NORTH ROAD
CARTOGRAPHY
DEVELOPMENT
SPATIAL ANALYSIS

north-road.com
info@north-road.com
ABN 80 769 844 078
North Road Consulting Pty Ltd

Dynamic Displacement of Lines in QGIS

Version

1.0.1

Date

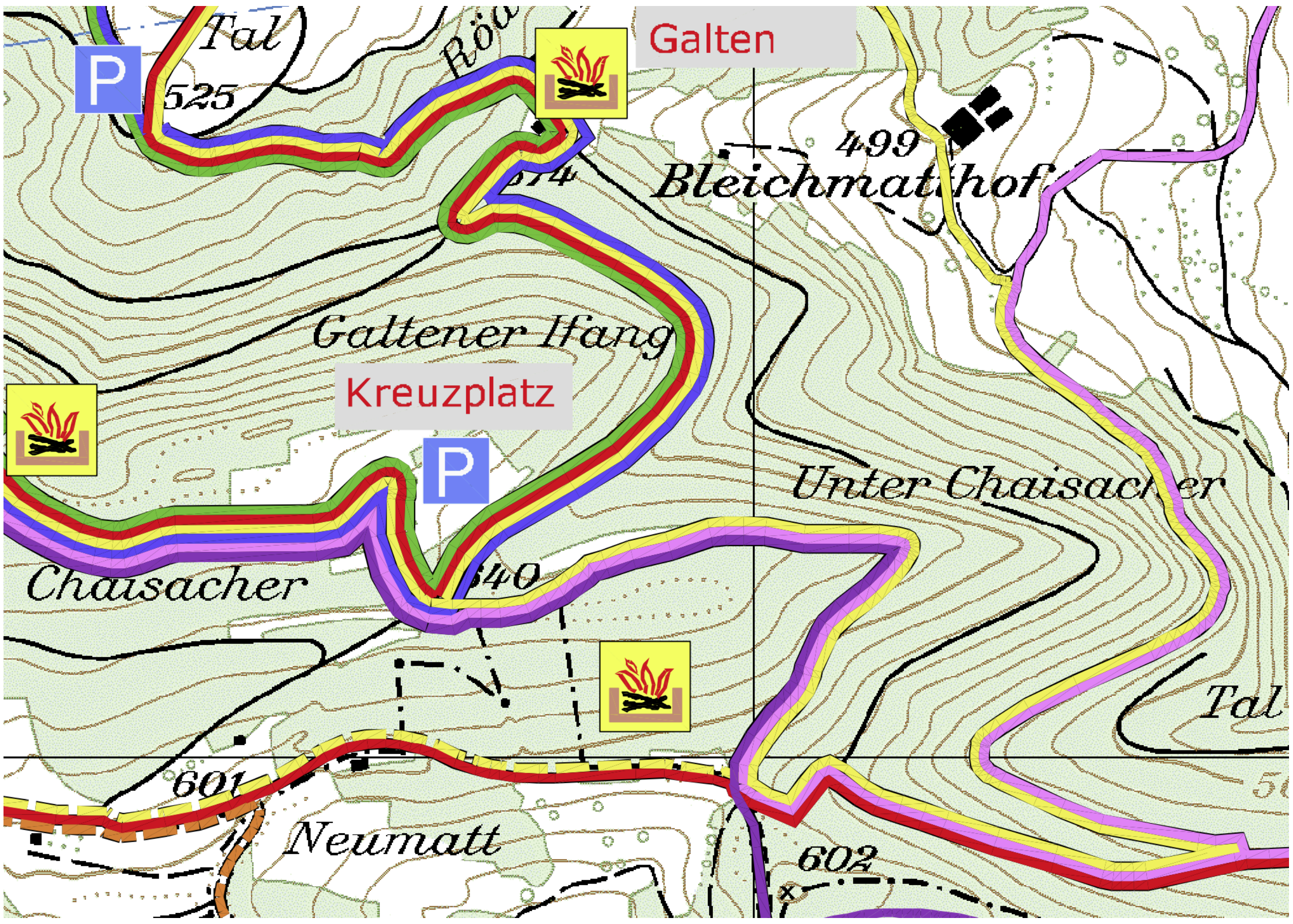
2021-12-23

Document Author

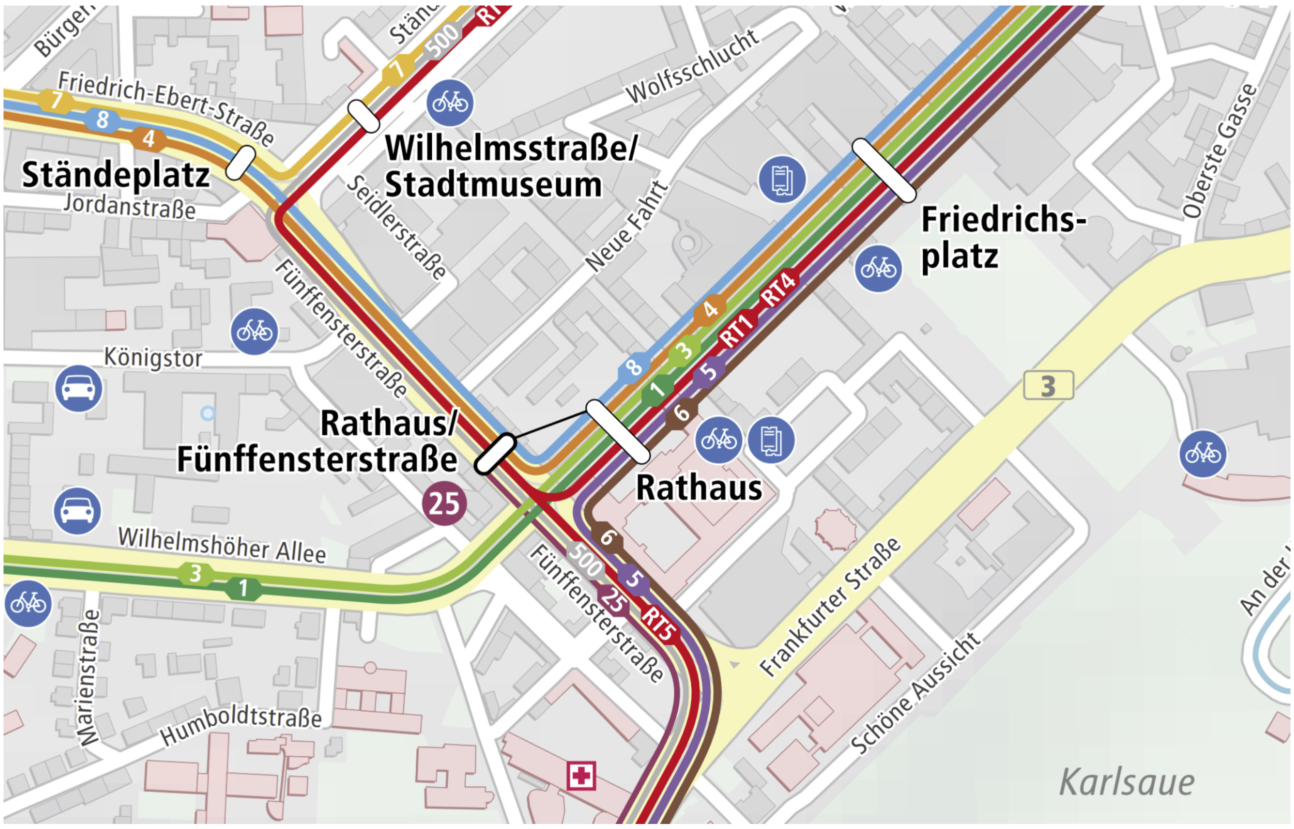
Nyall Dawson (nyall@north-road.com)

Background

A common cartographic technique which aids with the visualisation of overlapping routes is to create displaced lines, as shown in the examples below:



north-road.com
info@north-road.com
ABN 80 769 844 078
North Road Consulting Pty Ltd



