Feature Based Line Generalisation using Delaunay Triangulation

P. M. van der Poorten
University of Cardiff
Email: scmpv1@cardiff.ac.uk
C. B. Jones
University of Cardiff
Email: cbjones@cardiff.ac.uk

Abstract: *A method of generalising linear cartographic features using an approach based on the determination of the structure of such features is proposed. This structure is determined by examining the space surrounding the feature. This is achieved using constrained Delaunay triangulation. A variety of statistical measures are used to determine the nature of parts of this structure corresponding to segments of the line, allowing the line to be intelligently generalised and allowing the user to specify the style of generalisation required. The triangulation is updated dynamically to allow both sides of multiple lines to be processed in acceptable time.*

Keywords: Cartographic generalisation, topological consistency, feature detection

Introduction

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 deciding both what features to include and how to represent them[2]. Features may require simplification if they are to be legible at a reduced scale, a process that has traditionally been performed manually by cartographers. It is highly desirable to be able to automate this process (and many methods have been proposed to do this[2,9]). One outstanding problem in this regard is the automatic generalisation of linear features.

A failing common to many of the linear 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 nature - the fact that it represents a real physical feature. The line may represent a road, a coastline, or a river, say -- 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 incapable of achieving effective cartographic generalisation.

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) [8,10,12,14]. 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 many critical point methods do not exactly correspond to those that would be identified as such by a human observer. Additionally, some form of smoothing is required in order to obtain a hierarchy of features at different scales.

Visvalingam[12,13] takes a different approach, based on examining the area of triangles formed by consecutive triplets of line vertices. This goes a good distance towards identifying features, which it tends to selectively remove, in an order determined by their area. This approach is perhaps closest in its aims to the method outlined in this paper.

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 artifactual intersections between lines or parts of the same line, and this usually has to be cleared up with ad hoc post-processing[7].

This paper explores an alternative approach to the problem of identifying sub-features, expanding the ideas outlined in *van der Poorten and Jones 1999*[11] and *Ai et al 2000*[1]. The 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 keeping processing times down to acceptable levels. 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 ready for processing is then described in detail and the various ‘metrics’ used to determine the style of generalisation are defined.
- A more detailed description is then 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.

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) 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[5]. 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 essential procedure is to enclose the line(s) to be generalised in a containing box and apply a constrained Delaunay triangulation on the resulting area. The constraint is that the line segments making up the given lines and the bounding box must be retained as edges within the triangulation.

Figure 1 shows a sample line feature and corresponding triangulation. Leaf triangles are shaded dark grey, trunk triangles are white and branching triangles are light grey.

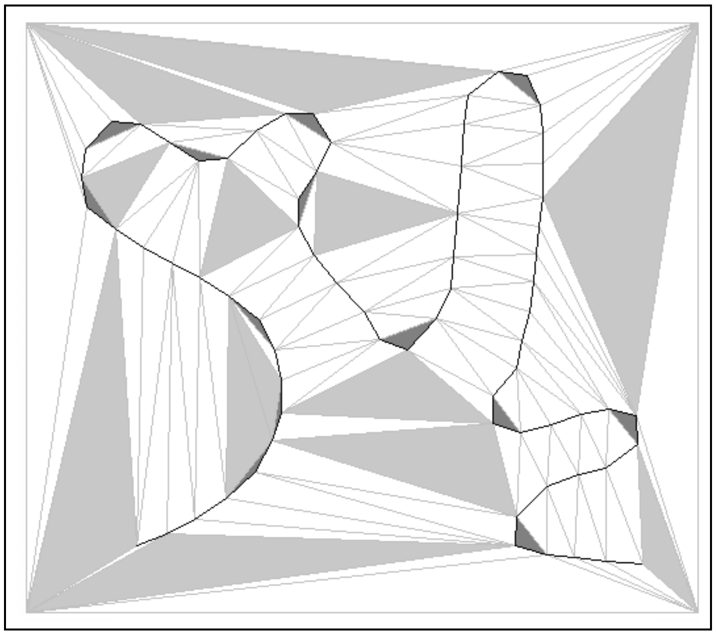

Figure 1 - A Triangulation.

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 - i.e. 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[3,4]. 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.

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 for purposes of line generalisation. An important characteristic of features composed of sets of triangles, is that, provided the vertices of the respective triangles belong to a single line, their elimination is guaranteed to avoid topological inconsistencies. This is because by definition the triangles cannot contain any other geometry and hence their collapse cannot result in overlap of other features.

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 2B** shows a complex branch with sub-branches.

