# Feature Based Line Generalisation using Delaunay Triangulation

Peter van der Poorten and Christopher Jones

University of Cardiff

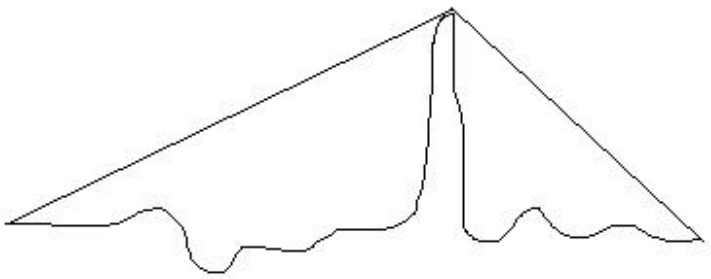
# 1/ Problems with most existing methods

Drawbacks of many existing methods?

- They are point filtering ones

- They don’t notice features and can distort them in an uncontrolled fashion e.g.

- Douglas Peucker can hang on to end points of features at the expense of grossly distorting everything else



Grossly unfair example of shortcomings of D-P

# 2/ Beyond Point Filtering

Finding features, allowing intelligent decisions to be made about what to do with them.

How to find them?

- Looking for points of inflection
  - requires work to get from inflections to features

- Smoothing by gaussian, either :

  smoothing the curve and then finding critical points, or
  smoothing the curvature signal and finding zero crossings

What is done with them after they are found is another question.

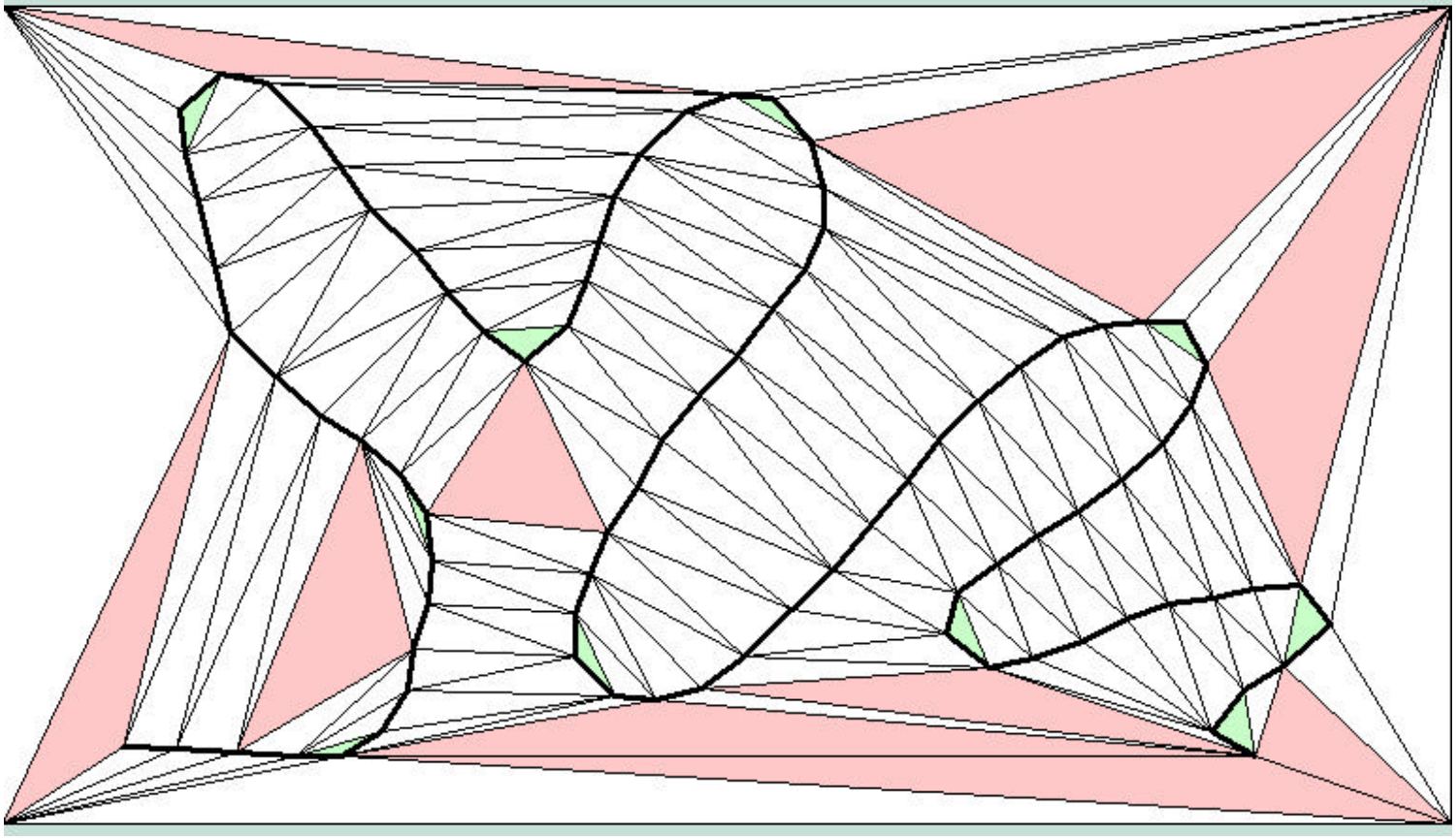
# 3/ Alternative Approach

Looking at the space defined by the lines -

- Can avoid topological inconsistencies implicitly
- Can detect features directly

How to examine the space?
- Use Delauney Triangulation

# 4/ Constrained Delauney Triangulation



# 5/ Finding ‘Branches’

How to find the ‘branches’?

