangle with one real edge is a *trunk* triangle and a triangle with no real edges is a *branching* triangle. As we will see below, branching triangles are further subdivided into *internal*, *root* and *external*.

Figure 1 illustrates the primary categorisation of triangles into three types according to their number of real edges.

A *branch* in the CDT of an open line is a contiguous set of connected triangles that is bounded by a sequence of real edges belonging to the line, and by a single virtual edge, referred to as the base edge of the branch (see figure 2). Figure 2(*b*) shows a complex branch with sub-branches.

The sequence of real edges is defined to be the *feature* of the line that the branch represents, ideally coinciding with the geometric feature defined above. The two vertices of the base edge are the first and last vertices of the feature.

A branch may be composed recursively of sub-branches, corresponding to sub-features within the feature represented by the parent branch. Triangles composing sub-branches are subsets of the triangles composing the parent branch.

In order to determine the branching structure associated with a CDT we make a distinction between different types of branching triangle. Sub-branches of an entire branch stem from internal branching triangles, while the entire branch stems from a branching triangle referred to as the root triangle. There is a third type of branching triangle referred to as an external branching triangle. We distinguish between these types of branching triangles on the basis of 'pathset attributes' of their component edges, as explained below. It should also be remarked however that all the vertices of an internal branching triangle belong to the parent branch, while a root triangle will have two vertices on the branch of which it is the root and a third belonging either to the bounding box or to another line. Those of an external branching triangle will belong to three different lines (or two lines plus the bounding box).

A *path* is an ordered sequence of connected triangles. Paths cannot backtrack on themselves, but they may form a loop. The virtual edges of each triangle are categorised with one of three pathset attributes, according to the paths that cross them relative to that triangle. For a given triangle, a virtual edge has a *leafward* pathset attribute if all paths across that edge from the triangle will lead inevitably to a leaf triangle. An edge has a *rootward* pathset attribute if traversal of the edge can lead to a root. It is always the case that an edge designated as leafward by one triangle will receive a rootward attribute from the neighbouring triangle that shares

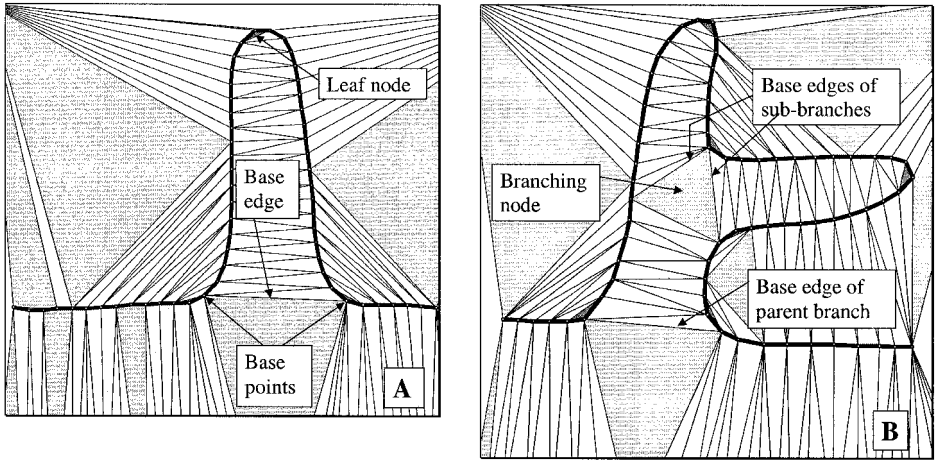


Figure 2. A feature, a branch and its base edge.

the edge. An edge is given a *neutral* pathset attribute if its traversal can lead to a loop, i.e. it is possible to follow a path that enters the current triangle via one of its other virtual edges. The neutral pathset attribute is attached irrespective of whether or not the edge could also lead to a root.

Ai *et al.* (2000) extends van der Poorten and Jones (1999) with the addition of a fourth triangle type, but here we retain the three basic types listed above, with the addition of a further sub classification depending on the pathset attributes of the given triangles. Hence branching triangles may be of either *internal* or *external* type.

An additional distinction is possible, between trunk triangles whose vertices all lie on a single line (or, equivalently, which has a leafward and a rootward exit) and those who's vertices are shared between two lines or a line and the bounding box (or has two neutral exits). Connected sets of the latter triangle type constitute channels between lines. This allows trunk triangles to be further subcatagorised as either *internal trunk* (the former case) or *external trunk* (the latter).

The above considerations give rise to a total of six types of triangle (leaf, internal trunk, external trunk, internal branching, root, and external branching). These definitions are easier to understand with reference to figure 3. Here pathset attributes are shown as lines from the centre of the triangle to the relevant edge. A thick solid line indicates a neutral pathset attribute, a thin black one a leafward pathset attribute and a thin grey one a rootward pathset attribute (the thickest black line is the

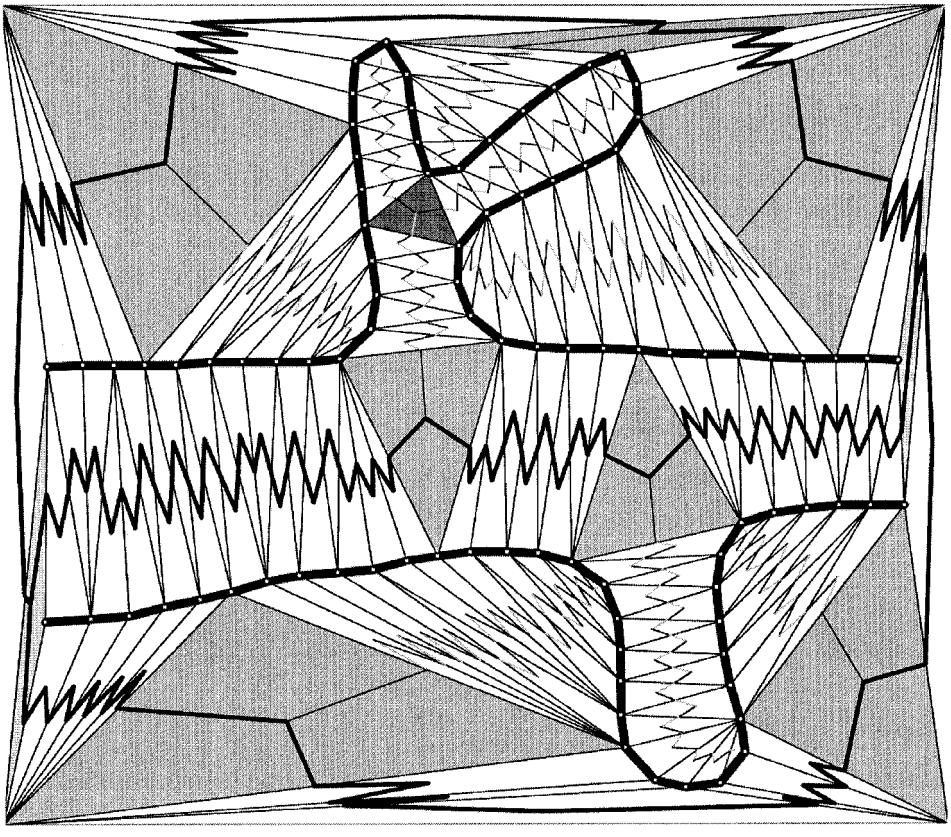


Figure 3. Triangulation with Pathset attributes.

original line). Triangles are shaded here not purely according to the number of internal edges, but also according to the pathset attributes of those edges. This is significant in identifying 'root' triangles.