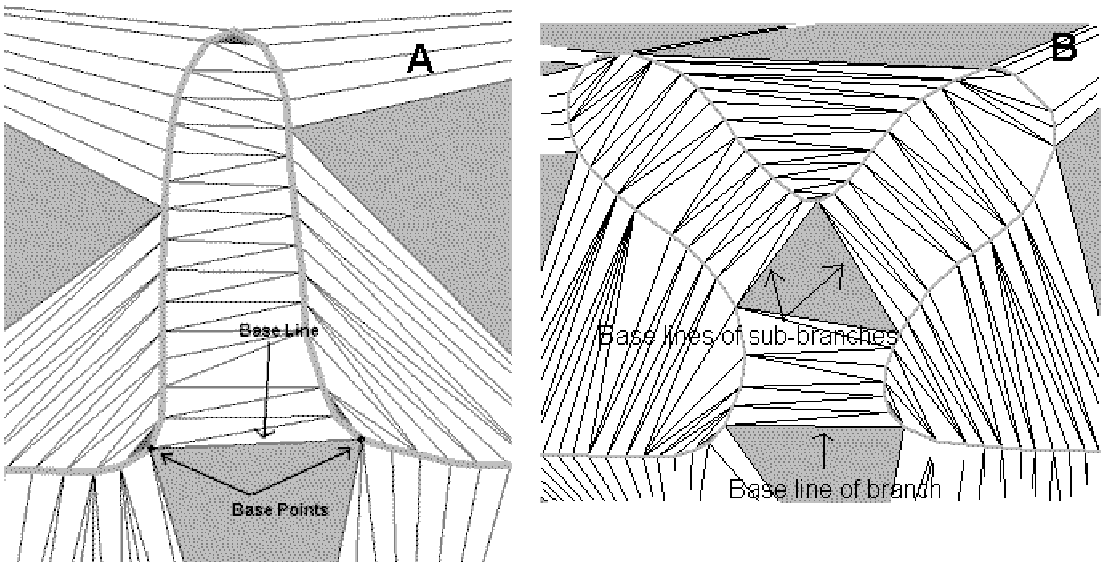

Figure 2– a feature, a branch, and its base line

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 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.

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 dashed line indicates a neutral exit, a black one a leafward exit and a grey one a rootward exit. Triangles are coloured here not according to the number of edges, but to the number of edges with *leafward* exits. This is significant in identifying 'root' triangles.

The triangles with exactly one such edge (medium grey) are the 'root' triangles of branches. Triangles with two such edges (dark grey) are internal branching triangles, while those with three neutral edges (light grey) are external branching triangles and occur only when more than one line is present.

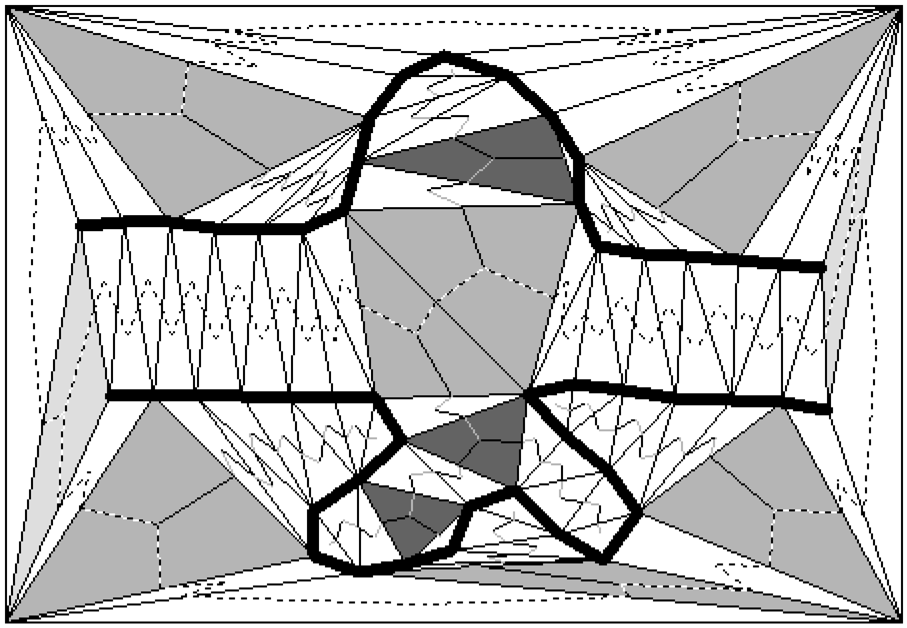
Figure 3 Triangulation with Pathset attributes

Ai et al[1] extends *van der Poorten and Jones*[11] with the addition of a fourth triangle type, but here we retain the three basic types listed above but with the addition of a further sub classification depending on the pathset attributes of the given triangles. This gives rise to a total of five types of triangle (leaf, trunk, internal branching, root, and external branching).

A further 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.

