Estimating and editing transit topology over the road graph using supply data feeds

Mark Dimond
Integrated Transport Planning Ltd.
Nottingham, UK
dimond@itpworld.net

Neil Taylor
Integrated Transport Planning Ltd.
Nottingham, UK
taylor@itpworld.net

Robert Houghton
University of Nottingham
Nottingham, UK
robert.houghton@nottingham.ac.uk

Abstract

Models of citywide public transit infrastructure work best when they have not just geographic data, but the underlying topology: how individual transit routes relate to each other and the road network. Having such information allows strategic analysis of transit supply and demand at varying scales. This is important in understanding which road segments are heavily utilised by multiple routes, or what each segment along a particular route looks like under different traffic conditions. However, capturing a topological model can be time consuming. Furthermore, it may not contain the full road network (just transit routes), or it may require frequent updates as information about new routes is added. We describe the use of simple map matching to link transit supply data (in the industry-standard General Transit Feed Specification format) to a segmented street network model based upon OpenStreetMap. This linking uses a routing engine, iterating over stops in a GTFS file to estimate the roads used by particular stoppings of buses. A novel web interface then allows correction of the selected road segments by a knowledgeable user. The final result is a 'road-aligned' transit supply database which the user can query for either route or road network intelligence.

Keywords: map matching, topology, public transport, GTFS

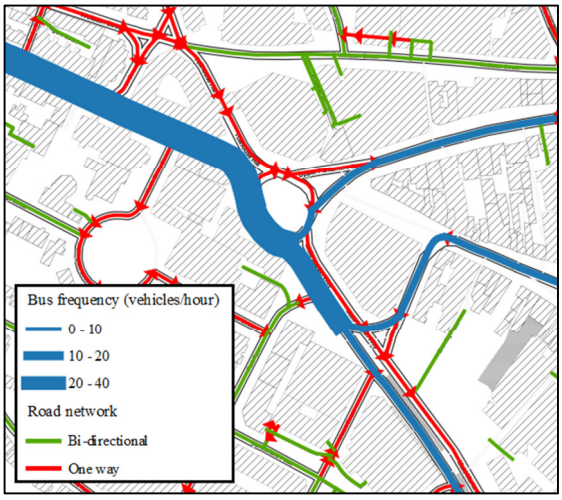
1 Introduction

The modelling of public transit availability with the General Transit Feed Specification (GTFS) [1] has facilitated a wide range of different software tools in recent years. Cities and authorities make 'feeds' of transit information available freely online, promoting their own public services but also supporting the development by third parties of different apps. These include apps for journey planning [2] or travel time analysis [3]. However, though it is based on a simple, flat-file structure, the collection and validation of GTFS feeds has some limitations. Firstly, the optional route 'shapes' showing the geography of the data are costly to produce and update. Secondly, and more importantly, feeds are not well suited to the analysis of transit as it relates to the wider road graph. A feed represents a separate graph of possible trips and stoppings which do not associate bus or tram services with the exact edges in the road graph they may use.

In this work we propose a solution to estimate the *topology* of the GTFS feed as composed of edges from the underlying road graph. Building GTFS using roads in this way has two benefits: it permits fine-grained analysis of transit provision at a graph-edge (street link) level of granularity, and it can exploit an existing, externally managed map dataset from which to draw road network topology. Analysis possible using topology includes frequency calculations for individual street links as shown in Figure 1, or viewing of congested sections of a specific planned route. These queries are not feasible using the stopping or shape data in GTFS alone, as the exact edges used by each route are not available.

For our work, we retrieve the road graph from OpenStreetMap and test a novel approach to matching transit trips against this dataset using a street routing engine. This process builds upon existing work in the field of map-matching [4,6,7]. A simple tool uses routing to batch-process the stoppings in a whole GTFS feed. It then presents route *alignments* for a user to validate through a UI – after which the detailed topology of the transit supply can be used for more sophisticated analysis of routes and their constituent edges.

Figure 1: Road network topology and bus routes



2 Background

Attempting to estimate from GTFS the road network used by a particular transit service is not trivial. A GTFS 'feed' consists of trips described by multiple stops, each of which includes a set of coordinates. The problem is therefore to detect the street graph edges used between successive pairs of stops, but intuitively one can see that almost any number of edges might be correct, from one (stops are on the same edge) to hundreds (a long-distance bus route).