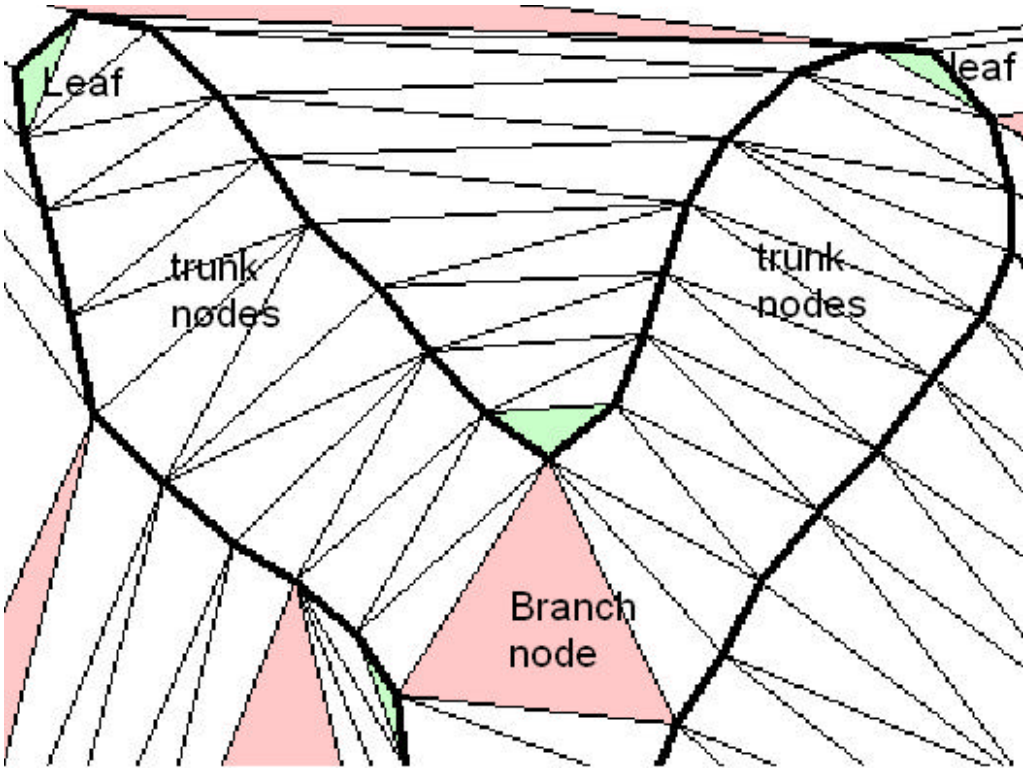
Simply examine each triangle in the triangulation and determine the number of (internal) neighbours

(Internal neighbour: an adjoining triangle whose shared edge is not part of the original line.)

1 neighbour – a leaf triangle
2 neighbours – a trunk triangle
3 neighbours – a branching triangle

(actually turn out to need more triangle types than this…)

follow the paths from each branching triangle.