The pathset attributes are initially calculated using the following procedure.

First all the leaf triangles are located. 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 branching 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 then 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 then 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.

Once the triangles and pathset attributes have been so categorised (and marked) we now 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 line of that branch (or sub-branch).

The statistics (or ‘metrics’) so calculated 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

For most of these metrics their value for a particular branch or sub-branch is partly determined by summing the values of any constituent sub-branches it may have (plus the contribution from the remainder of the branch).

For simple branches (or sub-branches) with no sub-branches the majority of these metrics are obtained by first calculating the contribution of each triangle in the triangulation, and then by adding up the contribution of the triangles in the branch. This is done for reasons of efficiency, so that changes in the 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.

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).

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 first define a 'node length' for each triangle in the triangulation. The node length of a trunk triangle is defined to be 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 - the distance from the centre of the rootward edge to the centroid of the triangle, plus the distance from there to either of its leafward edges.

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 (see **figure 4**). However, unlike when measuring the skeleton, the starting point is taken as the base line of the branch, not the point at which the skeleton branch connects with the parent branch. 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. See **Figure 4b.**

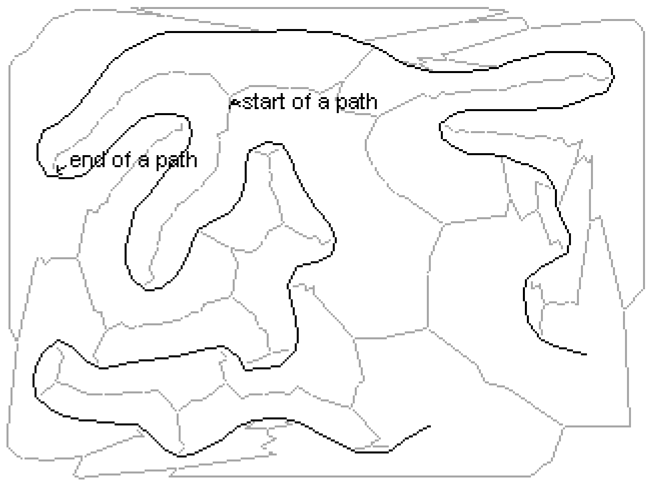

Figure 4 – path length of branches

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, requiring use of this definition but eventually we will reach simple branches and can start working back up again. (Essentially the process is recursive). See **Figure 4c.**

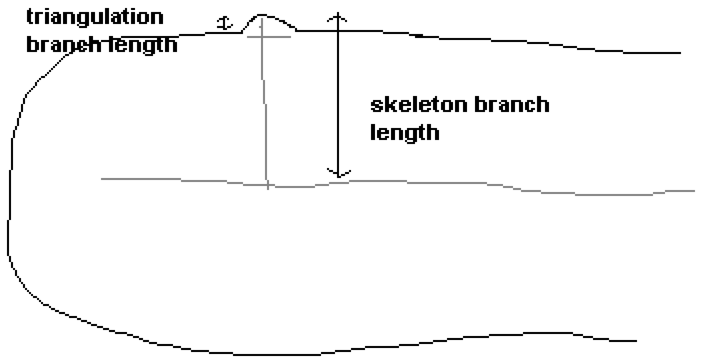

Figure 4b – branch length vs. skeleton branch length

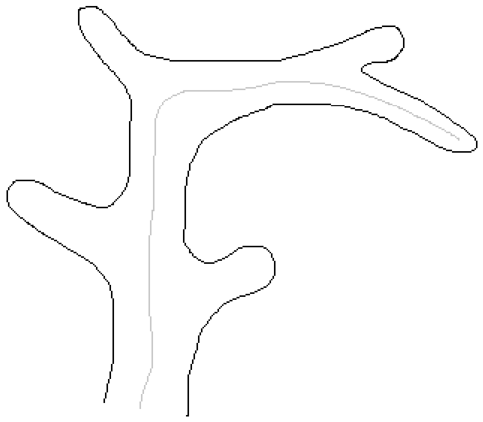

Figure 4c – path length of complex branch

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. An exception is made for leaf triangles where the whole height is used. The usefulness of this metric (as opposed to the branch length, as defined above) is still under investigation. However it is used in calculating the average width, as explained below.

The average width of a branch is defined to be the total area divided by the branch height. This is used for complex branches as well, where 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. This seems to be a reasonable way of regarding complex branches with regard to width. In effect it is a weighted sum of the average widths of all the triangles in the path.

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*[1] 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 line 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 use of these different metrics allows decisions to be made about processing the line so as to achieve different styles of generalisation. At present this simply allows 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 more circularly shaped features will be more likely to be removed than less compact ones of the same area. The metrics 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.