Triangles with one or more real edges (i.e. leaf and trunk triangles of either type) are white. Of the remaining (branching) triangles those with exactly one leafward edge are the root triangles of branches. These are medium grey in figure 3. Branching triangles with two leafward edges are *internal* branching triangles, and are shaded dark grey. Those with three neutral edges (and hence zero leafward edges), are *external* branching triangles and occur only when more than one line is present. These are light grey in the figure.

2.2. *Calculation of pathset attributes*

Calculation of pathset attributes starts by locating all leaf triangles. We work rootward from these, setting pathset attributes as we go (rootward in the direction we are travelling, leafward in the reverse) until a branching triangle is reached. Every such triangle has a count of the number of leafward exits it has, and when such a triangle is reached while travelling rootward from a leaf this count is incremented.

Once all leaf triangles have been processed in this way, we examine all branching triangles with two leafward exits. We then follow the remaining path from these triangles rootwards, until reaching another branching triangle, where we increment its leafward exit count, just as we did in the previous step. This second step is performed repeatedly, each time starting at all the branching triangles that were found to have two internal paths in the previous iteration. The process ends when we find no more internal branching triangles. Any remaining paths are neutral ones.

2.3. *Branch statistics*

Once the triangles and pathset attributes have been so categorised (and marked) we have an implicit hierarchy of features. Features stem from the leafward edges of the identified root nodes, continuing in a leafward direction from triangle to triangle, while sub-features spawn from the leafward edges of branching triangles. With this information we can calculate a number of statistical properties relating to each branch and sub-branch. For each branch (or sub-branch) these values are stored as part of a record associated with the edge of the root (or branching triangle) which forms the base edge of that branch (or sub-branch).

The statistics (or 'metrics') include:

- The area of the branch
- The length of the boundary of the branch
- The length of the branch
- The height of the branch
- The average width of the branch
- The standard deviation of the branch width
- The area of the convex hull of the branch
- The true error of the branch
- The boundary difference of the path
- The aspect ratio of the path
- The 'base angle' of the branch

These metrics are generally applicable to all branches and sub-branches, including those with sub-branches, (though in the latter case an expanded definition is required for some metrics).

For most of these metrics their value for a particular branch or sub-branch is partly determined by summing the respective values of any constituent sub-branches it may have (plus the contribution from the remainder of the branch). In the case of simple branches (or sub-branches) with no sub-branches, evaluation of the metrics usually starts by calculating the contribution of each triangle in the triangulation, before summing the contribution of the triangles in the branch. This is done for reasons of efficiency, so that changes in the triangulation mesh require the minimum amount of recalculation. When a region of the triangulation is retriangulated only the new or changed triangles have to have their statistics (area, contribution to boundary length, etc) recalculated.

The first two statistics are self-explanatory. The *area* of the branch is obtained by summing the area of its component triangles. The *boundary length* is found by summing the length of the real edges of its triangles. Both these definitions apply unproblematically to complex branches.

To define the *length* of a branch it is helpful to define a 'node length' for each triangle in the triangulation. The node length of a trunk triangle is the distance between the midpoints of its two internal edges. For the leaf triangle the relevant distance is that from the midpoint of its (single) internal edge to its opposing vertex. For a branching triangle there are two possible node lengths, depending on which sub-branch is being measured—the distance from the centre of the rootward edge to the centroid of the triangle plus the distance from the centroid to either of its leafward edges. The 'centroid' is defined as the point whose co-ordinates are the averages of all the corner co-ordinates.

The length of the branch is calculated by summing the node lengths of all the triangles that make up the branch. This is almost equivalent to measuring the length of the skeleton of the branch, an approximate form of which can be derived by connecting the midpoints of the edges of the triangles in this fashion. A pseudo-skeleton (hereafter referred to as the 'skeleton'), so derived, is shown in figure 4. The start and end points of a selected branch of the skeleton are shown, the branch length being the length of the line between them. However when measuring the length of a branch, the starting point is taken as the centre of the base edge of the branch, not the point at which the skeleton branch connects with the parent skeleton branch (figure 5). This is advantageous as it gives a far more accurate measure of the real size of the feature. One of the problems with using skeletons in shape simplification is that a small feature on the line can give rise to a large branch of the skeleton (figure 5 again), i.e. the skeleton is very sensitive to 'noise' (Ferley *et al.* 1997).

The length of a complex branch is considered to be the length of its longest path. That is, we follow the branch from its baseline, taking the longest branch at each junction. This also determines which of the two 'node lengths' of the branching triangle itself is used (we use the one associated with the longest total path). Of course, to determine which is the longest branch at each junction we have to measure each branch, and they might themselves be complex (figure 6), requiring use of this definition, but eventually we will reach simple branches and can start working back up again. Essentially the process is recursive.

The *height* of a branch is really an alternative definition of its length. In this case, instead of summing the distance between the centre points of the virtual edges of each triangle in the branch we sum half the height of each triangle in the branch (where the height is measured taking the rootward edge as the base—that is, the

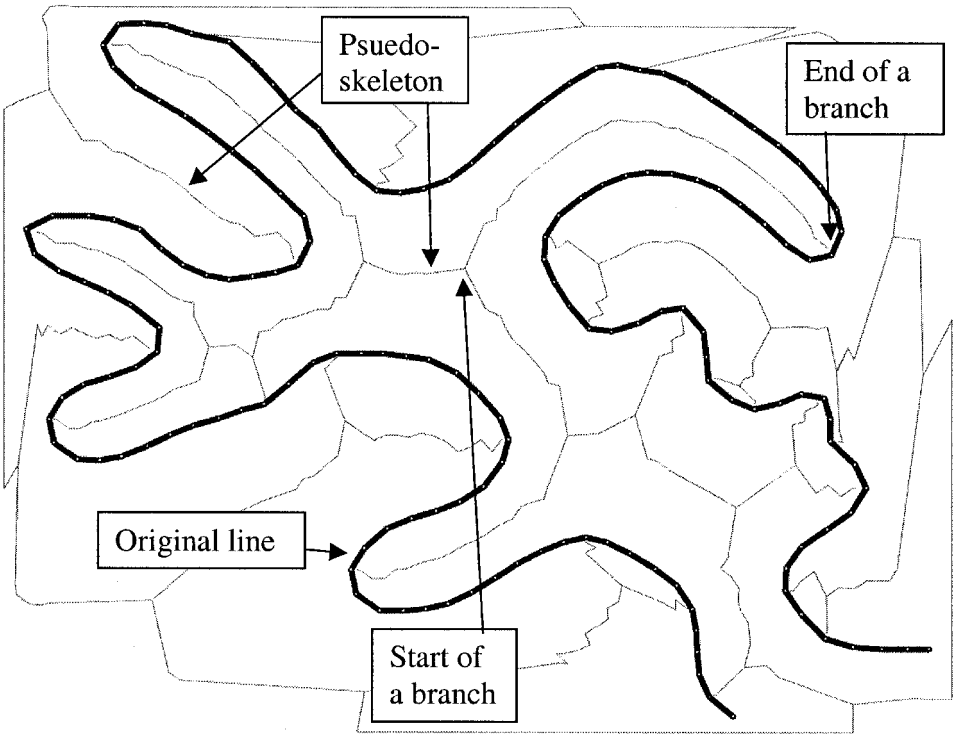


Figure 4. Pseudo skeleton.