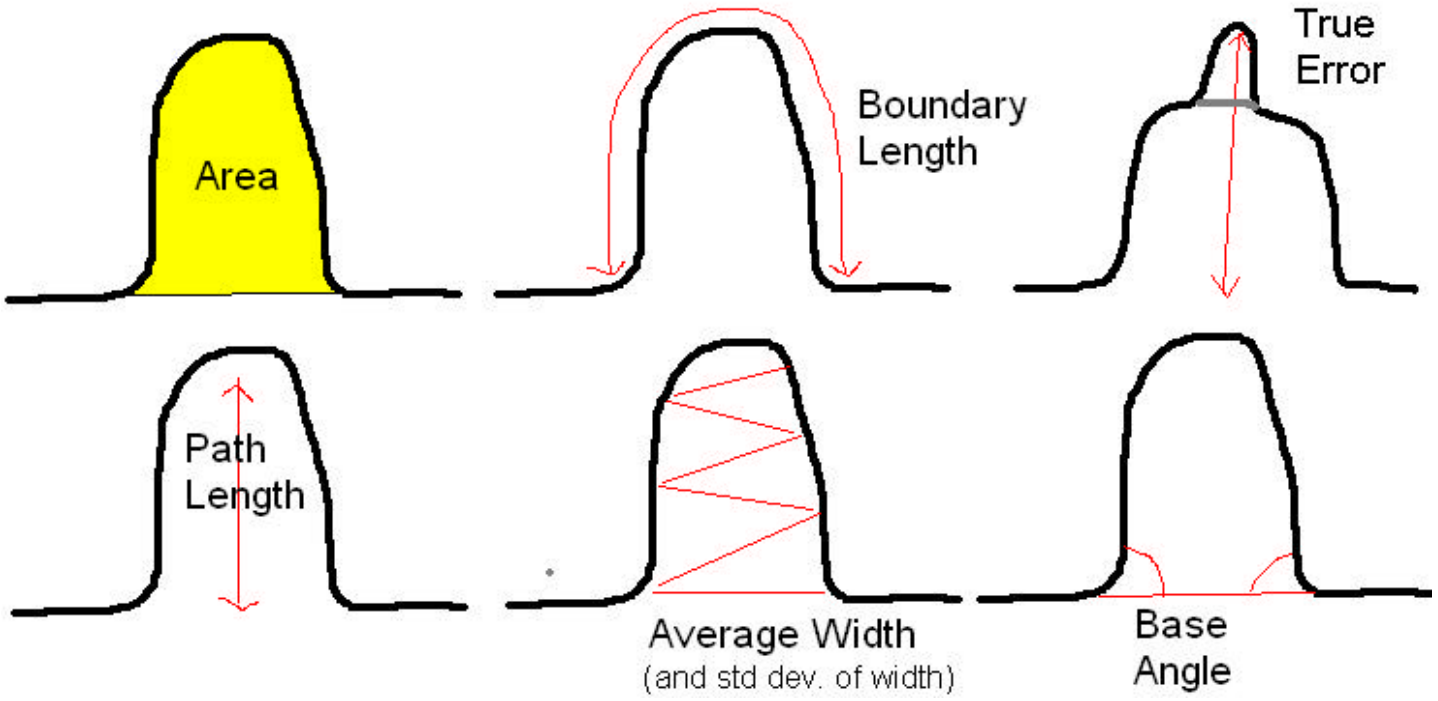

Leaf
leaf
trunk
nodes
trunk
nodes
Branch
node

# 6/ Characterising Features

What to do with features once they are identified?
First find suitable metrics with which to measure them

**Metrics:**

- Branch Area
- Length of the Boundary of the Branch
- Length of the Branch
- Average Width of the Branch
- Std Dev of the Width of the Branch
- The base angle of the branch
- True Error metric (compares with ORIGINAL unprocessed line)



# 7/ What to do with features

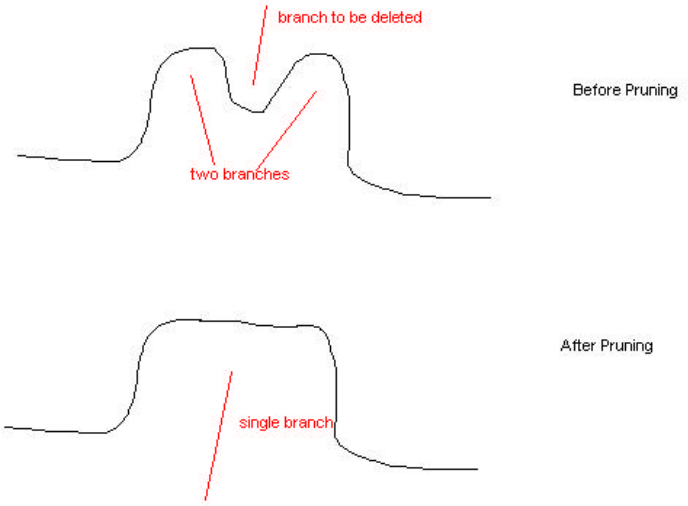
Having found features, we can:

- Remove features, based on a combination of their characteristics
- Merge features
- Exaggerate them, increasing the width of those which are too narrow
- Caricature them – for example where several similar bends are found close to each other they can be replaced by a smaller number of qualitatively similar bends.

  Currently only the first of these is performed, in what we have called ‘branch pruning’

# 8/ The Method

Crucial issue – when we remove a branch, the triangle mesh structure changes.




Important consideration – computational efficiency.

Previous approach – complete retriangulation each step. This required several hours to process ~1000 point dataset. (Never manged to complete process with larger datasets.)

New approach – can process 30,000 points in 5 minutes.

# 9/ Path Types

In order to do the minimum amount of recalculation with each iteration of the pruning algorithm, we need to analyse the structure in terms of ‘path types’, and to assign ‘node types’ to the triangles themselves.