This can be considered a form of *map matching*, a very well studied problem in the field of geographic information. Map matching attempts to identify the sections of a street graph that represent the true state of noisy position observations [4]. In existing literature, these positions often come from GPS or other satellite positioning, but we argue that the point features of GTFS stops form a very similar basis for map matching.

A variety of different algorithms have been demonstrated and refined over a period of decades. Much of the earlier work has been extensively documented by Quddus *et al* (e.g. [4]). In a prominent project, [5] report a hidden Markov Model to be particularly effective at matching noisy GPS points to an existing map, using probabilistic modelling of the transitions between graph edges to optimise the selection of each successive edge. Their method, which uses both street routing distance and straight-line distance as heuristics to aid this selection, applies the Viterbi algorithm to compute the most likely sequences of graph edges for a particular input. The authors report good performance even at sample rates of 30 seconds or less [5, p7]. More recently, [6] have produced both a substantively updated review and an algorithm that hybridises the local- and global-search techniques of other attempts. These sophisticated techniques are typically based around calculating the minimal Fréchet distance [10] between the sequence of observations and the road network (in some cases using a *free space graph* between the two sets to select a global minimum [6, p435]).

The remainder of this paper describes a method for map matching based upon street routing. To our knowledge, a comparable approach has been tested only once in existing literature: [7] report reasonable results using routing after pruning GPS logs of higher sample rates results to only the most significant points. In [11], a GTFS feed is matched against a road network using a breadth-first edge evaluation algorithm. The contribution of our work above this is twofold:

1. we test an off-the-shelf routing algorithm, which requires none of the parameter-setting required of the solution in [11] and is usable with different transit modes; and
2. we present an interface for a user to view and correct the topology immediately after it is estimated.

We note the intuition that the stops of a trip in a GTFS feed are similar to having very low sample-rate positioning logs [11, p8], and hence argue that a routing algorithm is most appropriate in order to maximise correct edge selection between stop points. Whilst some considerable work has been done to improve map matching from very sparse data [8,9], we suggest that the 71% accuracy achieved for a 1.5 minute sampling rate [9, p9] can be improved upon. There is the further advantage that a routing algorithm can very easily be used over different graphs for various modes of on- and off-road transit, such as trolleybuses, trams, and rail.

3 Routing-Based Map Matching

This section describes our workflow for deriving an estimated topology from a General Transit Feed Specification (GTFS) file. A brief description of the feed format is given, before summarising our workflow and implementation considerations.

3.1 GTFS

Developed by Google and the city of Portland, USA [1], the General Transit Feed Specification has become a de-facto standard for description of public transport (transit) data. A 'feed' file captured by an operator or authority contains details of transit supply at three levels:

- *route*, describing meta-information such as name, transit mode, description, and operator of a particular service;
- *trip*, which captures different directions and variations of a service over a given route;
- and *stop time*, which captures the timings and stoppings of a trip at known stop locations. [1]

The locations and meta-information of each *stop* are captured separately for use in different trips. A GTFS file may optionally include data regarding the *shapes* (geometry) of trips between stop times, captured as a coordinate linestring, to aid visualisation. This is not considered in our present workflow, as these shapes may be of varying levels of data quality (or completely missing). Using them to improve map matching is a promising avenue for future improvements to the system described below. However, note that using shape data would likely require a different form of map-matching to that examined below, as they are likely to contain significantly more vertices from which to match.

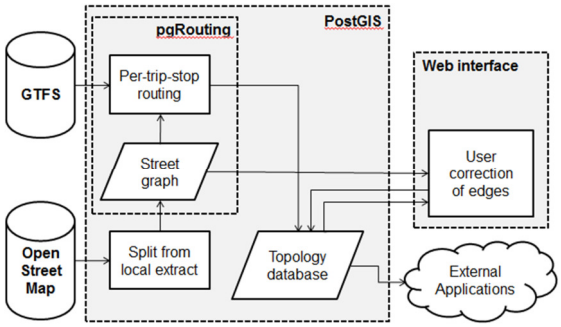
3.2 Overall Architecture

The road network graph used in our workflow is derived from OpenStreetMap (OSM), which provides reasonable or good topological road data in the cities for which we have tested the solution, and has the added benefit that it can be updated by users. Existing studies have shown that in Europe, OSM contains high quality road network data [14]. Nevertheless it is also possible to use our approach with other road databases.