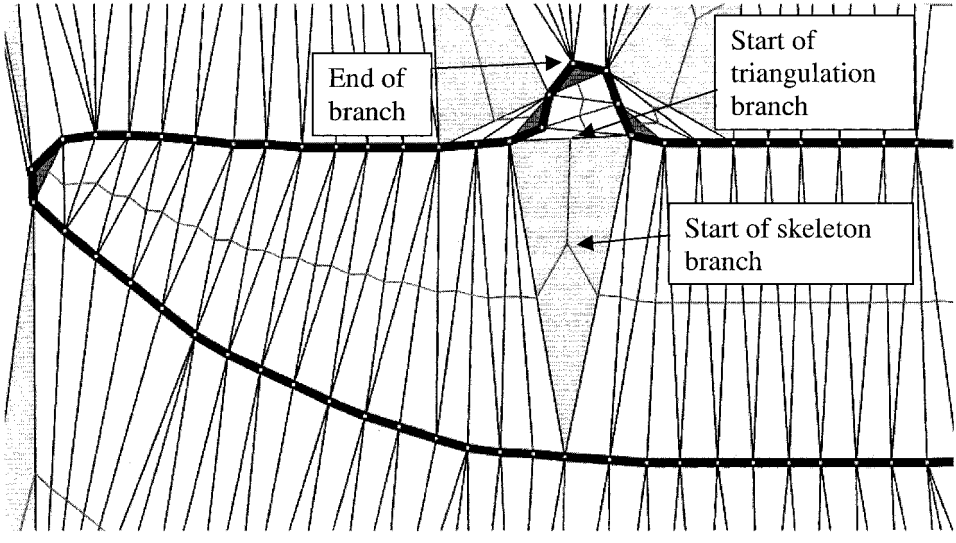


Figure 5. Branch length *versus* skeleton branch length.

edge on which the path from the root enters the triangle). An exception is made for leaf triangles where the whole height is used. As with *length*, the height of a complex branch is considered to be the height of its longest path.

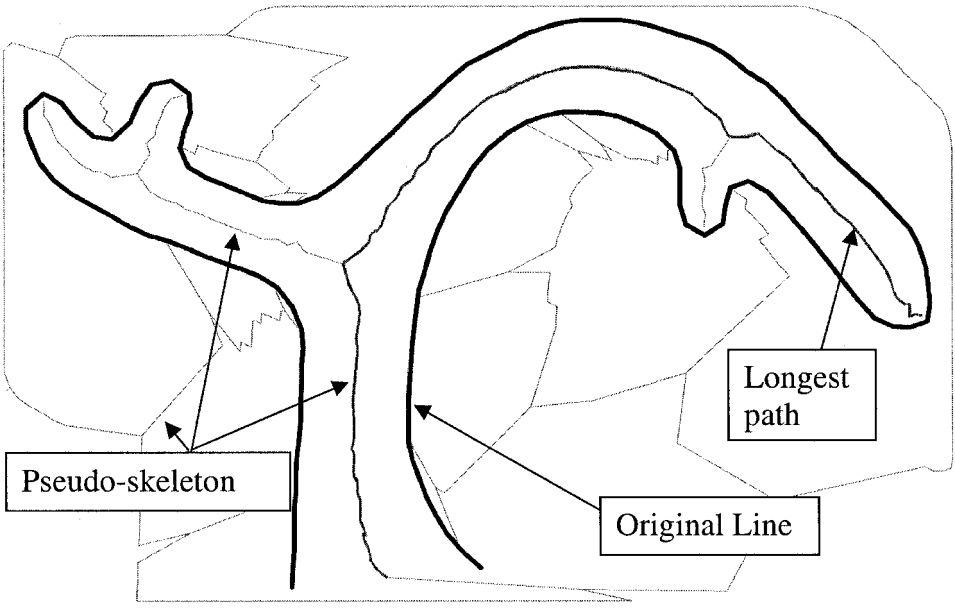


Figure 6. The longest path is used to calculate the length of a complex branch.

The *average width* of a branch is defined to be the total area divided by the branch height. For simple branches it is a weighted sum of the average widths of all triangles in the branch. For complex branches the use of this formula has the effect of treating the longest path as the main path of the branch and treating other sub-branches as if they were simply variations in the width of the branch.

The *area of the convex hull* of the branch is an alternative measure of the area of a branch, particularly relevant when considering complex branches (branches with sub branches). This was suggested by Ai *et al.* (2000) and is added here as another useful metric.

The *true error* metric is a measure of the displacement error that would be introduced into the generalisation if the relevant branch were to be deleted—that is, if it were to be replaced by a straight line segment between its end points. Unlike the other metrics this is not obtained from the triangulation itself, but from comparing the base edge of the branch to the original line between those points and calculating the Hausdorff distance.

The *boundary difference* is the difference between the boundary length of the branch and the length of its base line.

The *aspect ratio* is simply the ratio of the path length to its average width, giving a dimensionless measure of the shape of the path. Given the definition of average width used, this metric is equivalent to the ratio of path length squared to area.

The *base angle* measures, for a given branch, the increase in angularity of the line that would result from amputating the branch at its baseline. By only removing branches in which this number is negative or zero (i.e. for which pruning would not introduce a sharp angle in the line where none previously existed) one can, usually, avoid chopping the corners off features, while still pruning identically sized genuine features. This was the approach used in van der Poorten and Jones (1999) and a similar method is used by Ai *et al.* (2000).

More precisely, for each branch four angles are measured, two for each side of the base. The first of these is the angle between line segments A and B, where these are defined as follows. Line segment A is the segment between the base vertex and the vertex adjacent to the base vertex outside the branch. Line segment B is the segment between the base vertex and the vertex adjacent to the base vertex within the branch. This is shown in figure 7. Note that 'angle between' in this case is intended to mean the angle through which the line 'swings' at the common vertex of the two edges. The largest of the absolute values of two of these angles (one for each side of the branch) is then recorded (α, in figure 7).

The second angle is that between line segment A and the base line of the branch. Again there are two of these, one for each side, and again we take the maximum of the absolute values (γ). The difference between the value of these angles is then obtained as ($\gamma - \alpha$). This difference is a measure of how much larger the largest angle associated with the base of this branch will become if a cut is made at this point. The higher the value the less desirable the cut.

This approach is less than satisfactory because it depends entirely on the highly local properties of the line at the base of the relevant segment. The angle concerned is measured relative solely to the vertices directly adjacent to those determined to represent the base of the feature, and this small segment of line is not necessarily typical of the way the branch relates to the line when seen as a whole.

The use of these different metrics allows decisions to be made about processing the line in order to achieve different styles of generalisation. At present this simply provides a choice of different criteria for 'branch pruning', the selective removal of sections of the line. Each statistic allows a different style of pruning. For example, the use of the boundary length is equivalent to using the area multiplied by a 'compactness factor'—meaning that circularly shaped features will be more likely to be removed than less compact ones of the same area. The metrics that appear to be of most obvious interest are those of average width, branch length, and true error. The first two produce clearly contrasting styles of generalisation and the last is often required as a fundamental constraint on a generalisation.

