Developing a Dashboard of Last-Mile Freight Traffic

Kostas Cheliotis[1]

Centre for Advanced Spatial Analysis

University College London

Carried out as part of the Freight Traffic Control (FTC) 2050 project

www.ftc2050.com

15 July 2019

[1] k.cheliotis@ucl.ac.uk

Contents

1 Introduction

This report outlines the development of a proof-of-concept tool to be used in the visualisation of last-mile freight data. It focusses on a single local authority in London, UK, visualising all freight traffic reported from a major courier operator that was passing through the area over the period of a month.

The dashboard as presented here aims to be a visualisation tool for non-specialist users. It enables users to perform core analytics functions, relevant to the dataset and system, without requiring specialized software and tools.

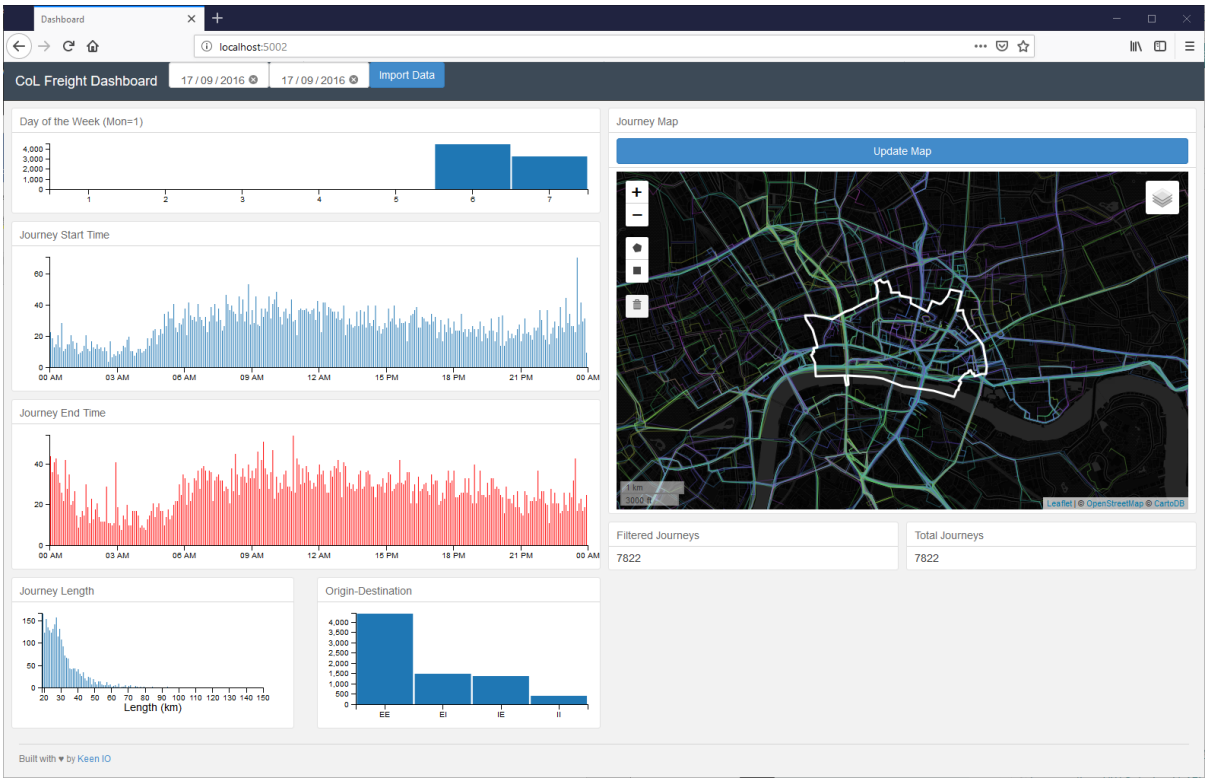

Figure 1: The dashboard window

2 Functionality

The dashboard tool focusses on two distinct types of functionality: Data visualisation, and data filtering. These aspects are designed to work in tandem, so that when users apply a filter on any of the dataset dimensions, all of the visualisation elements update in real-time to reflect the changes. The only exception to the real-time updates is the map, which for purposes of efficiency and usability does not update automatically for larger datasets (1000 trips by default) but requires a user prompt to trigger the update.