The graph is cut from a citywide OSM extract using the OSMSplit tool, part of the OpenTripPlanner project [2]. The resulting geodatabase can then be used by a route planning algorithm to estimate routes between stops. This process typically takes at most several minutes for a whole GTFS feed, though it is run as a batch process in the first iteration. For the geodatabase we use PostGIS because it supports the *pgRouting* extension (see http://pgrouting.org/). Using this

extension we are able to calculate the shortest route using Dijkstra's algorithm [12], which is proposed as the route alignment between stops. Dijkstra is chosen here for simplicity, though other routing algorithms such as A* are supported by pgRouting. Figure 2 shows the architecture of our system for trip alignment.

Figure 2: Architecture of map matching



3.3 Review Interface

Once completed, the route alignment results can be inspected through a web application by a knowledgeable user. Adjustments to the detected road alignments can be made where necessary using an intuitive click-select interface which allows the user to capture all of the road segments for a particular trip's 'stopping' (stop time). Selecting and editing the route's segments is achieved through asynchronous calls to a database API from the web interface. (This method is used as preliminary development showed that loading all possible street segments for a city in a web browser was infeasible using current browser technology.) Figure 3 shows the interface with an incorrect road alignment prior to correction; the user simply clicks a stop location, and then clicks to toggle the correct street edges in the appropriate sequence. If available, data from the *shapes.txt* feed file is displayed as a guide for alignment – shown here as the purple, thinner line.

Having checked the detected route alignments against the road network, the complete feed can then be re-exported to valid GTFS. However, included in the completed feed are two new data types:

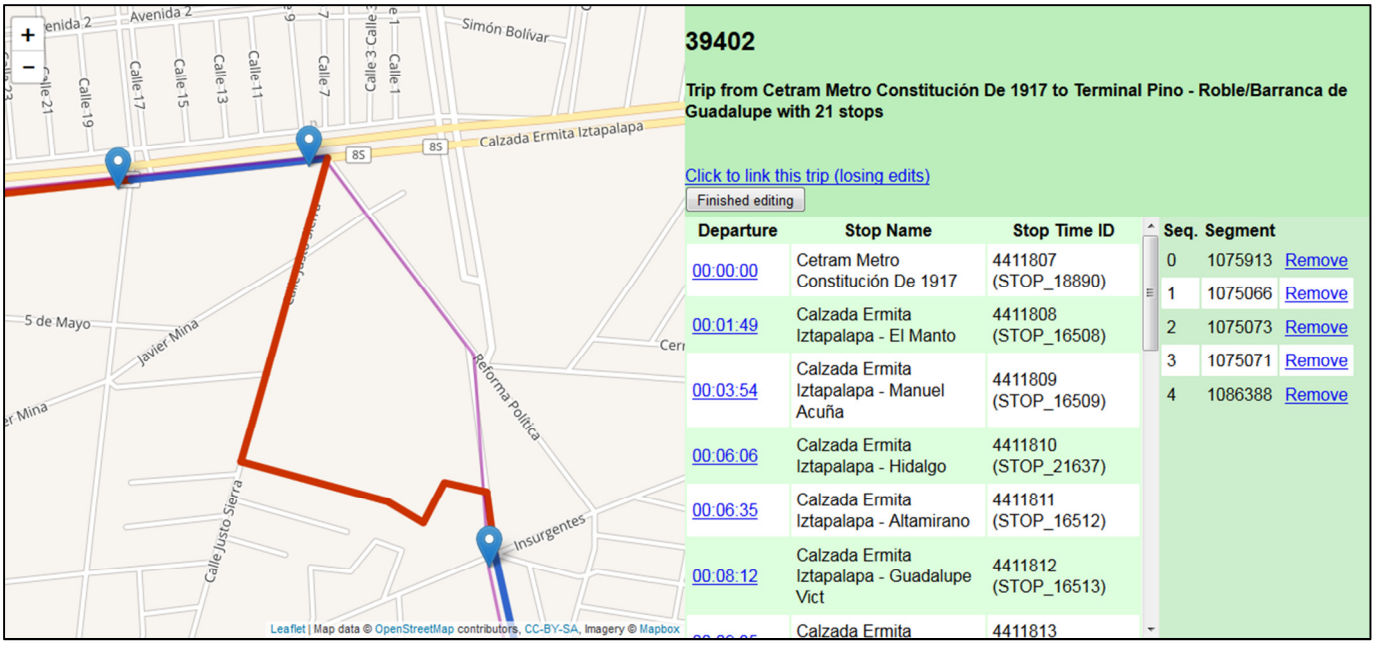
- *alignments*, which capture road segments with unique identifiers against each stop time in the feed; and
- *segments*, which capture the geography of the edges in the road graph and the original OSM 'way' identifiers from which they are derived.