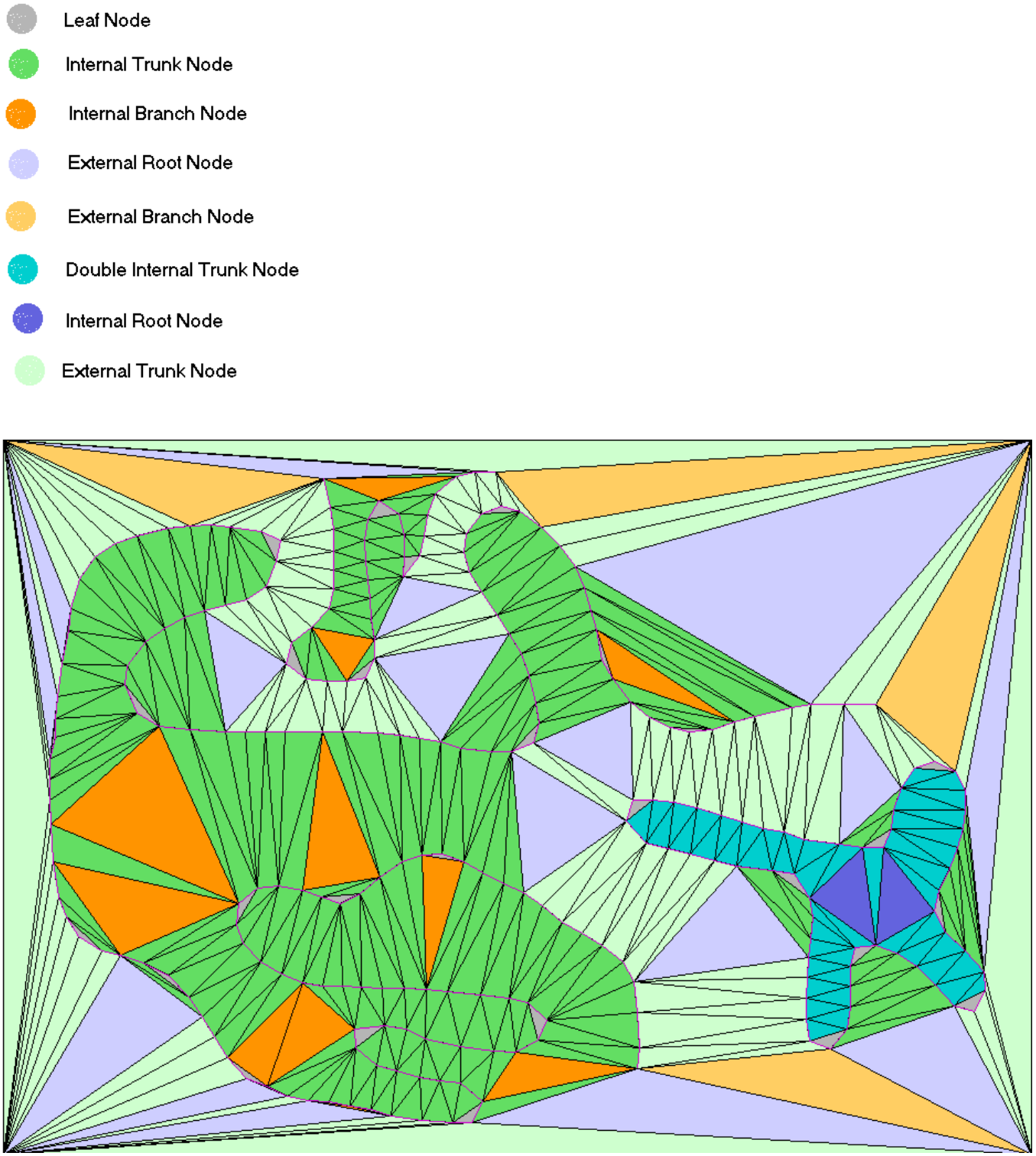
Insufficient time to give justification for all different types. Quick summary only.

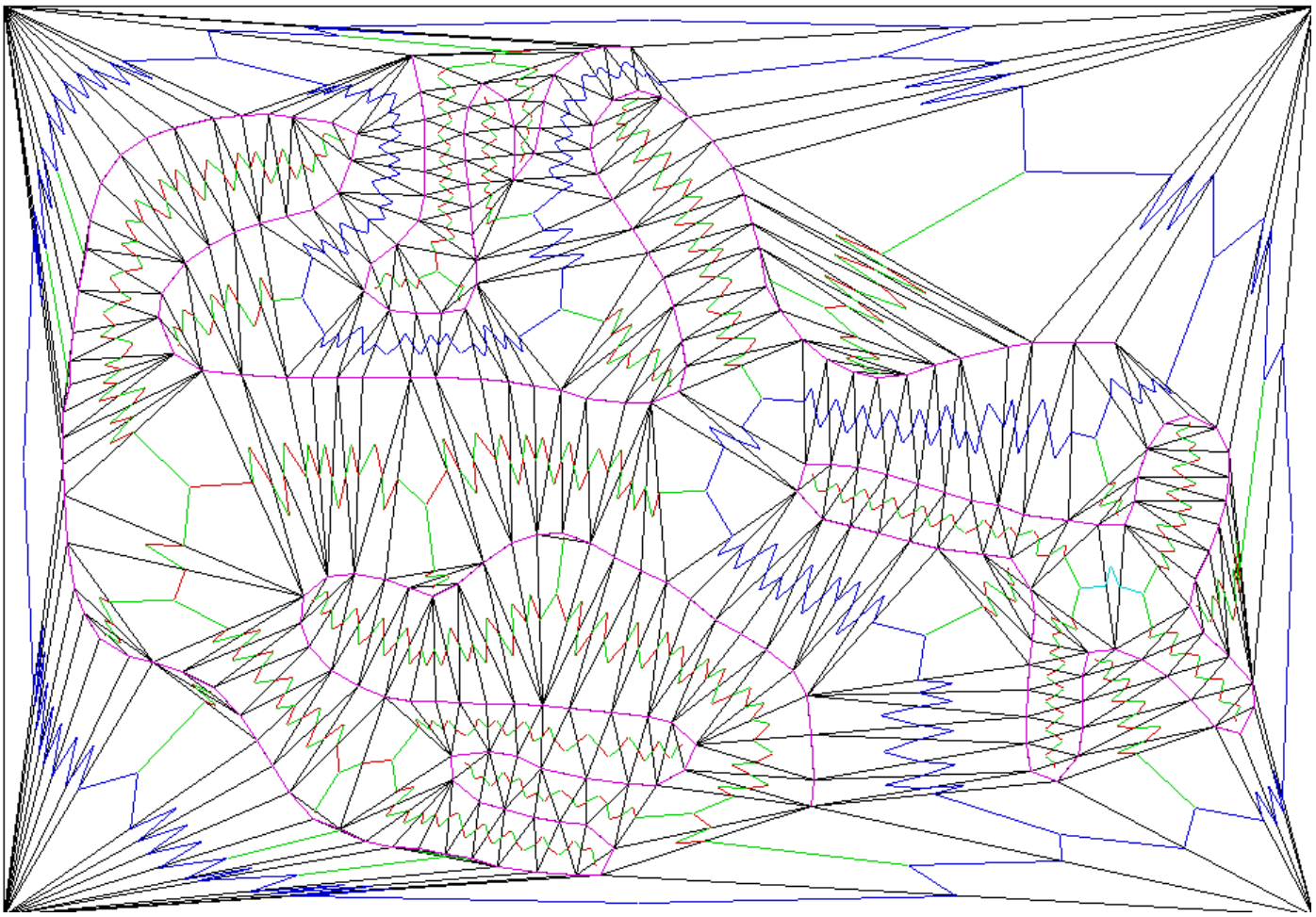
Path Types:

- Leafward - direction towards the end of a branch
- Rootward - direction towards the base/root of a branch
- Internal Neutral - path within a closed polygon
- External Neutral - path between different lines

Node Types:

- Leaf - end node of a branch
- Internal Branching - where a branch splits into sub branches
- External Branching - where external branches split/merge
- Internal Root - a branching node within a closed polygon
- External Root - node where a branch starts
- Internal Trunk - node within the body of a branch
- External Trunk – node between different lines
- Double Internal Trunk - node within a closed polygon


Leaf Node
Internal Trunk Node
Internal Branch Node
External Root Node
External Branch Node
Double Internal Trunk Node
Internal Root Node
External Trunk Node



Leafward path

Rootward path

Neutral path

Internal Neutral path

Essential point:

Using these Node and Path types allows us to **efficiently** find which paths are affected by the removal of any given set of points.

Then know which paths need to have their statistics recalculated.

We can trace back following ‘rootward’ paths from the affected nodes to the base of the affected path.

We can also quickly fill in the path types and node types of new or altered nodes.

# 10/ The Algorithm

## Old Algorithm

Set threshold values for all metrics
(e.g. min acceptable length, width, etc)
and choose one metric as primary

Repeat until the smallest branch satisfies the criteria
{
    Triangulate the entire area
    Find and Measure all the branches
    Delete the smallest branch, *by primary measure*
}