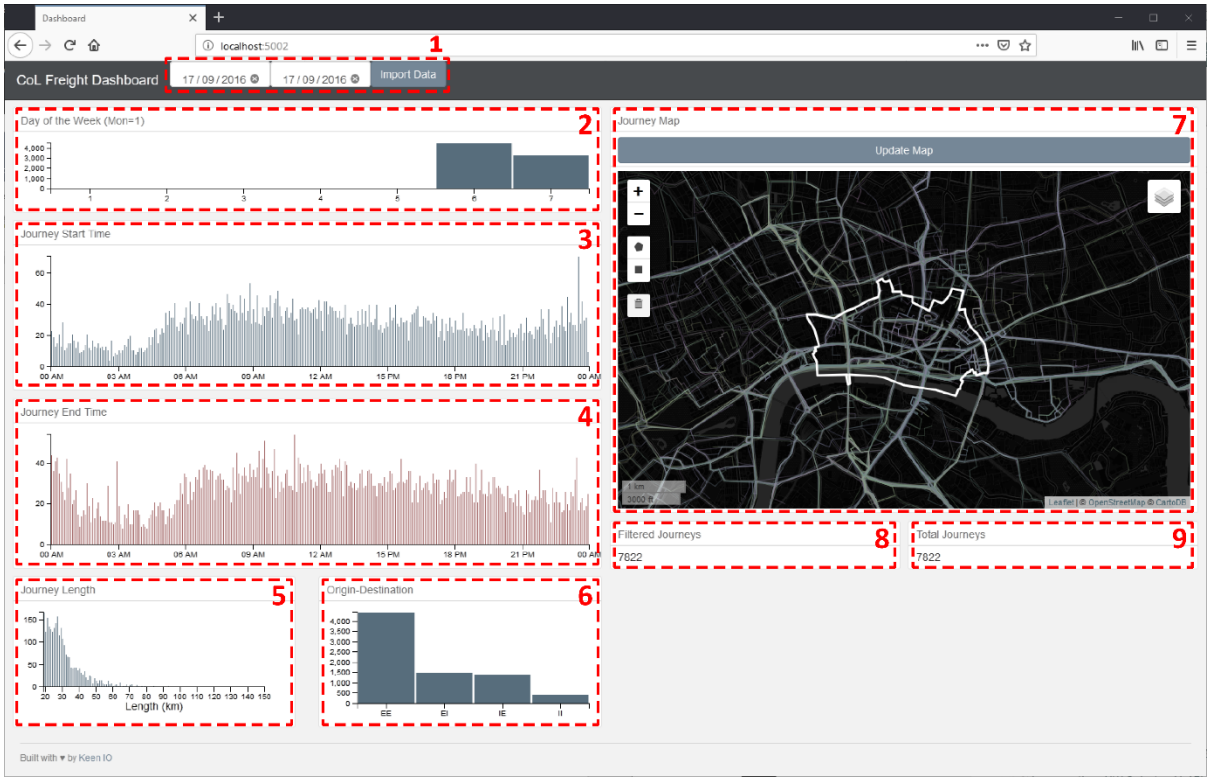

Figure 2: Dashboard window overview highlighting all elements

2.1 Visualisation

The dashboard tool has the capacity to visualise multiple attributes at the same time in a single view. In its current form, the dashboard visualises 7 attributes, as shown in Figure 2: number of trips by day of the week (2), by time of day (by trip start (3) and trip end (4) time), by trip length (5), by origin-destination pair (6), a map of the trip paths (7), and a heatmap showing the density of destinations (in 7, not shown). In addition to these seven elements, the dashboard includes two numeric outputs, one (9) showing the total number of trips contained in the dataset, the other (8) showing the number of trips currently selected via the application of data filters.

2.1.1 Charts

Two types of charts are currently used: categorical charts for nominal and ordinal data types, and histograms for interval and ratio data types.

2.1.1.1 Trips by day of the week

Number of trips by day of the week is visualised in a categorical bar chart. The 7 categories represent seven days of the week, ordered Monday to Sunday (Monday = 1, Sunday = 7). Each trip is counted once, by trip start time[2].

2.1.1.2 Trips by start time

Number of trips by start time is visualised in a histogram. The horizontal axis represents a 24-hour clock, divided into 5-minute intervals, resulting in 288 discrete time blocks. Trips are aggregated by start time, with each time block visualising the number of trips that began within that duration.

[2] In cases where a trip's duration spans two days, its day of the week is considered to be the day on which the trip began.