The Generalisation procedure

Generalisation processing by selective pruning of branches is accomplished using the following procedure.

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 then retriangulated and the branch sizes are recalculated.

The process is then repeated until the smallest remaining branch (measured by the chosen metric) is below the given threshold value.

In general one would specify thresholds for each possible metric and remove all branches that fall below *any* 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 lines that constitute 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 the 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 chosen to be *used*.

Note that because this procedure uses 'dynamic' retriangulation, only updating the mesh in the area affected by the removal of a branch, it can deal with lines in a genuinely two-sided fashion. *Ai et al*[1] only uses a single sided approach, while *van der Poorten and Jones*[11] did use a two sided approach but at the expense of having to retriangulate the entire region at each step, a computationally expensive procedure.

The procedure used is optimised for efficiency, and a more detailed description of how it works is as follows.

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' (triangles unaffected by the point deletions) and sections of undeleted line segments (imposing constraints on the retriangulation).

In order to avoid doing unnecessary work, only those branches that have been affected should have their statistics recalculated. To achieve this it is necessary to find which branches include the affected triangles. In order to do this, however, we need 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, and 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 in 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 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 {L R} or {N N}.

- If a triangle has three such edges they can only be one of the following combinations:
{N N N}, {N N L}, {L L R}

There is one awkward case where the process may become stuck, 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 we can easily retrieve the situation once we have detected that no progress is being made.

Once the path type 'map' has been filled in, 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.

Benefits of an 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 type methods having multiple parameters to tune is often seen as 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 discussed here, it is possible for the user to specify a particular style of generalisation, even with just the simple pruning routines so far implemented.

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[7]. With an area-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 6** (figures **A** to **F** showing progressively larger degrees of generalisation, corresponding to increasing values of the pruning threshold, in this case area)**.**

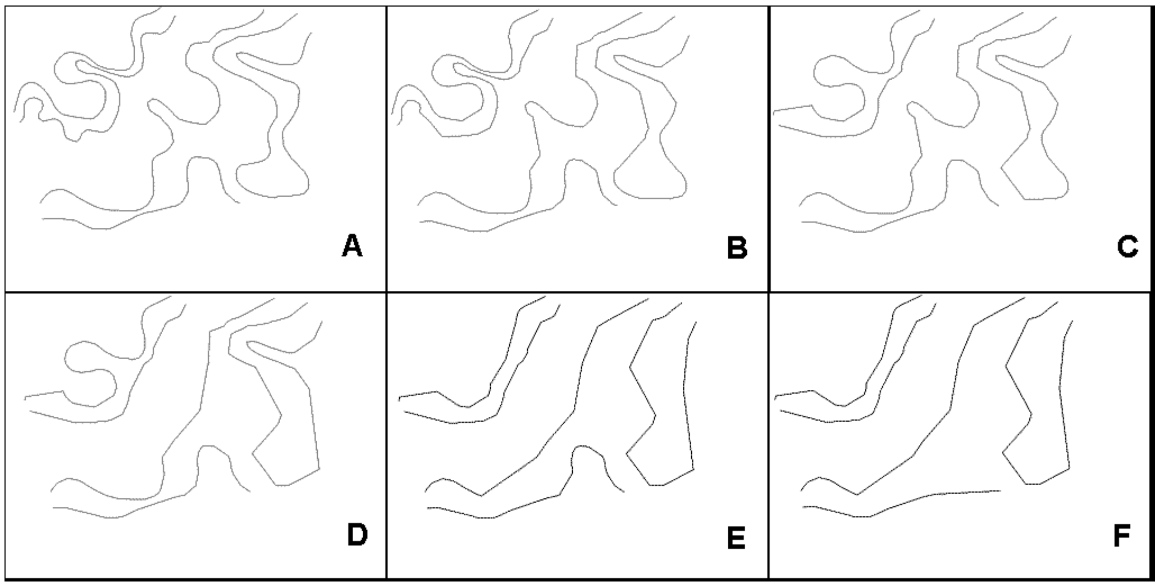

Figure 6 – preservation of topology

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 this is only possible if only one metric is used at a time, though any number of single metrics could be used by pre-processing the data with each separately and labelling the vertices with values for each.

Sample Results

Some example results (using synthetic data) are shown in **figures 7A-7E**. **Figure A** shows the original line. **Figure B** shows the same line subjected to 'pruning' where the constraining metric is the branch area. **Figure C** shows the same line pruned using the same metric with a higher threshold value. In **figure D** the branch width is used, while in **figure E** the pruning is performed by path length.

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.