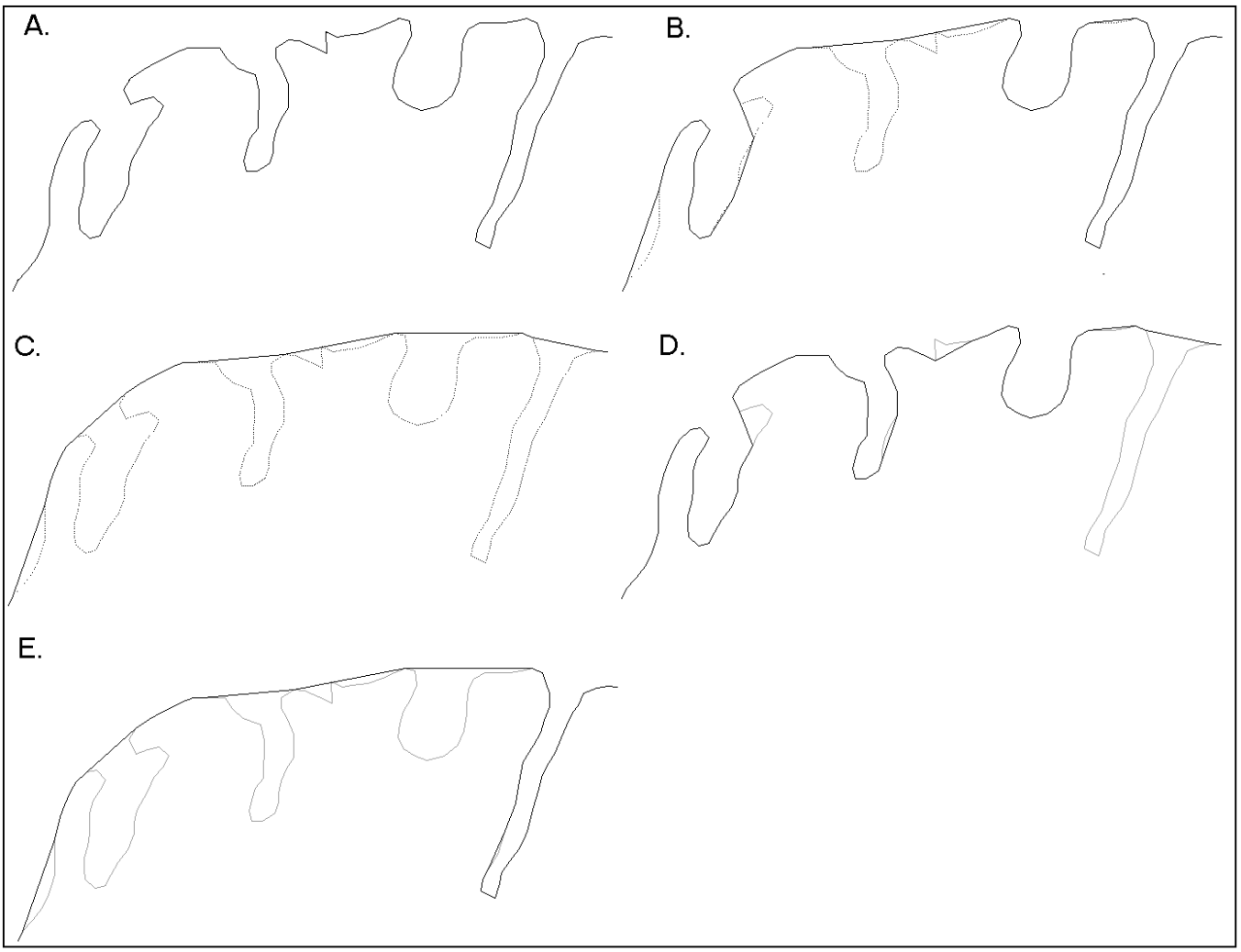
## Optimised Algorithm

Set threshold values for all metrics
(e.g. min acceptable length, width, etc)
and choose one metric as primary

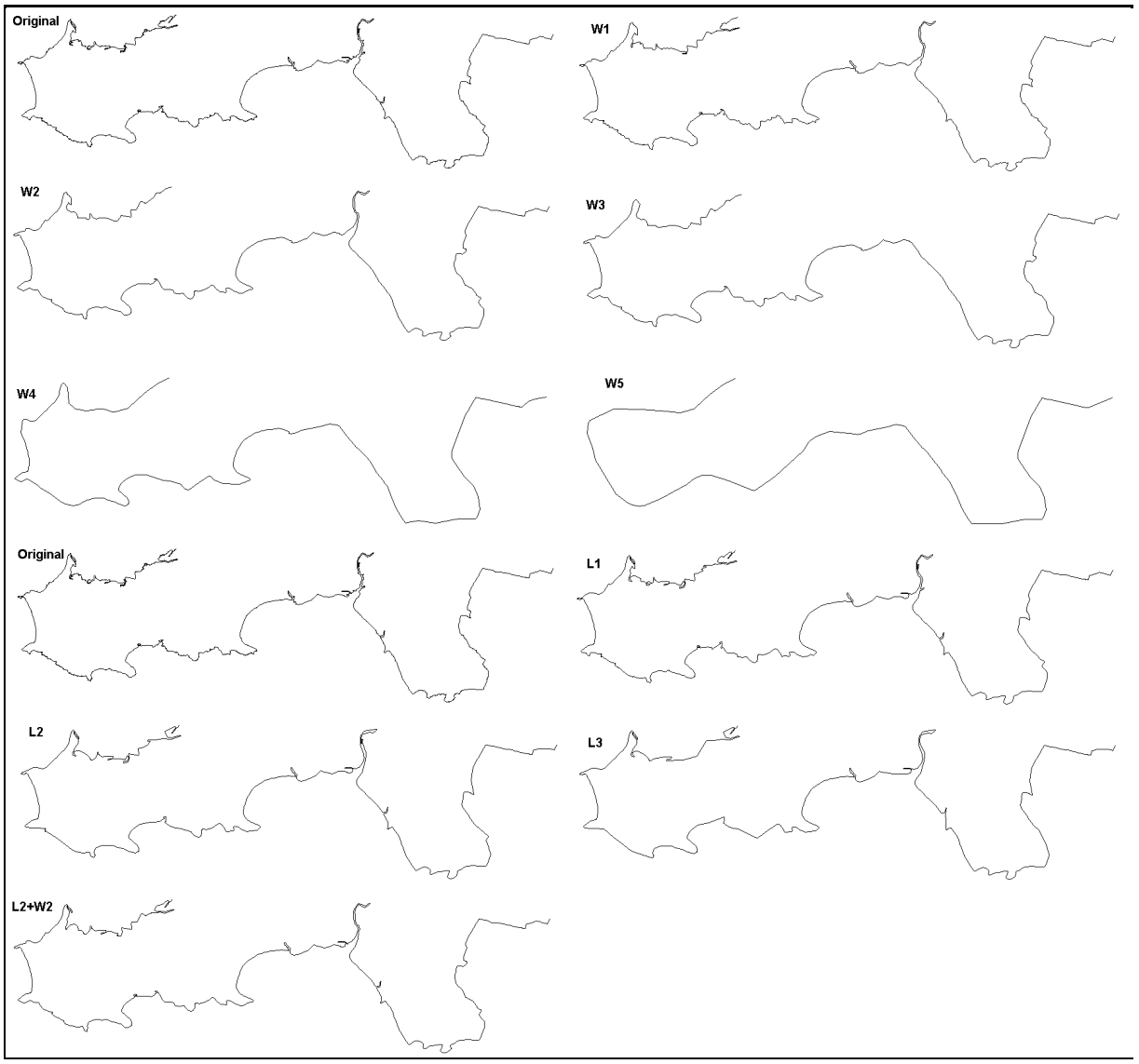
Triangulate the entire area
Determine the nature of each triangle or *Node*
Determine the structure of the branches
Fill in the *Path Types* for each *Node*
Find and Measure all the branches