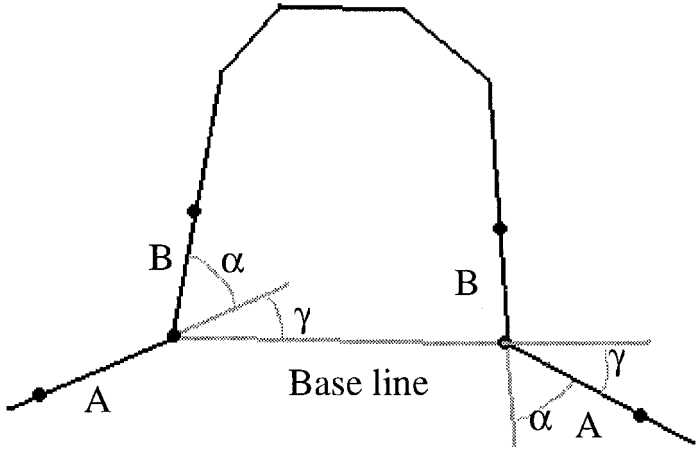


Figure 7. Angles α and γ.

3. The Generalisation procedure

Here we describe the procedure for generalisation processing by selective pruning of branches. Initially, all the branching triangles are checked and the smallest branch, according to a selected metric, is found. The segment of the line that defines this branch is then deleted and replaced by its baseline, the affected area is retriangulated and the branch sizes are recalculated. This process is repeated until the smallest remaining branch (measured by the chosen metric) is above the given threshold value.

3.1. *Combining metrics*

In general one can specify thresholds for each possible metric and remove all branches that fall below *all* of the relevant thresholds. However, it is still necessary to specify one metric as the primary one. This is due to the two-sided nature of the line. When a branch is removed, the branches adjoining it (on the other side of the line that constitutes its boundary) are affected and will almost certainly change size. These affected branches may consequently become newly eligible for pruning or cease to be so eligible, requiring metrics to be recalculated before pruning continues. An apparently simpler strategy would be to remove all the branches with metrics below the chosen thresholds in one pass. However if this strategy were employed it is quite possible that some of the new branches thus created would have smaller measurements than those deleted. Thus pruning must be done *sequentially*. This requires one particular metric to be chosen to determine the *order* of pruning, though it must be emphasised that *all* the chosen metric threshold values are used to determine *which* features are pruned.

In practice which metric is chosen to be *primary* does not greatly affect the outcome. It can have a small effect where there are neighbouring features that are both eligible for removal. In this case the prior removal of either feature may sometimes cause the remaining one to increase in size slightly and so escape pruning. Thus choice of a given metric as primary will cause the process to give marginally greater priority to removing features that fall below the specified threshold for that metric. For example making length primary when pruning by both length and width will lead to the program producing a generalisation which minimises the number of short branches remaining, while still satisfying the criteria that all surviving branches must be either longer than the length threshold *or* wider than the width threshold. For the most part however the end result is dependent only on the total set of metrics that are applied.

It should be noted that with the current method of applying multiple metrics (specifically that removable branches must be above *all* thresholds rather than *any* threshold) for a given set of metrics the line will tend to be less simplified than it would be using a subset of the given set, assuming the same threshold values.

3.2. *Two-sided dynamic triangulation*

Note that because this procedure uses dynamic retriangulation, only updating the triangulation mesh in the area affected by the removal of a branch, it can deal with multiple lines in a genuinely two-sided fashion. Ai *et al.* (2000) perform two single sided generalisations sequentially, one from each side (a method only applicable to a single line), while van der Poorten and Jones (1999) used a two sided approach but at the expense of having to retriangulate the entire region at each step. The procedure used is optimised for efficiency, and we now provide a more detailed description of how it works.

3.3. *Updating branch statistics and pathset attributes*

When a branch is deleted, it is relatively straightforward to work out which triangles in the triangulation are affected. These triangles are removed from the triangulation and the affected area is retriangulated. It should be noted that depending on the arrangement of lines in the dataset this region may well include both 'holes', i.e. triangles unaffected by the point deletions, and sections of undeleted line segments that impose constraints on the retriangulation.

In order to avoid doing unnecessary work, only those branches that have been affected should have their statistics recalculated. Finding these branches requires updated information about the paths. This is obtained by working out the pathset attributes of the new triangles deductively, making use of certain rules about what combinations of pathset attributes are allowed, together with the fact that the triangles not affected by the branch removal retain their existing pathset attributes.

Essentially one is presented with a region of the triangulation where the pathset attributes are missing. The neighbouring triangles, however, do have known pathset attributes associated with the edges that they share with the outer triangles of the unknown region. This information allows us to start filling in the missing pathset attributes, working from the outer edge leafwards using certain rules about what combinations of pathset attributes are allowed within a triangle and what pathset attribute pairings are legal for neighbouring triangles. We alternate a 'reciprocation' step with a 'deduction' step. In the 'reciprocation' step we simply look at the known pathset attributes on one side of an edge and fill in the complementary type on the other side. In the 'deduction' step we look at the known pathset attributes of each triangle and see if it is possible to deduce the remaining unknown ones. The first step spreads information from one triangle to the next, the second fills in the information within each triangle.

The rules used are as follows (***R*** denotes a rootward exit, ***L*** a leafward one, ***N*** a neutral):

- An ***R*** is always matched by an ***L*** for the shared edge of the neighbouring triangle.
- Similarly, an ***N*** is paired with another ***N*** (because if you can form a loop in one direction you must be able to reverse it and form a loop going in the other direction).
- If a triangle has only one virtual edge it must be an ***R*** type.
- If a triangle has only two virtual edges they must be {***LR***} or {***NN***}.
- If a triangle has three such edges they can only be one of the following combinations: {***NNN***}, {***NNL***}, {***LLR***}.

There is one awkward case where the process may become 'stuck' (following the above rules results in no further pathset attributes being filled in), but in this case the very fact of becoming stuck tells us what the situation is—two 'external branching triangles' (that is, {***NNN***} type) connected to each other. Hence the program can retrieve the situation once it has detected that no progress is being made.

Once the path types have been assigned, we can quickly trace back from the affected triangles to the roots of the branches they lie on. We then know which branches need to have their metrics recalculated.

4. Benefits of a triangulation based approach

The primary benefit of this approach is that it allows a significant degree of control over the *style* of generalisation produced. While many existing algorithms

allow the specification of parameters, generally these parameters merely control the degree of point reduction obtained. In fact, for the existing point reduction methods, having multiple parameters to tune is often considered a drawback, as it is not clear what each parameter in fact *means*. Instead the user is confronted with multiple means of achieving the same end, a reduction in the number of points used in the line, with no clear indication of what the difference is between tweaking parameter *a* or parameter *b* in terms of the type of generalisation obtained. In the method described here, it is possible for the user to have more control over the style of generalisation, choosing for example the widths and lengths of features that are to be retained on the generalised map.

A further important benefit is the fact that topological consistency is maintained implicitly. A major drawback of most existing methods, simple point filtering algorithms in particular, is their tendency to create bogus intersections between lines, or even within the same line. These problems often have to be cleared up with post-generalisation processing (Muller 1990). With a triangulation-based method such problems generally do not arise, provided all the linear features on a particular map are generalised together, producing a single triangulation. This method preserves topological consistency as is illustrated in figure 8 (A to F showing progressively larger degrees of generalisation, corresponding to increasing values of the pruning threshold, in this case area).