This technique is currently a painstaking manual process to create in GIS applications such as QGIS, and any changes to the underlying data structures or geometries means that the user has to re-create the offset lines accordingly.

Thanks to funding from the Swiss QGIS User Group, research has been conducted into possible approaches for developing an automated process for displacing overlapping lines, with the ultimate intention of exposing this functionality as a “Displaced Lines” renderer in the QGIS desktop application.

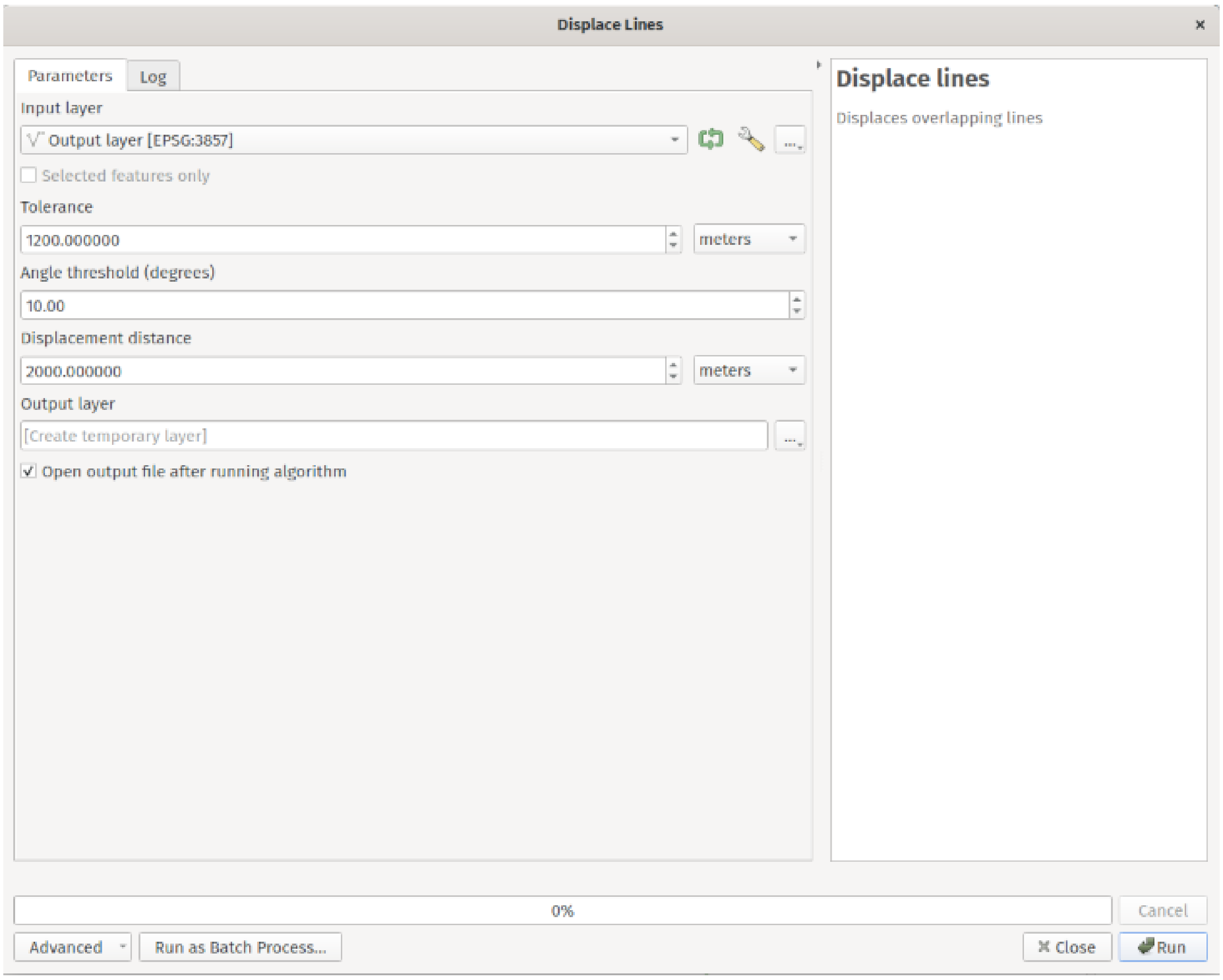
This document describes the research conducted and the logic used in a proof of concept QGIS plugin for displaced line rendering.