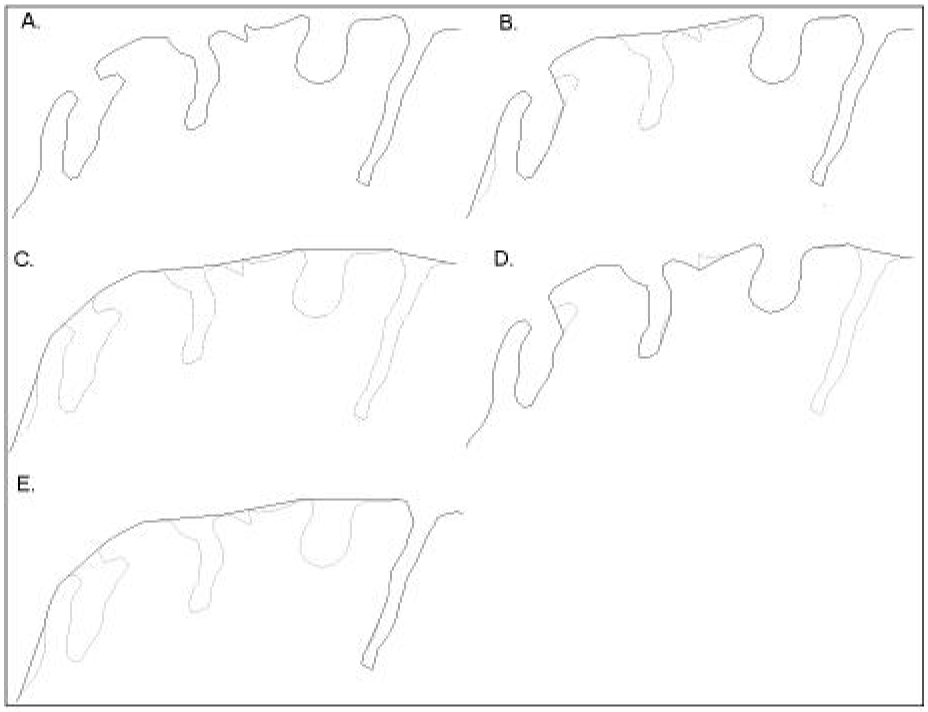

Figure 7 - alternative Generalisations.

Figure 8 shows some real data (part of the Gower county boundaries, including a large 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.**