The method is also suitable for a pre-processing approach, that is, a complete generalisation can be performed once, and all vertices of the dataset labelled with the threshold value at which they are to be deleted. The data may then be displayed at any level of pruning requested more-or-less instantaneously. However a caveat must be added that the level of pruning should be described by a single metric. Use of multiple simultaneous metrics is still possible but only if all but one metric is dependent in some fashion on the remaining one (for example, being a simple multiple of it). Note that any number of *single* metrics could be used, by pre-processing the data with each metric separately and labelling the vertices with the respective values for each.

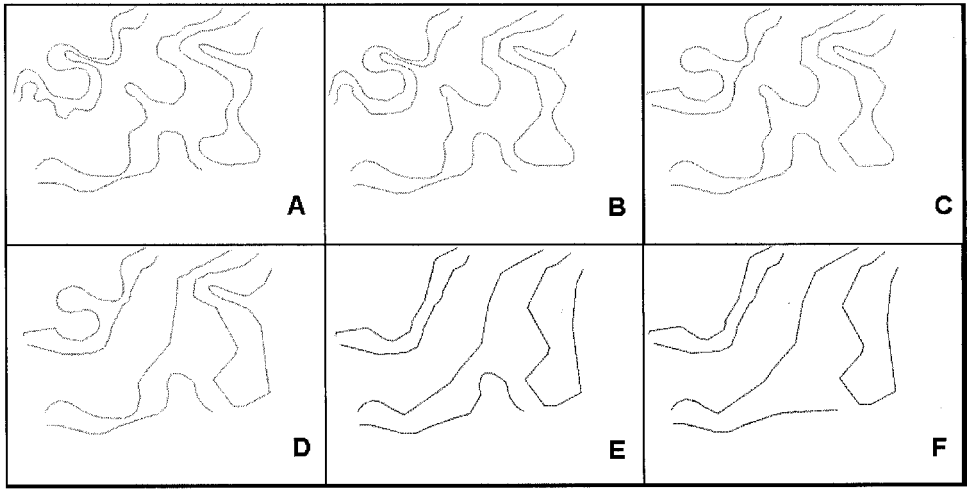


Figure 8. Preservation of topology.

5. Sample results

Some example results (using synthetic data) are shown in figures 9(*a–e*). Figure 9(*a*) shows the original line. Figure 9 (*b*) shows the same line subjected to 'pruning' where the constraining metric is the branch area. Figure 9(*c*) shows the same line pruned using the same metric with a higher threshold value. In figure 9(*d*) the branch width is used, while in figure 9(*e*) the pruning is performed by branch length, and in figure 9(*f*) pruning is by branch length with a larger value. In all cases 'base angle' is used as a secondary metric.

As one would expect, when pruning is performed on the basis of branch width, the narrow features are removed (regardless of length) while the wide features are retained. With a length metric the shorter features are pruned, whether wide or narrow. It is of course possible to specify a combination of metrics.

Figure 10 shows some real data (county boundaries, including a section of coastline) generalised using different metrics. W1, W2, W3, W4 and W5 are generalisations using progressively greater width thresholds only. L1, L2, and L3 use only a length metric, with progressively greater values. W1 + L2 uses both width and length metrics. It uses the same width threshold as W1 and length as L2.

Note that when more than one threshold is used the generalisation is more conservative, as for a branch to be pruned the relevant metric must fall below the specified value for *both* thresholds. Hence W1 + L2 retains a set of features which combine those from both W1 and L2.

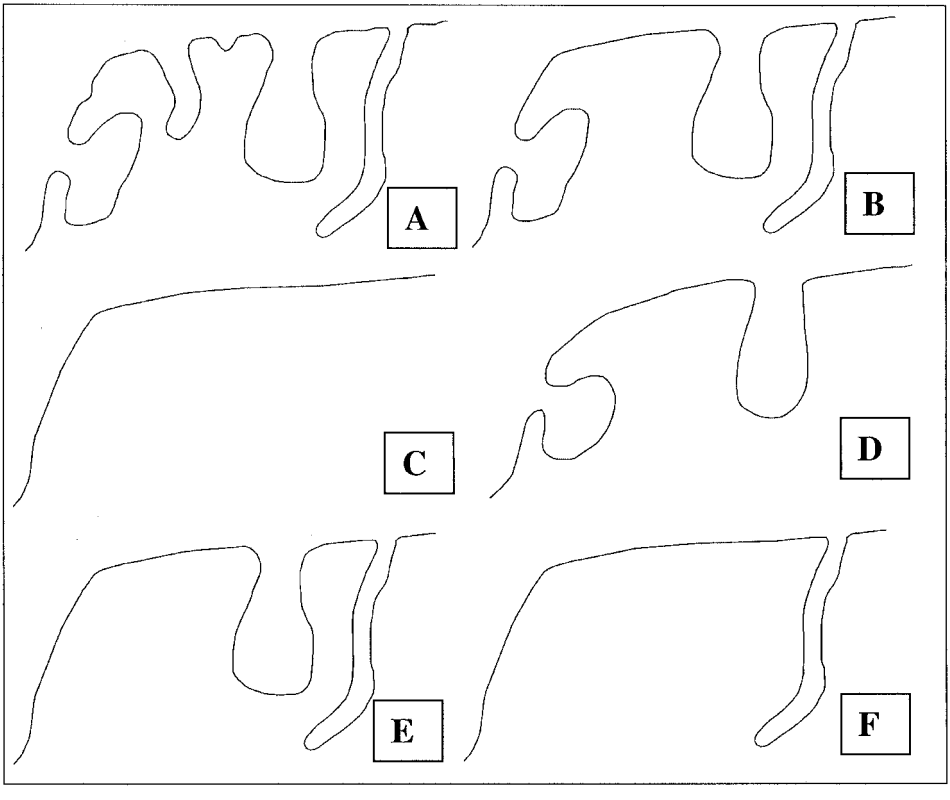


Figure 9. Alternative generalisations (see text for explanation).