2.1.1.3 Trips by end time

Number of trips by end time is visualised in a histogram, similar to the start time histogram. The horizontal axis represents a 24-hour clock divided into 5-minute intervals, with each interval visualising the number of trips that ended within the duration. The end time histogram is coloured differently to the start time histogram, to visually differentiate the two.

2.1.1.4 Trips by length

Number of trips by trip length is visualised in a histogram. The horizontal axis represents the trip length in kilometres, rounded to the nearest integer value. The vertical axis represents the number of trips for each trip length.

2.1.1.5 Trips by origin-destination pair

Number of trips by origin-destination (OD) pair is visualised in a categorical bar chart. This chart visualises the location of a trip's endpoints (start and end locations) in relation to the area if interest. The endpoint values may be: External (E), when the location falls outside the area of interest, and Internal (I), when the location is inside the area of interest. Each axis in the OD matrix has two values, resulting in four classes for the OD chart: EE (a trip began and ended outside the area of interest), EI (a trip began outside and ended inside the area of interest), IE (a trip began inside and ended outside the area of interest), and II (a trip began and ended inside the area of interest).

2.1.2 Maps

The dashboard implements a web-map canvas, which allows for the visualisation of geospatial attributes of the dataset. Different geospatial attributes are visualised using multiple layers (one layer per attribute) overlaid in the same map canvas, to allow for visual comparison between layers.