The former file encodes the topology of the road network, allowing analysis of the road segments which are used by different services. Using the PostGIS geodatabase also permits export to different formats that support topology, such as the compact and flexible TopoJSON format [13].

3.4 Challenges

One of the main advantages of using a routing framework to estimate topology is the straightforward capture of traffic rules, directionality, turn restrictions etc. This ensures that the proposed route alignments selected for a GTFS trip form a connected route and hence are more likely to be correct. Nevertheless, in certain instances the wrong lane of a multi-lane carriageway may be selected by the routing algorithm, and must be corrected by the user. Validity of the suggested route *for the trip* is a related issue: the transit mode used must be able to navigate all suggested segments, so trams cannot be routed via bus lanes and buses cannot use subways! For our prototype application, only roads and bus lanes are included, meaning only bus-based trips are currently tested.

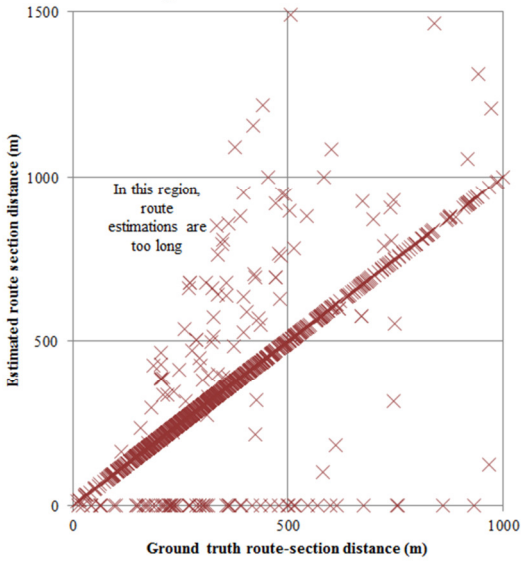
Figure 3: Editing interface with an incorrect alignment shown. (The correct route is shown by the thinner line)



4 Results

In this section, we present the results of attempting road alignment using the freely available Mexico City GTFS feed (available for download at http://transitfeeds.com/), and report statistics regarding its performance. Not all routes are perfectly matched: poor quality stop data, incomplete OSM, and unusual real-world behaviour of the transit system all mean that the road alignments selected by the routing algorithm are not perfect w.r.t. the correct behaviour of the trip in question.

Figure 4: Map matching performance by ground truth distance

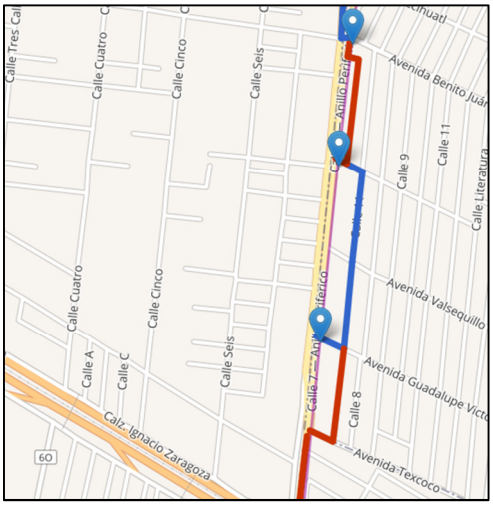


To evaluate the performance of the routing-based map matching, the correct segments for 34 GTFS routes, containing 1,237 stoppings, were selected from the OpenStreetMap graph using a desktop GIS. This represents approximately 2% of the entire feed's trips, chosen at random amongst agencies that use buses. The batch route linking process was then applied and compared to this ground truth. For 82.9% of the stoppings, the exact correct sequence of edges was selected by the algorithm (with no overestimation). Figure 4 gives further indication of the algorithm's performance at selecting edges: points that are not on the 'correctness' line represent estimated alignments that are either too long or short relative to the correct route for a stopping. One observes that for shorter distances between stop times (250 metres and less), the routing algorithm is more likely to select a route of the correct length, whilst for longer stop-to-stop journeys, more mistakes are made. Note that an estimated route being the correct length does not necessarily mean it is correct: for example if a route travels between opposite corners of a city block, it can travel either west followed by south, or south followed by west, and both routes will be almost identical in length. Nevertheless, distance correctness is used as an indicator.