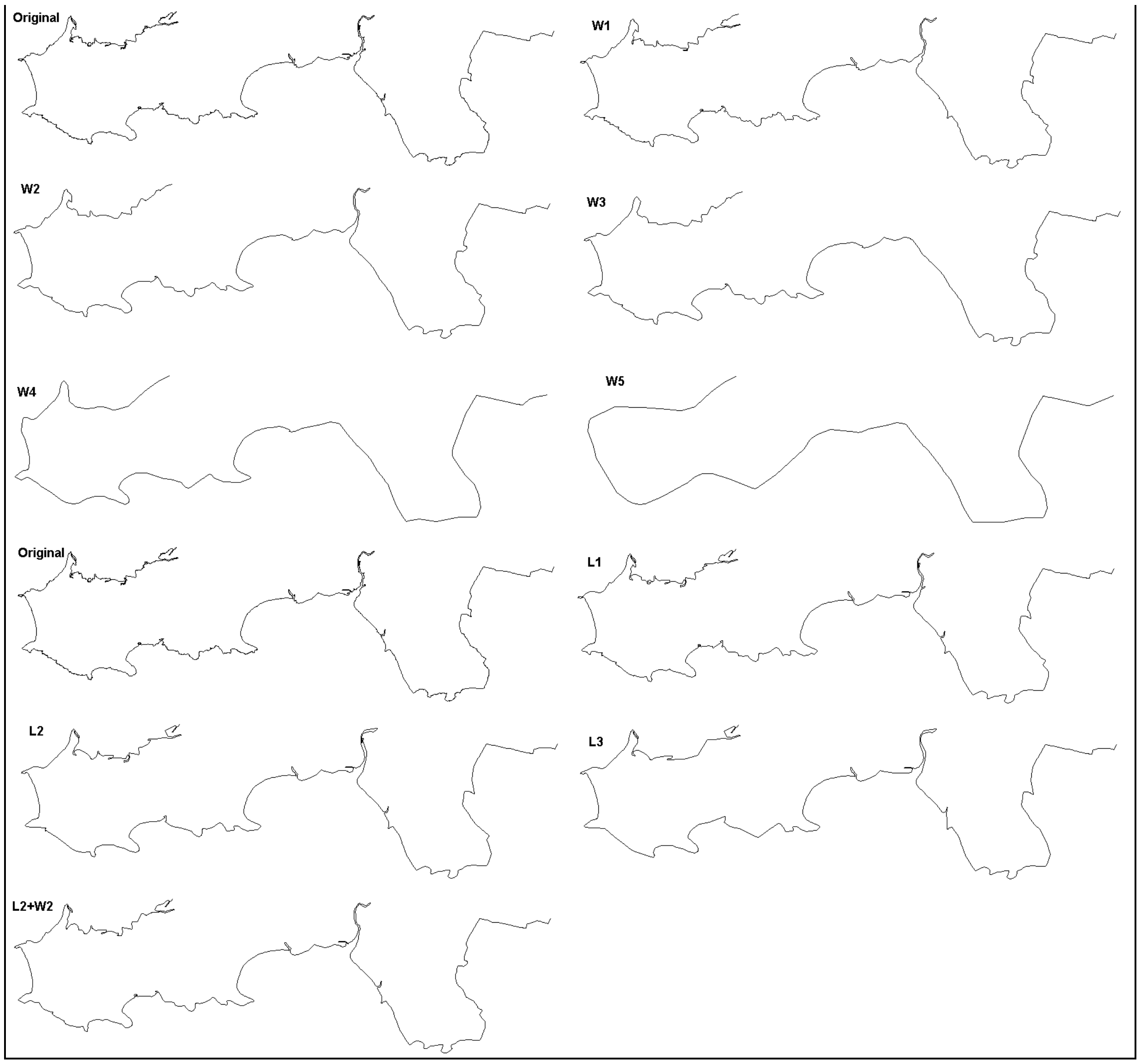

Figure 8 – Real data, alternative generalisations

Figure 9 shows a close-up of part of the same data. The labels have the same meaning as for **Figure 8.**