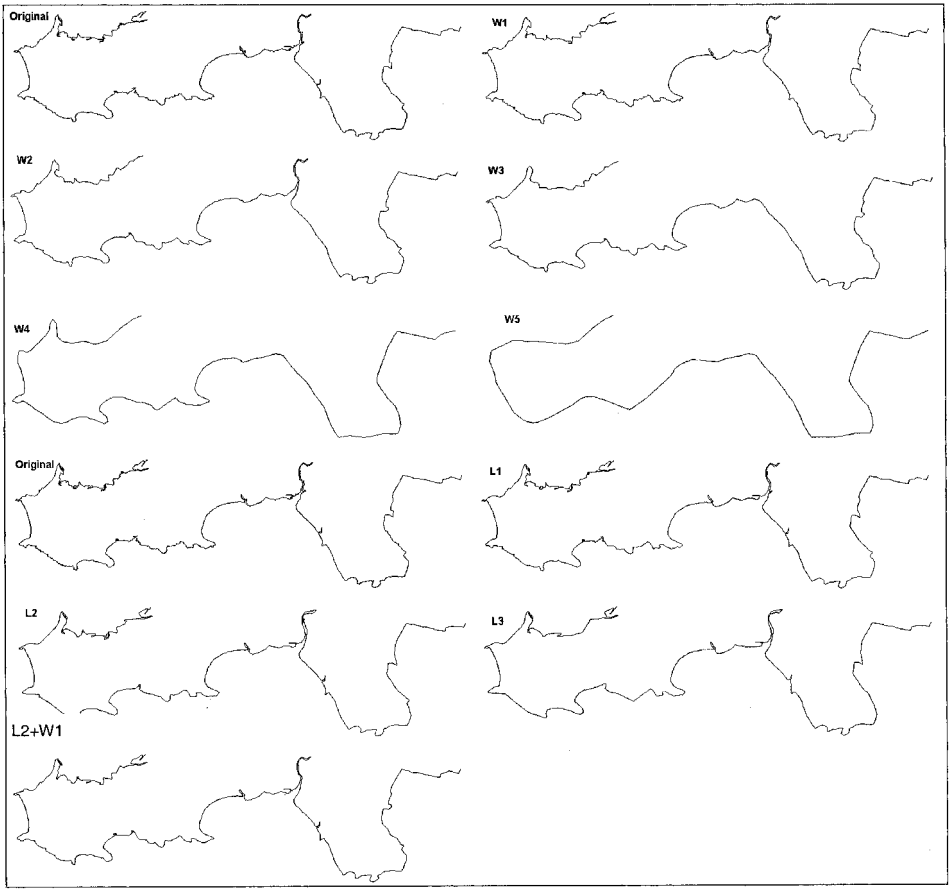


Figure 10. Real data, alternative generalisations. Ordnance Survey data. © Crown copyright, Ordnance Survey, all rights reserved, nc/01/586.

Figure 11 shows a comparison of data generalised with the Visvalingam and Whyatt (1993) algorithm and the same data generalised using branch pruning combined with a post-generalising point-filtering step using the Douglas and Peucker (1973) algorithm. This is necessary as branch-pruning is *not* intended as a point reduction algorithm (it will not remove completely co-linear points, for example). The generalisations are performed at four different levels, (the numbers in the captions refer to the number of points retained). The actual values of the thresholds used for each stage of each case are shown in table 1. These values were chosen arbitrarily as it is not easy to predict how many points will be retained for given combinations of threshold values—these values were the ones that happened to give similar numbers of points retained for each method.

The branch pruning is done by average width only and also by a combination of average width and path length metrics. Note that once the major estuary is removed the length metric has no effect and the combined metric pruning gives identical results to those using width only. This is because the estuary is the longest feature present and once the length metric exceeds this value in the combined metrics

method no features are retained specifically due to their lengths, and in all cases width is then the limiting factor.

For the combined metrics the length and width metrics were chose so as to keep a constant ratio between them. The post-processing with the Douglas-Peucker algorithm was also performed using a threshold that was a (small) multiple of the width and length thresholds.

Note that Visvalingam and Whyatt's method (1993) *does* selectively prune 'features'. It finds 'features' roughly equivalent to those detected by branch pruning, though it also tends to partition features, distinguishing where a 'feature' (as detected by the branch pruning algorithm) changes width significantly. In this case it partitions what branch pruning considers a single feature into several smaller features, e.g. the estuary in this figure. As with branch pruning (and Douglas-Peucker) it allows a pre-processing approach, enabling points to be labelled for selection at display time. The method also effectively includes its own point-filtering, so it doesn't require a separate point filtering step. The disadvantages of Visvalingam and Whyatt's method compared to that described in this paper are that it is (roughly) equivalent to pruning by area only with no other options and so lacks customisability, and that it is not guaranteed to preserve topological relations.

We chose to use Visvalingam and Whyatt's method for comparison because it seemed to be one of the few existing methods which attempted to identify features in a manner similar to branch pruning (as well as being relatively straightforward to implement).

6. Remaining issues

In this section we identify some limitations of the methods described and suggest directions for future work.

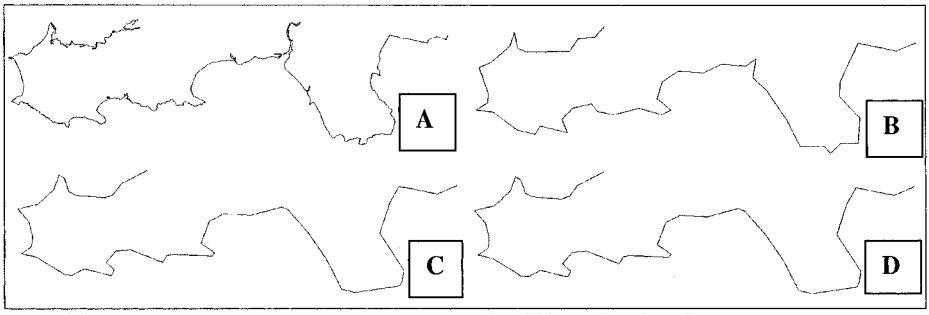
6.1. *'Corners'*

One *potential* source of problems is that the method for identifying branches cannot distinguish between psuedo-features, referred to here as 'corners', and branches that correspond to obvious perceived features (figure 12). A 'corner' is something that is detected as a feature, whose base vertices do not appear to be associated with any corresponding bends in the original line. Usually it forms part of a genuine feature, and pruning it amounts to shaving off the corners of that feature, whereas it would be preferable to leave the parent feature untouched until we wish to remove it in its entirety.

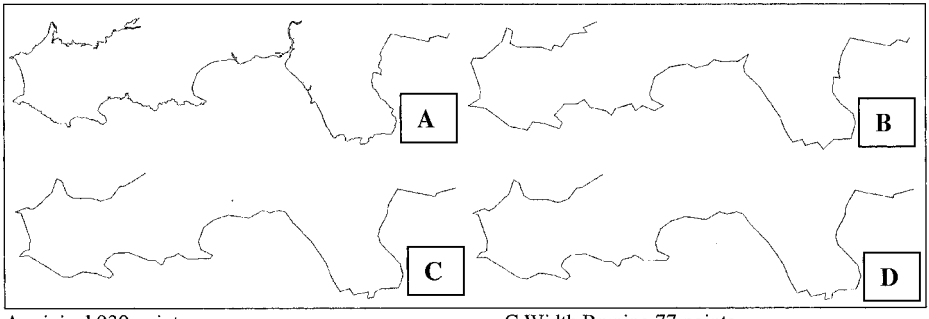
One solution to this is to examine the angle the line makes at the base of the feature (using the 'base angle' metric discussed above), and only perform pruning if the angle exceeds some threshold. Although this does sometimes work, it is sensitive to local changes in line direction that may not reflect the overall situation. It is apparent that better methods could be found. However it should be noted that the problem does not appear to arise with the real test data used in our experiments.