The figure shows a number of estimated routes with zero distance (zero on the vertical axis). We applied a heuristic to discard incorrect routings: if the best available route found between two stops is greater than three times the straight-line distance, the estimation is not recorded, as it is assumed to be incorrect. The user can then select the actual route taken using the interface without first having to remove incorrect alignments.

The quality of stop data within the GTFS feed causes minor problems because, in our experience, some transit authorities record their stop data less precisely than others. The majority of GTFS feeds contain one stop for each direction of a trip, on the appropriate side of the road. However in the case of one feed for which we tested the linking process, one recorded stop was used to denote an approximate stopping point for two directions of travel (typically 'outward' and 'return'). In cases where the road in question has multiple carriageways, this causes great difficulty as the routing algorithm could not account for a 'return' trip along the outbound carriageway (see Figure 5). In future work we will address this by making it simple to add corresponding stops in the application interface.

Figure 5: Stop pairs perform better than single stops.



5 Conclusions

This work has presented two contributions: a method for inferring transit network topology from commonly available data feeds; and a UI with which a knowledgeable user can edit these inferences to ensure the resulting model is fully correct. Our results show good performance using routing between trip stops as a form of map matching. The manual correction process is a minor drawback, but this will be reduced as OSM and GTFS data quality improves. Data quality of stop locations in particular is an issue, and in cases where one stop is used to represent journeys on opposite sides of a multi-carriageway road, the correct routing cannot be determined automatically.

In future this will be mitigated by making it simple to add GTFS stop locations in the UI, at which point the inference

process can be re-run. Further work will allow extraction of separate graphs for different transit modes, allowing inference for trams, subways and rail routes using data already in OSM. Finally, the ability to monitor a given city's OpenStreetMap data for changes will allow the user to ensure the transit topology is updated when necessary.

Acknowledgement

This work was co-funded by Innovate UK, ESRC and EPSRC, through a Knowledge Transfer Partnership (no. 9472).

References

[1] *General Transit Feed Specification Developer Reference*, https://developers.google.com/transit/gtfs/ Google

[2] Hillsman, E., and Barbeau, S. (2011) *Enabling Cost-Effective Multimodal Trip Planners through Open Transit Data*, National Center for Transit Research Report.

[3] Transport Analyst http://conveyal.com/projects/analyst/ Conveyal

[4] Quddus, M., Ochieng, W., and Noland, R. (2007) *Current map-matching algorithms for transport applications: State of the art and future research directions*. Transportation Research Part C.

[5] Newson, P. and Krumm, J. (2009) *Hidden Markov map matching through noise and sparseness* ACM SIGSPATIAL 2009

[6] Wei, H., Wang, Y., Forman, G., and Zhu, Y. (2013) *Map matching: Comparison of approaches using sparse and noisy data* ACM SIGSPATIAL 2013

[7] Griffin, T., Huang, Y., and Seals, S. (2011) *Routing-based map matching for extracting routes from GPS trajectories*. COM.Geo 2011

[8] Lou, Y, et al (2009) *Map Matching for Low-Sampling-Rate GPS Trajectories*, ACM SIGSPATIAL 2009

[9] Yuan, J., et al (2010) *An Interactive-Voting Based Map Matching Algorithm*, ACM Mobile Data Management 2010

[10] Alt, H. and Godau, M. (1995) *Computing the Fréchet distance between two polygonal curves*, International Journal of Computational Geometry and Applications 5.

[11] Perrine, K., Khani, A., and Ruiz-Juri, N. (2015) *A Map-Matching Algorithm for Applications in Multimodal Transportation Network Modeling*, Transportation Research Board 94th Annual Meeting, Jan. 2015, Washington DC

[12] Dijkstra, E.W (1959) *A note on two problems in connexion with graphs*, Numerische Mathematik 1

[13] Bostock, M., (2013) *How to infer topology* http://bost.ocks.org/mike/topology/

[14] Ludwig, I., Voss, A., and Krause-Traudes, M. (2011) *A Comparison of the Street Networks of Navteq and OSM in Germany*, Springer Lecture Notes in Geoinformation and Cartography.