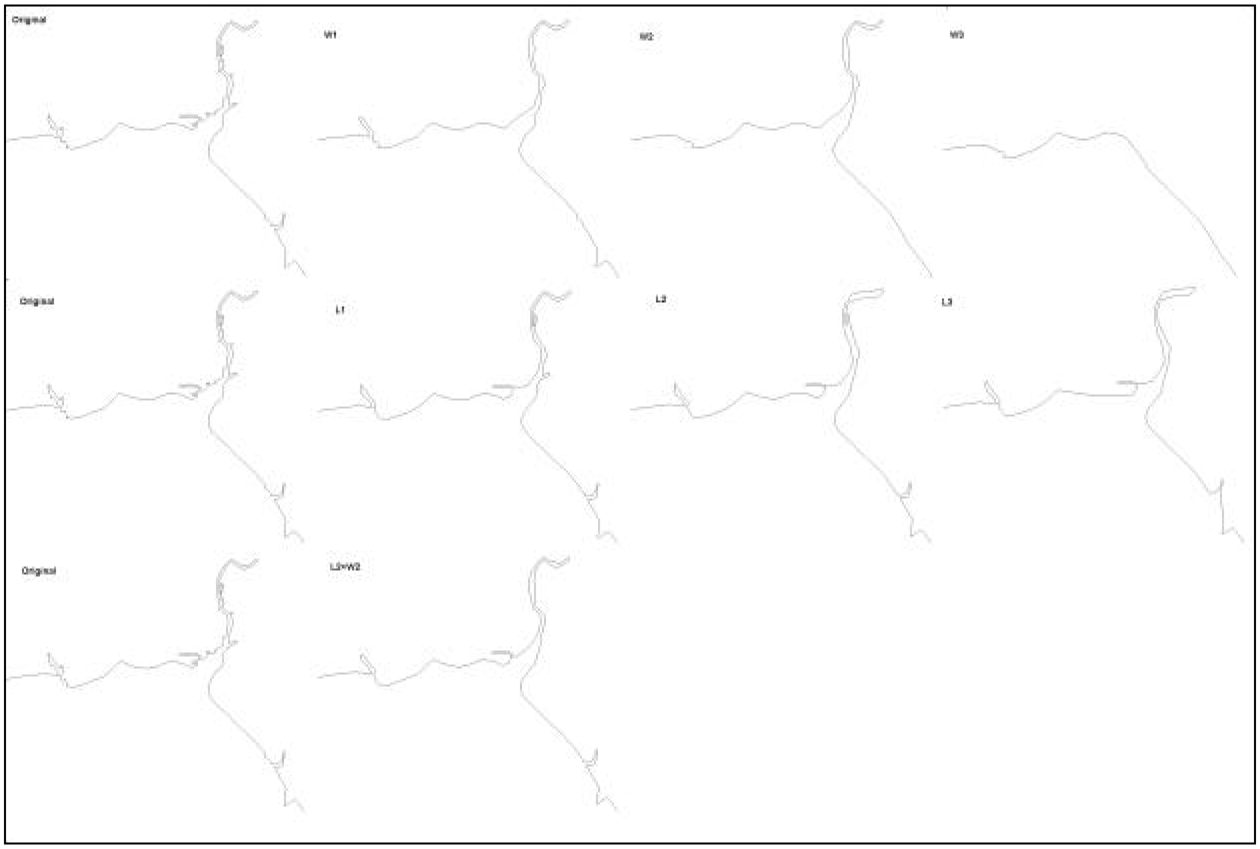

Figure 9 – Detail from Figure 7

Figure 10 shows a comparison of data generalised with Douglas-Peuker only (**10A)** and data generalised using branch pruning by average width combined with a post-generalising point-filtering step using Douglas-Peuker (**10B**). This is necessary as branch-pruning is *not* intended as a point reduction algorithm (it will not remove completely co-linear points, for example). In both cases the resulting generalisations are using 4% of the original points.

Figure 11 shows the same data as **Figure 10** generalised using the method of Visvalingam. **Figures A-F** show results using 20,10,6,5,3 and 2 percent of the original points respectively. This method *does* selectively prune 'features'. It finds 'features' 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'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.

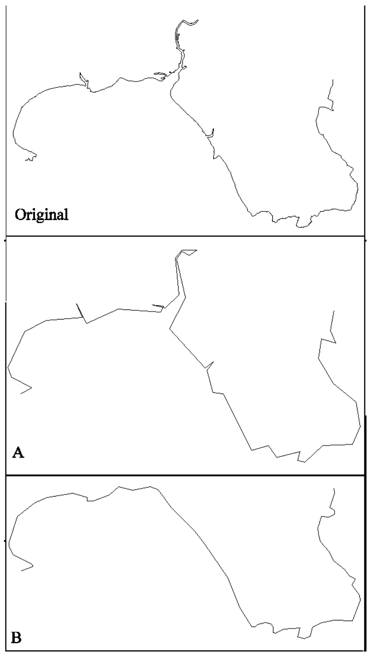

Figure 10 – Douglas Peucker vs Branch Pruning plus Douglas-Peucker

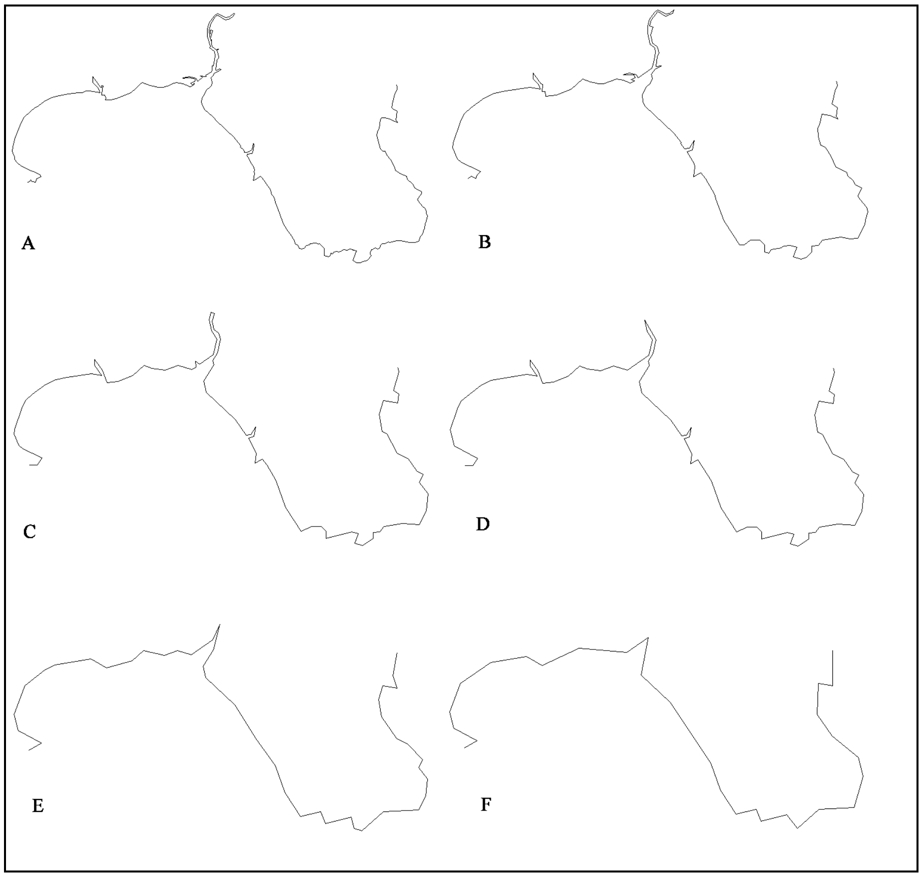

Figure 11 – Visvalingam's method

Possible Further work

A feature that could be added, in addition to the metrics discussed above, is to give users the option to declare lines, or segments thereof, to be single sided. What this would mean is that only branches identified on one (specified) side of the line would be considered. 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 in which promontories such as peninsulas only exist as areal features on the landward side.