6.2. *'Stumps'*

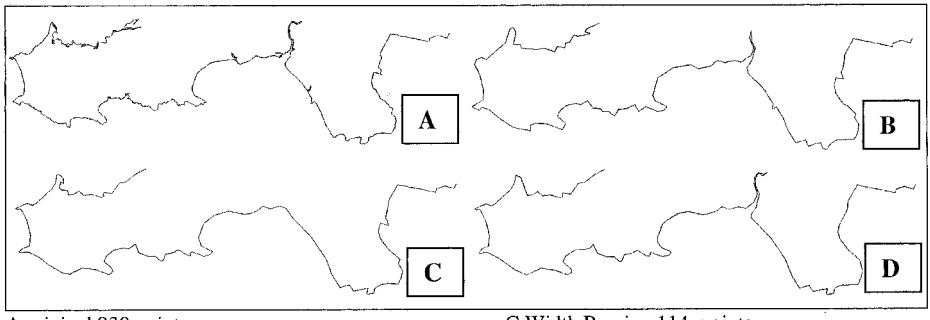
Although the triangulation and branch detection method in general performs well at identifying visual 'features' of the line, it does not always detect the base of such a feature perfectly, particularly when there is another linear feature close by. This is illustrated in figure 13. Here a cut along the base line would leave an ugly stump as a residue. This problem is not noticeable in the real data used so far. It is



A original 930 points
B Visvalingam 47 points
C Width Pruning 47 points
D Width and Length Pruning 47 points



A original 930 points
B Visvalingam 75 points
C Width Pruning 77 points
D Width and Length Pruning 77 points



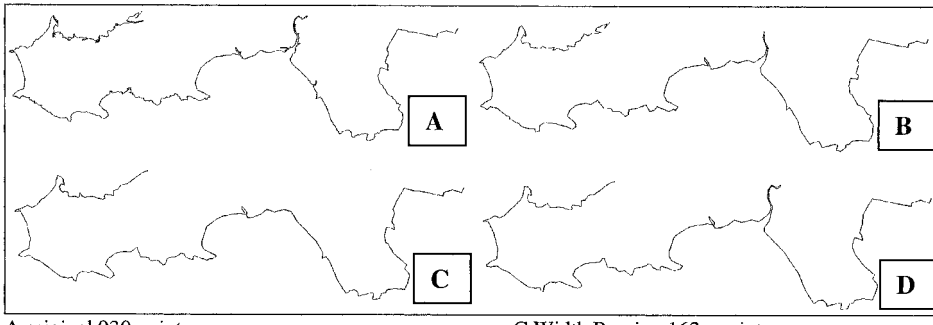
A original 930 points
B Visvalingam 112 points
C Width Pruning 114 points
D Width and Length Pruning 113 points

Figure 11. Comparison of branch pruning and Visvalingam and Whyatt (Ordnance Survey data ©Crown copyright, Ordnance Survey, all rights reserved, NC/01/586.

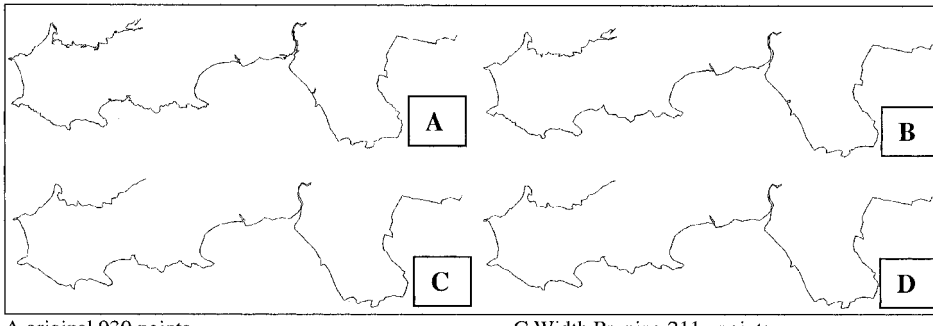
however a potential problem, particularly with datasets with multiple closely spaced linear features.

Again the 'base angle' metric discussed above is relevant as it prevents the creation of stumps by simply preventing such cuts occurring in the first place. This is not a satisfactory solution, as we wish to allow such cuts (which are not 'corners' in the sense described above), but to remove or smooth over the stump they leave behind.

A possible solution is 'resampling'. This involves adding additional points whenever a cut is made, to avoid creating excessively long line segments. The heart of the problem is that at present making such cuts creates line segments which may have



A original 930 points
B Visvalingam 159 points
C Width Pruning 162 points
D Width and Length Pruning 162 points



A original 930 points
B Visvalingam 214 points
C Width Pruning 211 points
D Width and Length Pruning 211 points

Figure 11. (*Continued*).

Table 1. Threshold values used for results in figure 11.

Num. points	width + dp	width + length + dp	Visvalingam & Whyatt
47	1390 + 276	1390 + 13 900 + 276	689 701.5
77	800 + 169	800 + 8000 + 169	281 247.5
114	600 + 120	650 + 6500 + 131	128 461.0
162	400 + 81	450 + 4500 + 90	59 120.5
211	300 + 61	300 + 3000 + 61	33 656.0

Values in metres except for Visvalingam in square metres. dp refers to Douglas-Peucker filters.

a large point spacing compared to the distance between the line and a neighbouring line. This is a problem discussed by Gold (2000) in relation to his skeleton retraction approach to line generalisation, and he addresses it by means of resampling to increase point density on the affected part of the line.

Resampling has been implemented, and appears to work satisfactorily, though at the expense of increasing execution time. Further testing is required to determine if the benefits justify this. Because the 'stump' problem does not appear with the real test data used so far, resampling has merely served to produce a slightly 'smoother' generalisation. However different cartographic data may show greater differences when resampling is enabled.

The principle is shown in figure 14. The new points lead to the creation of new

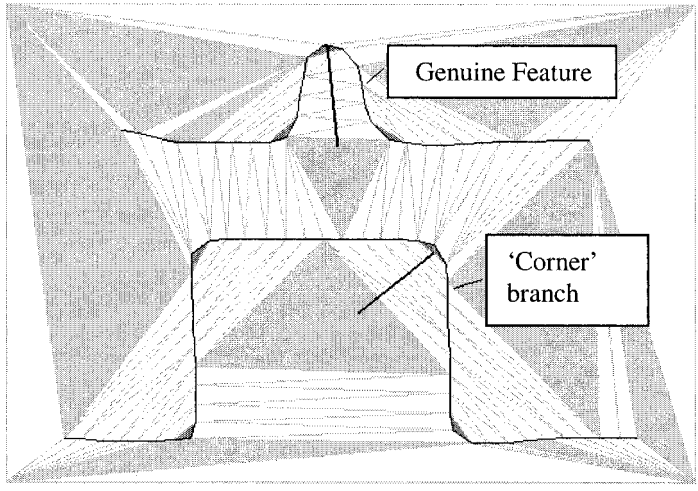


Figure 12. A corner.

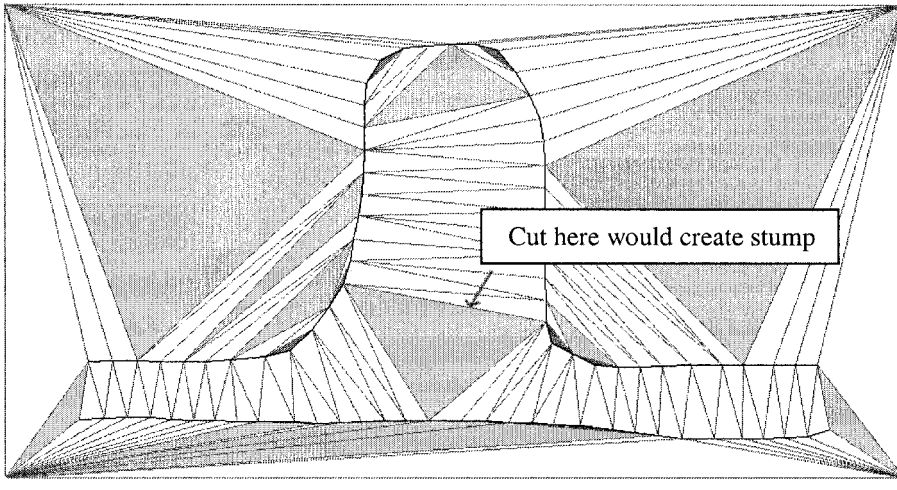


Figure 13. A stump.

smaller branches, the deletion of which smoothes the stump left by the deletion of the original branch. Further resampling (of the cuts created by the removal of the new branches) will increase the degree of smoothing of the bump.