Proof of concept plugin

The proof of concept plugin is available from https://github.com/north-road/displaced-lines-renderer .

When installed, it exposes a new “Displace lines” tool inside the Processing toolbox. This tool allows users to test the proof of concept line displacement routine by transforming a vector line layer to an output line layer with displaced line sections.

Options are present for the user to select a distance tolerance, an angle threshold and a displacement distance. (These options are described in detail in the discussion sections below). It is proposed that a dynamic QGIS “Displaced Lines” renderer would expose similar options for user control (also allowing users to set the distance values in paper-based units such as millimeters for scale-independent results.)



Input layer: Output layer [EPSG:3857]
Selected features only
Tolerance: 1200.000000 meters
Angle threshold (degrees): 10.00
Displacement distance: 2000.000000 meters
Output layer: [Create temporary layer]
✓ Open output file after running algorithm

Displace lines
Displaces overlapping lines

Existing Implementations and Literature

Prior to finalizing the approach proposed, we conducted a study into previous literature and software implementations of line displacement tools.

Our findings were:

- The ESRI ArcGIS software exposes some tools for automatically separating overlapping linework via the “Schematics” extension. While we were unable to test this extension directly, the documentation (see eg https://desktop.arcgis.com/en/arcmap/latest/extensions/schematics/separate-overlapping-links-schematic-layout-algorithm-properties-page.htm or https://pro.arcgis.com/en/pro-app/latest/help/data/network-diagrams/partial-overlapping-edges-layout-reference.htm) describe a tool which processes linework in advance (As opposed to a dynamic, “on the fly” tool).
- Previous literature has focused on individual components of the goal, e.g. Cartographic Displacement in Generalization: Introducing Elastic Beams (Bader and Barrault 2001) describes a process for cartographically pleasing displacement of two line features which are known to be overlapping. Given the goal of this study (to determine whether a dynamic displaced lines renderer is feasible), we decided to focus instead on the simplest possible implementation of the renderer, leaving more advanced offsetting techniques as a possible future enhancement. This decision is discussed in more detail in the Proposed Implementation section later in this document.
- Some code snippets and functions have previously been made available for similar use cases via PostGIS and via QGIS geometry generators, however these are of limited use and do not satisfy the goals of this study alone.

Proposed Approach

The task of rendering displaced lines can be split into two tasks:

1. Scanning a set of linestring features to determine which portions of the linestrings are considered should be considered as overlapping
2. Applying a offset or simplification to the overlapping segments in order to visualise them as desired

These two tasks are effectively independent of each other, in that the technique used to identify the overlapping segments does not inherently place any constraints on the technique used to visualise these sections.

Accordingly, the two tasks will be considered separately.

Task 1: Identification of overlapping linestring segments

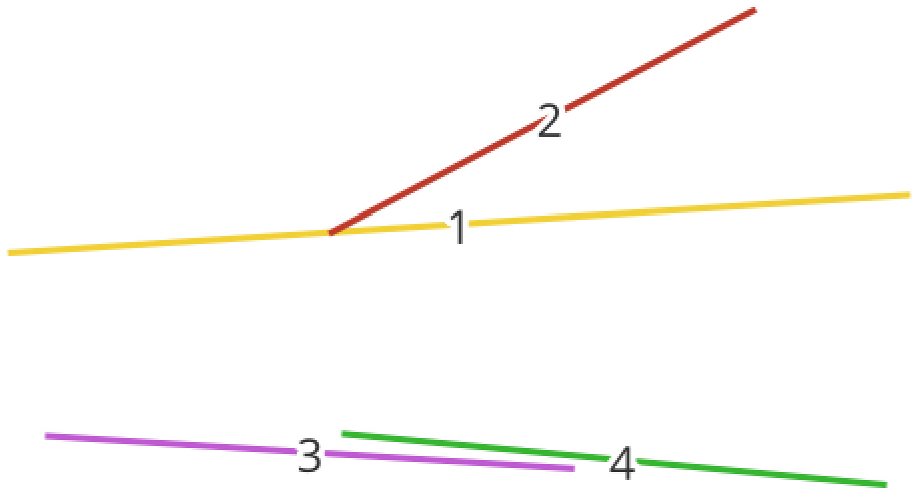
This task consists of scanning the input linestrings in order to extract the portions of those linestrings which are considered overlapping (or near-overlapping). Depending on the data structure, this is either a trivial task (e.g. if the input data is already a topological model describing the relationship between features) or an involved task (e.g. when the dataset is a non-topological spatial dataset, consisting of nearby but non-coincident vertices, partially overlapping features, and complex linestrings with many vertices).