Repeat until the smallest branch satisfies the criteria
{
    Delete the smallest branch, *by primary measure*
    Delete the region of the triangulation affected
    Retriangulate the affected region
    Determine the *Path Types* for the new or changed *Nodes*
    Determine which branches have been affected
    Remeasure the affected branches
}

# 11/ Results




**A** original line.
**B** pruned by branch area.
**C** pruned by area with larger threshold value.
**D** pruned by branch width.
**E** pruned by path length.




Progressively generalised by Width (W), Length (L) and by combination (L+W)

# 12/ Benefits

Advantages of method:

- Preserves topological consistency for a set of lines that are generalised together.
- Detects sections of the line that a human observer might select as significant.
- May allow a choice of different *styles* of generalisation, by allowing a choice of different metrics for describing features.

# 13/Remaining Issues

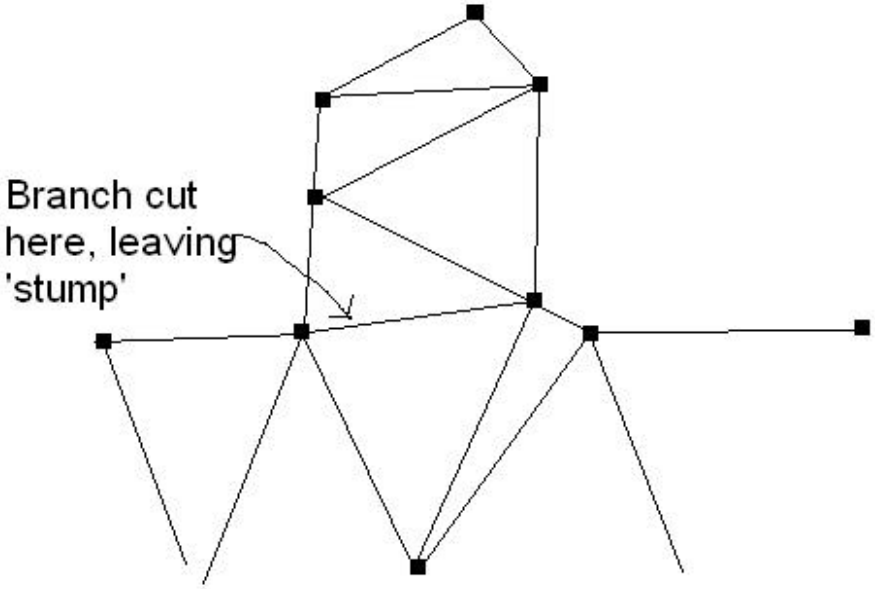
1/ Stumps.

2/ Differentiating between Corners and bumps.

3/ Two-sidedness of line.

4/ Funnel features.

1/Stumps:



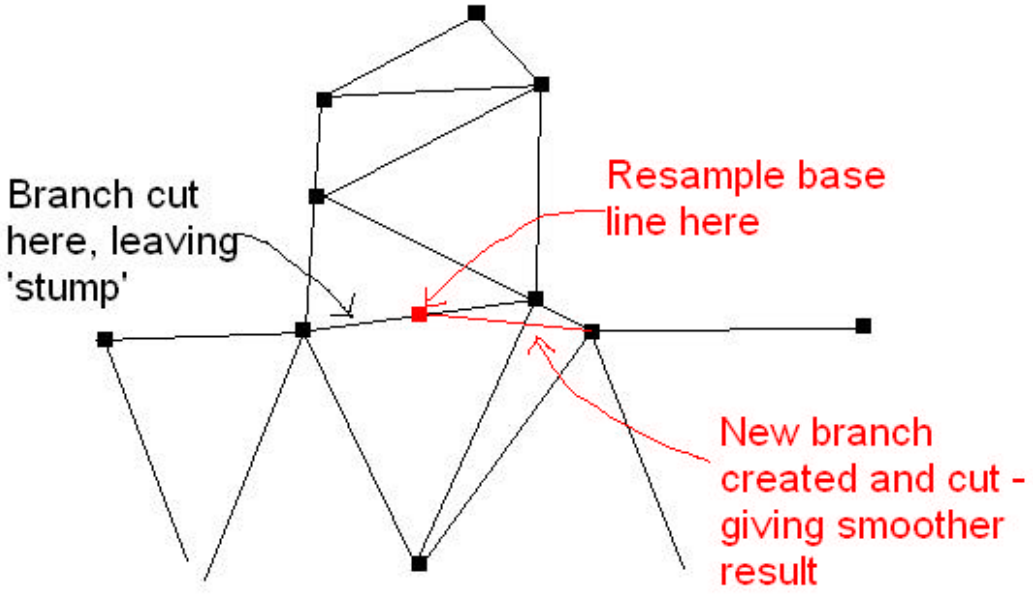

Possible Solutions -

Resample the base line when a cut is made–

Leads to a much smoother result…
but reduces control over total displacement.