6.3. *Option for one-sidedness*

A capability that might usefully be added, in addition to the metrics discussed previously, is to give users the option to declare lines, or segments thereof, to be single sided. This would mean that only branches identified on one (specified) side of the line would be considered for pruning. This would hugely simplify the processing, as the deletion of a branch would not require retriangulation and re-evaluation of the area on the opposite side of the line. More to the point, however, it would provide another option when determining the style of map to be produced. This might be particularly relevant when considering features such as coastlines

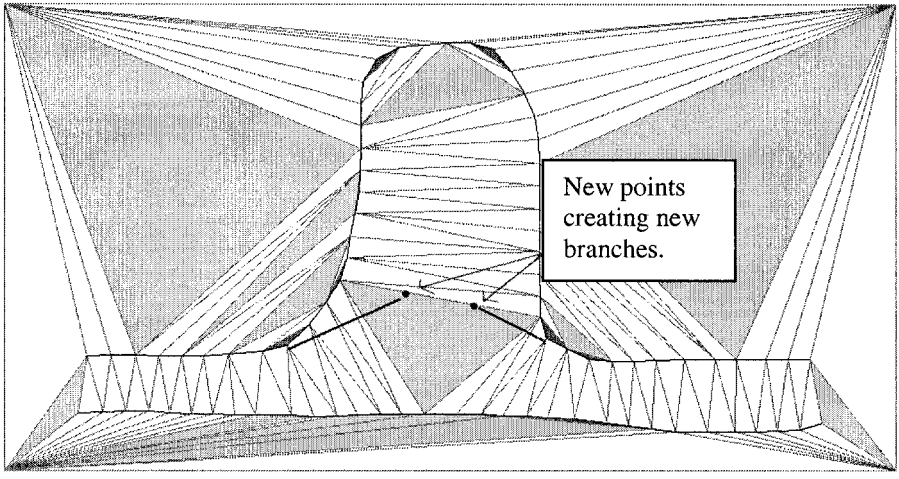


Figure 14. Resampling a stump.

in which promontories such as peninsulas only exist as areal features on the landward side.

6.4. *Logical combinations of metrics*

Another desirable addition would be to allow more flexibility in combining metrics. That is, while at present the system only allows multiple metrics using an AND operator (e.g. 'prune branches which are short AND narrow') it might be useful to be able to also use an OR (e.g. 'prune branches which are either short OR narrow'). One problem that would have to be overcome would be how to determine the order of pruning in this case.

6.5. *Networks and polygons*

A significant limitation of the system at present is that it works only for disjoint non-intersecting lines. Further work is needed therefore to extend the technique to handle networks and polygonal features. This will require introducing additional types of triangle and of path attribute, as well as addressing, in the case of networks, the possibility of moving nodes of the triangulation.

6.6. *Choosing metrics*

Perhaps the most challenging issue is the choice of metrics to control the procedure. At present it is envisaged that some kind of machine learning approach may be useful, whereby combinations of the available metrics are found that result in derived maps that match training examples of particular styles of map. The present work therefore concentrates on providing as large a degree of flexibility and control as possible.

7. Conclusion

Existing point-filtering algorithms are for the most part not designed for cartographic generalisation. Such generalisation requires understanding of the structure of the line that is to be generalised, allowing operations that are aware of the local geographical features. Previous attempts to address this issue focused on the detection of points of inflection and maxima of curvature. This paper has described an

alternative approach, which analyses the shape of line features in terms of the characteristics of local regions of a constrained Delaunay triangulation of the line. The structure of the triangulation is used to create a hierarchy of branches that correspond to line features at different levels of detail. Generalisation of the line is performed by pruning branches that meet user specified criteria, expressed in terms of a set of metrics associated with the branches. The method has the advantage of guaranteeing topological consistency. It also enables multiple lines to be generalised simultaneously. Of particular importance however is the fact that it offers the possibility of controlling the style of generalisation. This paper has demonstrated with real datasets that application of different metrics results in quite different types of generalisation, distinguishing, for example between the length and the width of features of the line. Further work includes extending the technique to work for networks of line features and finding methods for automatic specification of appropriate metrics to obtain a particular style of generalisation.

References

AI, T., GUO, R, ZHONG, G., and YAOLIN, L., 2000, A binary tree representation of curve hierarchical structure based on Gestalt principles. In *Proceedings of 9th International Symposium on Spatial Data Handling* (Beijing: International Geographical Union), section 2a, pp. 30–43.

BUTTENFIELD, B., 1985, Treatment of the cartographic line. *Cartographica*, **22**, 1–26.

DE BERG, M., VAN KREVELD, M., and SCHIRRA, S., 1998, Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Systems*, **25**, 243–257.

DOUGLAS, D., and PEUCKER, T., 1973, Algorithm for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, **10**, 112–122.

FERLEY, E., CANI-GASCUEL, M-P., and ATTALI, D., 1997, Skeletal reconstruction of branching shapes. *Computer Graphics Forum*, **16**, 283–293.

GOLD, C., 2000, Primal/dual spatial relationships and applications. In *Proceedings of the 9th International Symposium on Spatial Data Handling* (Beijing: International Geographical Union), section 4a, pp 15–27.

JONES, C. B., BUNDY, G. L., and WARE, J. M., 1995, Map generalisation with a triangulated data structure. *Cartography and Geographical Information Systems*, **22**, 317–313.

MCMASTER, R., 1987, Automated line generalisation. *Cartographica*, **24**, 74–111.

LEE, D. T., 1982, Medial axis transformation of a planar shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI**-4, 363–369.

MULLER, J. C., 1990, The removal of spatial conflicts in line generalisation. *Cartography and Geographical Information Systems*, **17**, 141–49.

PLAZANET, C., AFFHOLDER, J. G., and FRITSCH, E., 1995, The importance of geometric modelling in linear feature generalisation. *Cartography and Geographical Information Systems*, **22**, 291–305.

SAALFIELD, A., 1999, Topologically consistent line simplification with the Douglas-Peucker algorithm. *Cartography and Geographic Information Science*, **26**, 7–18.

THAPA, K., 1988, Automatic line generalisation using zero crossings. *Photogrammetric Engineering and Remote Sensing*, **54**, 511–517.

VAN DER POORTEN, P. M, and JONES, C. B., August 1999, Customisable Line Generalisation using Delaunay Triangulation. In *Proceedings of the 19th ICA Conference* (Ottawa: International Cartographic Congress), section 8, CD Rom.

VISVALINGAM, M., and WHYATT, J. D., 1993, Line generalisation by repeated elimination of points. *Cartographic Journal*, **30**, 46–51.

VISVALINGAM, M., and HERBERT, S., 1999, A computer science perspective on the Bendsimplification algorithm. *Cartography and Geographical Information Systems*, **26**, 253–270.

WANG, Z., and MULLER, J. C., 1988, Line generalisation based on analysis of shape characteristics. *Cartography and Geographic Information Systems*, **25**, 3–15.

WEIBEL, R., and DUTTON, G., 1999, Generalising spatial data and dealing with multiple representations. In *Geographical Information Systems:Principles*, Volume 1, edited by P. Longley, M. Goodchild, D. Maguire and D. Rhind (New York: Wiley), pp. 125–155.