Since the ultimate goal is to create a dynamic, “on-the-fly” line displacement routine, the solution ideally needs to be able to handle complex, non-topological datasets without any pre-processing by users. Accordingly, while more efficient methods are available for determining overlapping segments for a topologically correct model, we instead focus on less efficient but more flexible techniques capable of operating on a set of any linestrings.

The proposed approach is as follows:

1. Break apart the linestring geometries into their component straight line segments, and store these segments in a spatial index for efficient retrieval and analysis
2. Iterate through each of the individual line segments, and
   a. Find all other nearby segments which are within a user-specified distance tolerance, excluding the adjacent segments which belong to the same original feature as the segment being processed
   b. Filter these nearby segments by applying a user-specified angle threshold to the segments.

Specifically, we want to find only segments which are near-parallel and do not want to consider segments meeting at a "T" type intersection as overlapping segments. See illustration below:



In this case features "1" and "2" should not be considered as overlapping segments due to the large angular difference between them, while features "3" and "4" should be considered as having an overlapping section.

c. Split the segment into "overlapping" and "non-overlapping" sections, as there is no guarantee that the entire section will overlap other sections (as per features "3" and "4" in the illustration above).
d. Store each overlapping portions of the segment in groups, alongside all other linestring sections they overlap.

The result of this process is that all linestring features will be divided into parts which are considered overlapping vs those which are not, with all overlapping sections of the linestrings grouped accordingly.

A proof of concept PyQGIS script was developed to apply this logic and test the results. Our findings were:

1. Performance is acceptable given reasonable input datasets. Running the process over simplified rail network datasets resulted in near-instant execution. Running over a more complex statewide rail network consisting of 20k features and a total of 900k vertices took 20 seconds on a mid-range desktop system. This is considered an extreme case, as given the original intention of the displaced lines renderer it is not generally expected to be utilised on such a large, complex dataset at whole-of-state scale. Significant optimisations to the proof of concept code are possible, and porting the logic to native c++ code would also result in orders of magnitude speed

ups.

2. Appropriate selection of the distance tolerance is crucial to obtaining good results. This value needs to be selected based on the relationship between the actual features in the target layer. Setting the value too high results in too many features being identified as overlapping, with generally unpredictable and undesirable results.
3. Similarly, we ran into issues with the proposed logic incorrectly identifying overlaps when the distance between vertices in the linestrings was typically less than the distance tolerance value. Further refinement of the logic would likely alleviate these issues.

We are confident that with further refinement and tweaks the proposed logic is a feasible solution for the overlap identification task.

Task 2: Visualization of overlapping sections

The second task consists of applying a visualization technique to all identified overlapping segments. Specifically, either the linestring geometry from the input features needs to be modified or the appearance of these sections tweaked to visually separate the overlapping sections.

There are numerous possible techniques for this task, including:

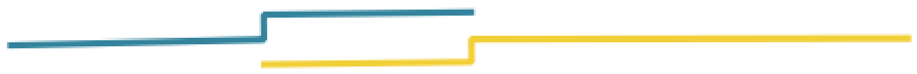
1. Changing the symbol used to render overlapping sections of lines to a partially transparent style, or to a line style which allows display of underlying objects (such as a dashed or dotted line)
2. Modify the geometry of these sections, e.g. applying a line offset to the overlapping sections of line to separate them
3. Showing the line section in a fatter line width to indicate the overlapping sections (e.g. an equivalent of the increasing sized circles often used to visualise clustered points)

Ideally, a flexible QGIS Line Displacement renderer would expose options allowing the user to select between all these different approaches. In fact, the greatest complication associated with the “modify the symbol” approaches such as (1) and (3) are designing a user interface which exposes these options in a usable manner! The actual changes to the map rendering is minimal – simply the sections of lines which were identified as overlapping during Task 1 would be rendered using a different line symbol. Potentially, this overlapping symbol could also dynamically alter appearance based on the number of overlapping features or the individual attributes of the overlapping features (e.g. through use of QGIS data defined expression symbol overrides).

Since the proof of concept code was written as a Processing algorithm which takes a line layer and outputs a new line layer containing displaced lines, it uses the geometry modification approach (2) instead.

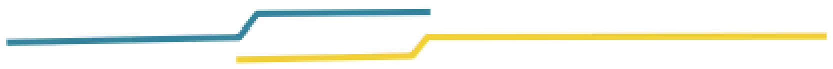
Again, many options are possible for modifying the geometry of overlapping portions of lines. Some advanced techniques are discussed in prior work such as Cartographic Displacement in Generalization: Introducing Elastic Beams (Bader and Barrault 2001) or Automatic Metro Map Layout Using Multicriteria Optimization (Stott, Rodgers, Martınez-Ovando, and Walker 2007).

The proof of concept tool uses a very simple parallel line offset for the overlapping sections of lines, as illustrated below:

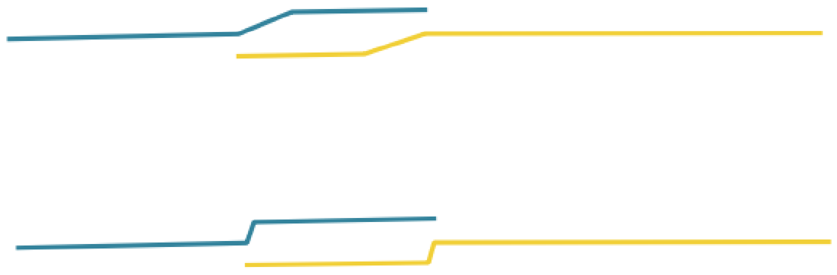


Specifically, each line in each group of overlapping segments is offset from the average line through the group by a user-specified amount, with a straight line segment also added to join the non-displaced parts of the line to the displaced parts. While this technique allows us to visualise all the overlapping line segments, further refinement is desirable in order to produce cartographically pleasing results.