Include check on ‘base angle’

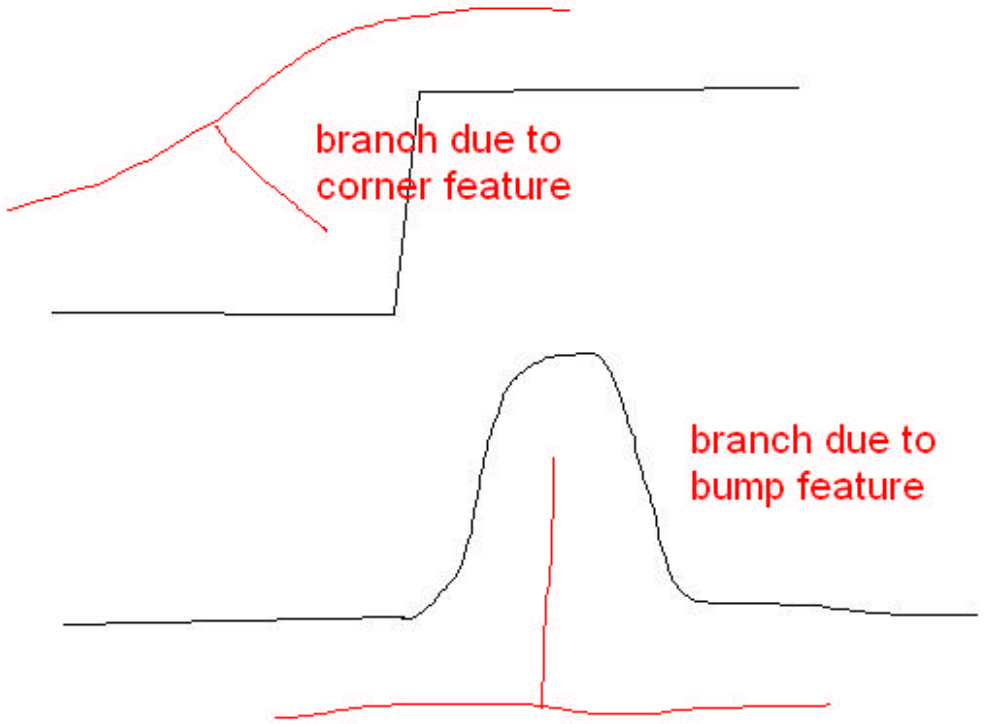
Prevents ugly cuts being made in the first place.




Resampling base line to smooth stumps.

- Has been implemented
- Usefulness not yet assessed
- Has ‘smoothing’ effect
- Greatly increases processing time

2/ Differentiating between Corners and bumps



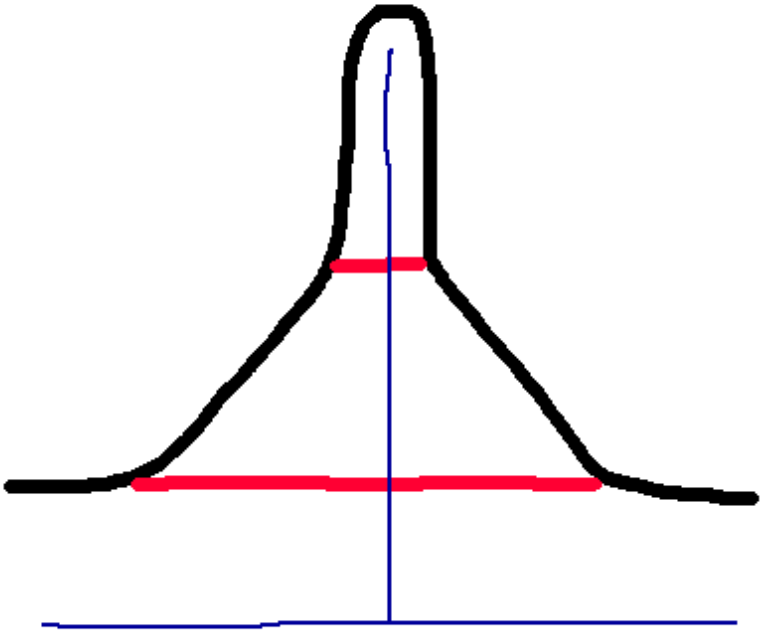
3/ Two sidedness of the line (triangulations on both sides)

- Makes decisions about merging/pruning etc more complex

But could also be advantage – allowing more choice for style of generalisation (e.g. coastlines)

4/ Funnel features.

Features such as this:



Here only one feature is detected, but arguably it could be subdivided into two, the upper ‘spout’ part and the wider ‘funnel’ component.
(Visvalingam’s method does this, though not always)

This could be handled if we allowed the pruning of parts of branches, based on average width.

In order to do this, however, some refinements are required to prevent an undesirable ‘salami’ like repeated slicing of the terminating leaf node, which can lead to the premature removal of a feature.

# 14/ Further Work

- Can apply exaggeration and merging and caricature operators in addition to simple pruning
- Find solutions to corners and stumps
- Allow the part pruning of branches.