A further obvious metric to use would be minimum width and an additional development would be to allow for widening branches that fell below a certain width threshold either in order to meet a legibility criterion or as part of a typification operation. For example, the estuary in **Figure 8** and **9** could be widened rather than deleted. This would require, however, a means of ensuring such an adjustment did not reduce the width of the neighbouring branches below the allowable threshold. The triangulation itself might provide a means of achieving this, with displacement of points radiating out via the virtual edges of the triangulation.

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 (‘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.

A significant limitation of the system at present is that it does not provide for networks. That is, only non-intersecting, disjoint lines can be generalised. In principle it should be possible to extend the approach to networks but aside from the modifications to the mechanics of the implementation, there are also conceptual issues to be addressed. These include how to treat intersection points (can they be moved?) and what is the most desirable way to maintain connectivity. Much additional work is needed here. An extension to enable the system to generalise closed polygons is, however, close to completion. This requires the addition of further triangle types and path attributes but is otherwise straightforward.

The largest unresolved issue however, is how to determine which metrics to use. At present it is envisaged that some kind of interactive machine learning approach might be useful, whereby a human cartographer ‘trains’ the system to select appropriate combinations of metrics for particular styles of generalisation. The present work therefore concentrates on providing as large a degree of flexibility and control as possible.

Conclusion

Existing point-filtering algorithms are for the most part not capable of cartographic generalisation. Such generalisation requires a method capable of identifying the structure of the line that is to be generalised, allowing operations that are aware of the local geographic features to be applied to it. Existing work aimed at analysing line structure relies on the detection of such features as points of inflection and maxima of curvature. This paper describes an alternative approach, which perhaps offers some advantages over inflection-point type methods, including the ability to maintain topological consistency implicitly, and the ability to offer control over the *style* of generalisation desired.

References

1. Ai, Tingua, Guo, RenZhong, Guo and Yaolin, Liu, A Binary Tree Representation of Curve Hierarchical Structure Based On Gestalt Principles, Proceedings 9th International Symposium on Spatial Data Handling, Aug 2000, sec. 2a, pp 30.
2. Buttenfield, Barbara, Treatment of the cartographic line, Cartographica Vol. 22, No. 2 1985, 1-26.
3. Ferley, Eric, Cani-Gascuel, Marie-Paule, Attali, Dominique, Skeletal Reconstruction of Branching Shapes, Computer Graphics Forum, Volume 16 (1997) number 5, pp 283-293.
4. Gold, C., Primal/Dual Spatial Relationships and Applications, Proceedings 9th International Symposium on Spatial Data Handling, Aug 2000, sec. 4a, pp 15.
5. Jones, C.B., Bundy, G. L. and Ware, J.M., Map Generalisation with a Triangulated Data Structure. Cartography and GIS vol 22 (1995) no 4, pp 317-31.
6. McMaster, R., Automated Line Generalisation, Cartographica, 24 (2), pp 74-111.
7. Muller, J.C., The Removal of Spatial Conflicts in Line Generalisation, Cartography and Geographical Information Systems, Vol. 17, No. 2, 1990, pp 141-149.
8. Plazanet, Corrine, Affholder, Jean Georges and Fritsch, Emmanuel, The Importance of Geometric Modelling in Linear Feature Generalisation, Cartography and Geographical Information Systems, Volume 22 No. 4, 1995, pp 291-305
9. Shea, S. and McMaster, R., Cartographic Generalisation in a digital environment: how and when to generalise, Auto-Carto 9, Baltimore, ACSM/ASPRS, pp 56-67.
10. Thapa, K. Automatic Line Generalisation Using Zero Crossings, Photogrammetric Engineering and Remote Sensing, Volume 54, No. 4, April 1988, pp 511-517.
11. Van der Poorten, P.M, Jones C.B., Customisable Line Generalisation using Delauney Triangulation, CD ROM proceedings of the 19th ICA conference in Ottawa, Aug 1999, section 8.
12. Visvalingam M., Herbert S., A Computer Science Perspective on the Bendsimplification Algorithm, Cartography and GIS, Vol 26, No4, 1999, pp253-270.
13. Visvalingam, M, Whyatt J D "Line Generalisation by Repeated Elimination of Points", Cartographic Journal., 30 (1), 46–51
14. Wang, Z. and Muller, J.C., Line Generalisation based on analysis of shape characteristics, Cartography and Geographic Information systems Vol. 25 No. 1, 1988, pp3-15.
15. Lee, D.T, Medial Axis Transformation of a Planar Shape, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 4, July 1982.