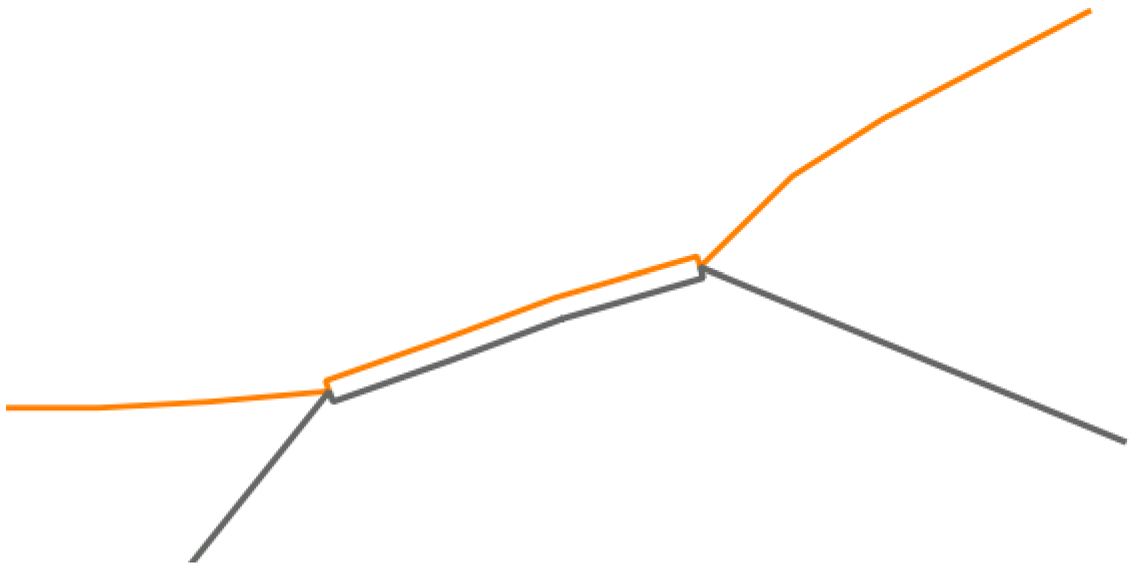
For instance, it would be preferable to angle the lines connecting the overlapping vs non-overlapping sections, as mocked up below:



Likely these options for angling the incoming and outgoing offset sections would need to be exposed for user control, allowing the user to specify the distance over which the angling should be rendered. The appearance of the overlapping sections can vary dramatically depending on this distance:

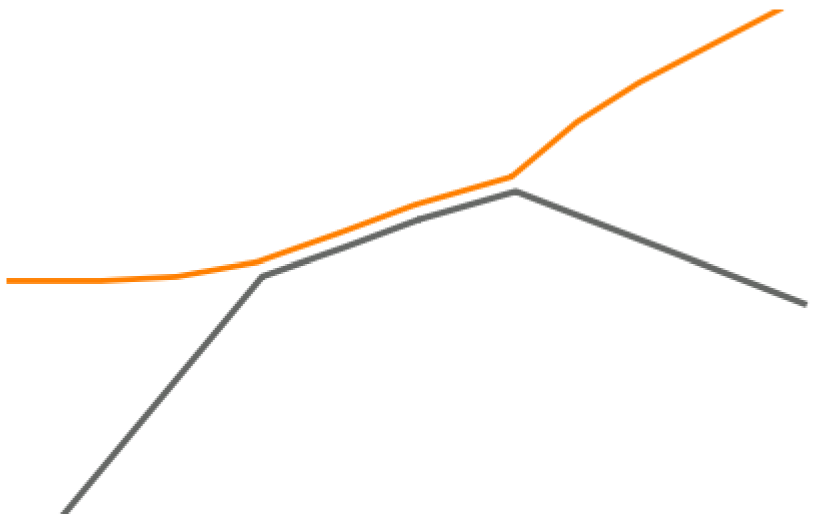


Similarly, some refinement would be desirable to drop the vertices at the start and end of the overlapping sections illustrated below:

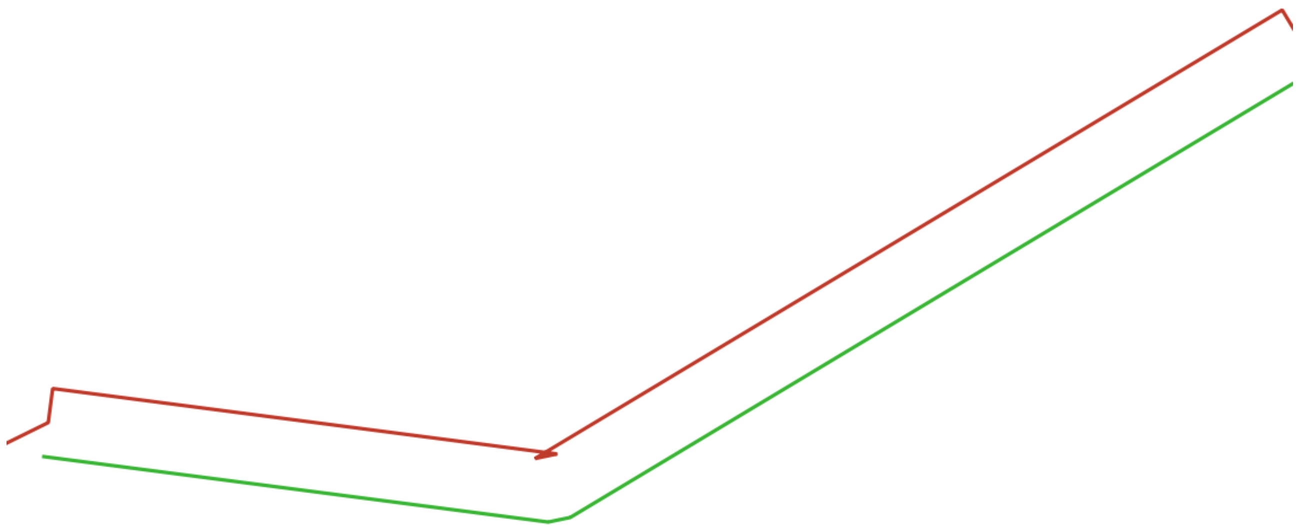


A preferable result would be something like this, where those unwanted vertices are skipped:

north-road.com
info@north-road.com
ABN 80 769 844 078
North Road Consulting Pty Ltd



Another issue which was identified when testing the simple parallel offsets approach is illustrated below:

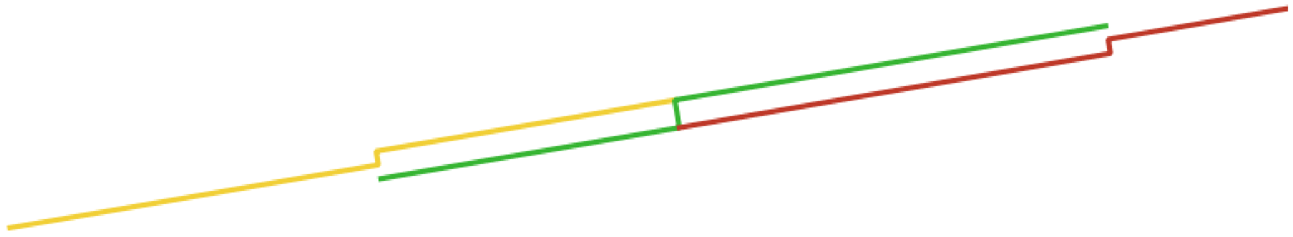


Because each line segment is individually offset, the result can include self intersections when the overlapping portion consists of a bent shape. Ideally, a simple parallel offset would not be used, and instead the “GEOS” line offset routine would be used. This would eliminate the self intersections, and also allow “join styles” such as curved, mitered or bevel joins to be applied to bent sections. (The choice of these should also be exposed for user control).

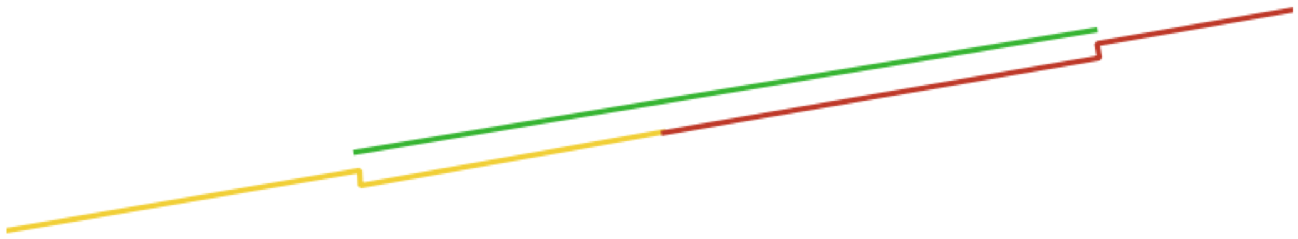
Lastly, an ideal implementation would also include logic to test different ordering of the overlapping

north-road.com
info@north-road.com
ABN 80 769 844 078
North Road Consulting Pty Ltd

feature offsets in order to minimise the number of visual bends are present in the offset lines. The illustration below represents the result of displacing 3 line features, where the green line originally overlapped the start and end of the red and yellow lines respectively:



In this particular situation the ideal routine would have avoided the bend in the middle of the green line, with the following mocked up result:

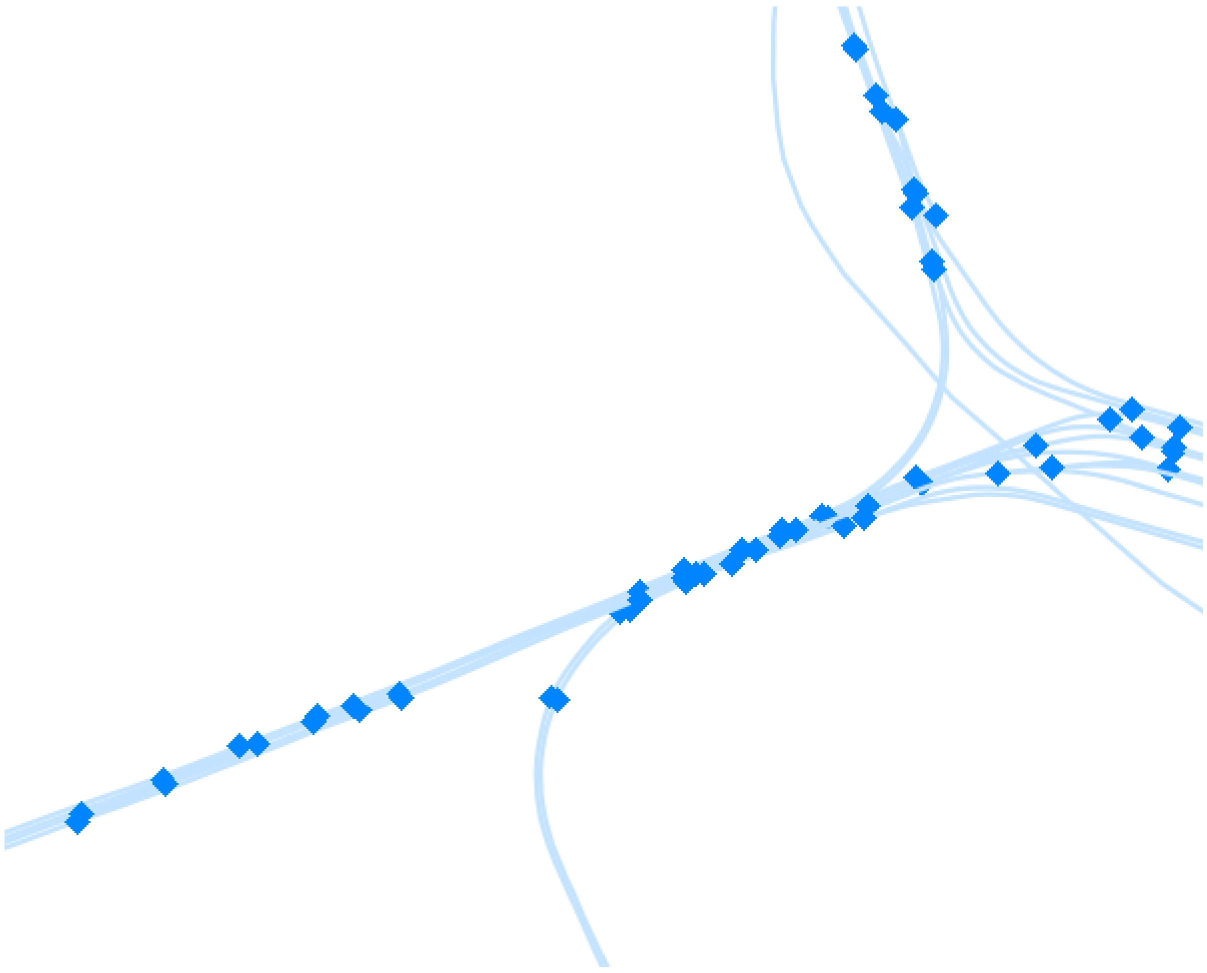


(This refinement would considerably add to the complexity of the routine)

Other identified issues

While the original goal was for the line displacement to work nicely regardless of the input dataset, we ran into issues when testing the process on a line layer which consisted of many very short, consecutive line features. This dataset is illustrated below, where the diamond markers are placed over the start and end vertices of each individual feature:

north-road.com
info@north-road.com
ABN 80 769 844 078
North Road Consulting Pty Ltd



In this case it is impossible to tell from the linestrings alone the best way to separate these. Generally, linework of this nature would be displaced so that features belonging the same routes are offset by the same amount and form a continuous visual line. So in order to handle a dataset of this nature there are two potential options:

1. Require the user to perform some data-preparation manually, e.g. aggregating features with the same route number into a single continuous linestring feature
2. Apply this aggregation dynamically, “on the fly”, by exposing some “group by” setting to users which allows them to pick an attribute (or build an expression) to identify which features should belong to the same offset lines.

Proposed implementation

Based on the research undertaken, our finding is that a dynamic line displacement renderer is a feasible enhancement for QGIS.

We propose that the functionality could be exposed in QGIS in two ways:

1. As a layer transformation tool available through the Processing toolbox (such as the proof of concept plugin does). This tool would allow users to perform the line displacement as a data preparation step, allowing them to then manually refine the result through the use of standard tools like the QGIS Vertex Editor. Furthermore, exposing the functionality as a Processing tool allows users to utilise the operation as part of a larger Graphical Model, e.g. one which includes steps for pre-simplification, aggregation and generalization of the line work prior to displacing lines.
2. As a new layer renderer option for line geometry layers

The new renderer option would be a functional equivalent of the existing “Point displacement” renderer for point layers, which allows users to dynamically offset overlapping points in a map view only (i.e. without any data modification).

In a minimal implementation, the renderer should expose options for:

- Distance tolerance for overlapping lines, allowing distances in paper based units such as millimeters, map units, or pixels.
- A displacement distance for moving overlapping line segments, again available as either paper based units, map units or pixels.
- A “sub renderer” option which dictates how the individual lines are rendered after the displacement is applied, with options for Categorized, Graduated and Rule Based renderers (following the pattern exposed by other renderers such as the point displacement renderer)

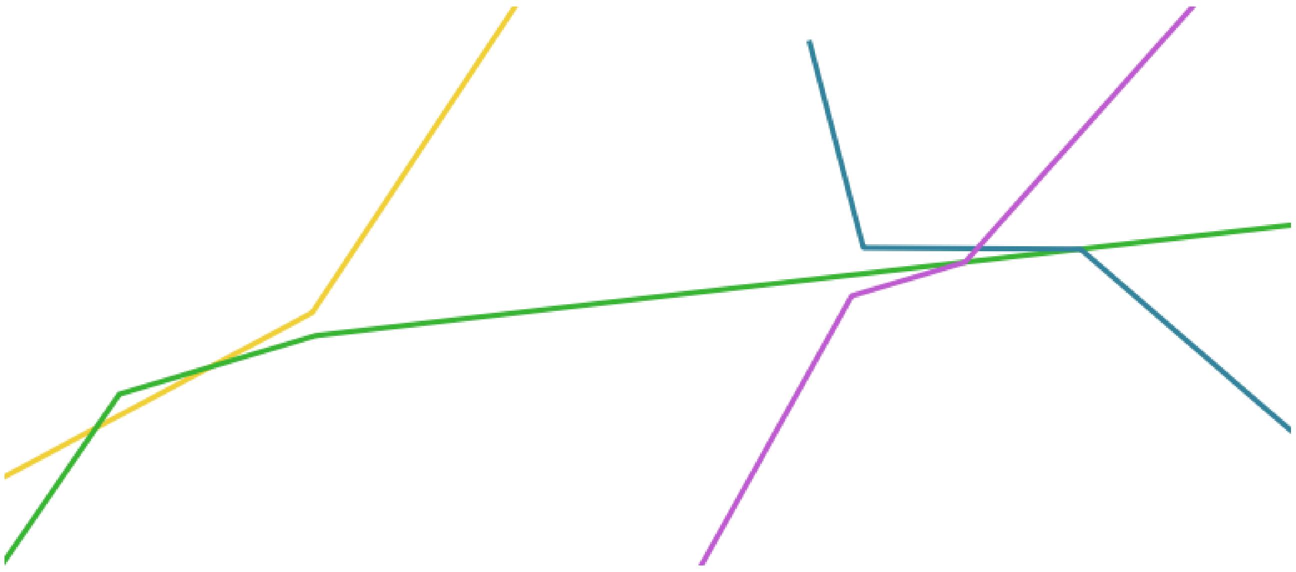
Desirable rendering options would include:

- Control over the angle or distance of lines connecting overlapping and non-overlapping sections of a line feature
- Options for controlling the line offset technique, e.g. by exposing the join styles of round, beveled or mitered joins
- An option switch between offsetting overlapping lines or rendering overlapping lines using a different line symbol

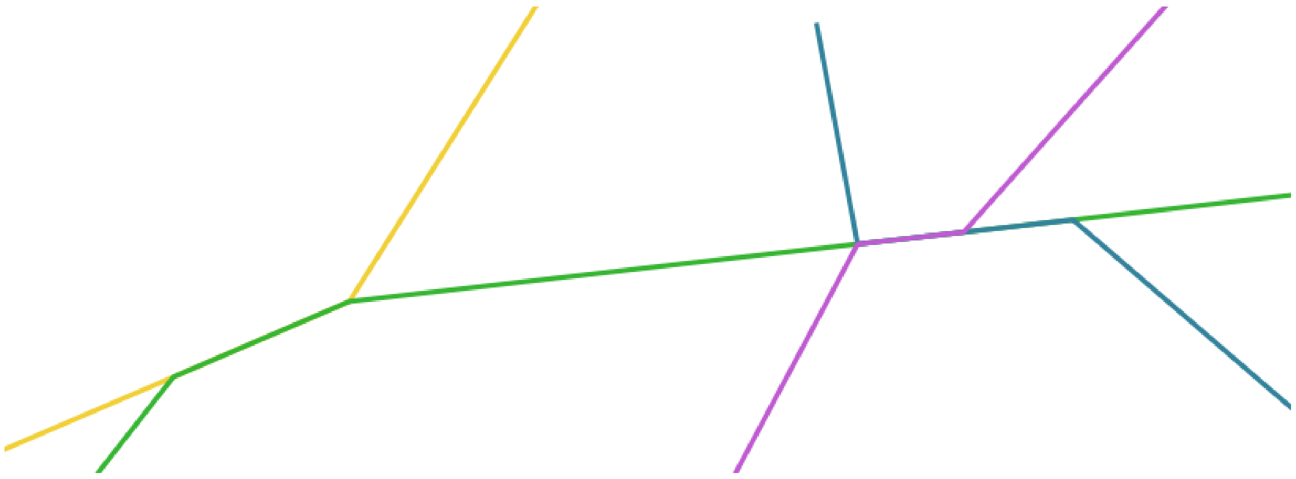
Other refinements possible outside of the minimal implementation could include:

- Inclusion of logic for refining the ordering of offset lines to minimise the number of bends in contiguous offset sections

- Implementation of more complex geometry modification techniques for display of overlapping sections, such as the elastic approach proposed by Bader and Barrault (2001)
- Topologically aware on-the-fly simplification of line sections – after identifying the overlapping sections of lines, it would be possible to simplify and generalize both the non-overlapping and overlapping sections of lines individually, which would avoid any unwanted differences in the overlapping sections which results from simplifying or generalizing all line features independently. See illustration below:



when an optimal on-the-fly generalization would correctly leave overlapping sections as coincident:



north-road.com
info@north-road.com
ABN 80 769 844 078
North Road Consulting Pty Ltd

Next Steps

1. Distribute this document and gather feedback regarding the proposed approach. (Please send all feedback to info@north-road.com)
2. Gather feedback from user testing of the proof of concept tool, and identify issues and potential improvements (via the issue tracker on https://github.com/north-road/displaced-lines-renderer)
3. When sufficient feedback has been received, a quotation for implementing the solution within QGIS will be put together by North Road.

Conclusion

A dynamic, on-the-fly line displacement renderer is a feasible enhancement for QGIS, offering many styling options which are not currently possible without manual, painstaking data preparation. This functionality should be considered for future investment and implementation in QGIS.