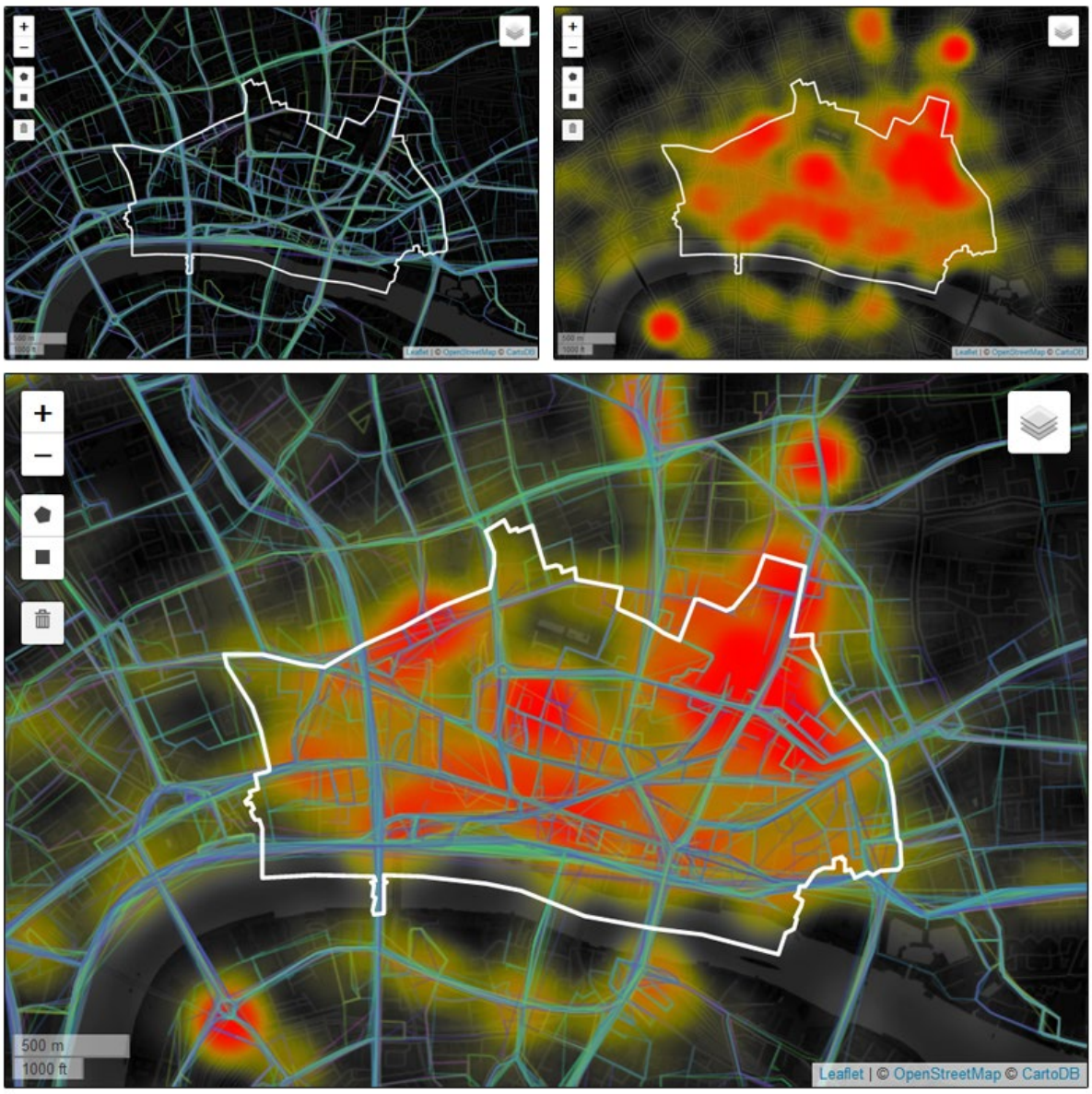

Figure 3:Map Layers. Top left: Trip paths layer. Top right: Trip destinations layer. Bottom: Combined view of both layers.

2.1.2.1 Trip paths layer

Trip paths are visualised using a polyline layer, with each trip represented by at least one polyline[3]. Each line is coloured by device ID (a unique device ID broadly corresponds to a unique vehicle), and therefore multiple trips taken by the same vehicle are represented in the same colour. Due to processing limitations, the number of concurrent trip paths shown on the map is capped. The cap is set to 1000 by default, meaning that only a maximum of 1000 trip paths are shown on the map at any given time.

2.1.2.2 Trip destinations layer

Trip end locations are visualised using a heatmap layer, showing the density of destination points at different locations. The heatmap is produced using a kernel density estimation (KDE). The heatmap layer is affected by the web-map parameters and is dynamically recalculated every time the web-

[3] In cases where input data contains errors or inaccuracies, a trip path may contain gaps, i.e. a trip may be split into multiple polylines

map zoom level changes, in order to provide a visualisation more appropriate to the current view. The kernel radius parameter for the KDE is expressed in pixels, set to 25 by default, meaning that each pixel's value is calculated based on the points within a 25 pixel radius (which corresponds to different length distances, dependent on current zoom level). Heatmap intensity is capped to a fixed value, set to 10 by default, and each point's weight is expressed as the reciprocal of the heatmap intensity value, meaning that a pixel showing full intensity has at least that number of points (10 points by default) within and/or around it within a 25 pixel radius.

2.2 Filtering

Data filtering in the dashboard is applied on the same objects used for visualization, i.e. charts and maps are interactive elements. Filters for each dimension are applied differently depending on data type and visualisation method. Filters can be can be combined over multiple dimensions allowing for fine scale analysis, for example identifying all lengthy journeys during business hours that traversed a specific junction.

2.2.1 Charts

Filters on chartable data dimensions are implemented directly on the chart visualising the data dimension.

2.2.1.1 Categorical Data

For datasets visualised through a categorical chart, the user can select which categories to include in the selection. Multiple categories can be selected simultaneously. The filter is facilitated by clicking directly on the bar representing the category in question. Clicking on a bar toggles its filtering status: Clicking on a category currently active deactivates it, thereby excluding all data that fall within the category from the current selection, and vice versa. For a given dimension, all categories are included in the initial filter by default.

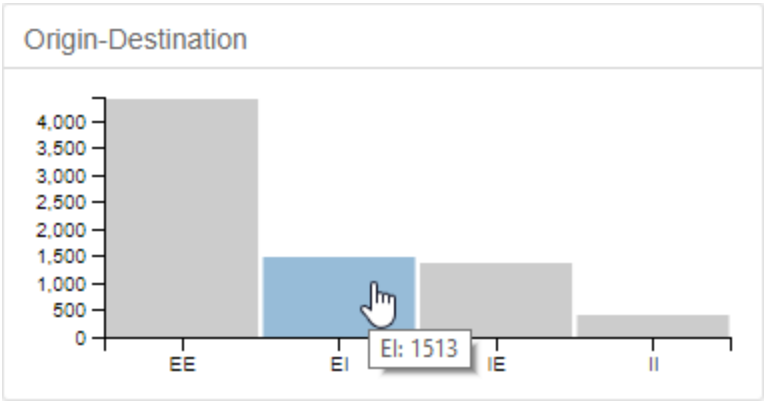

Figure 4: Filtering on a bar chart

2.2.1.2 Histograms

For continuous numerical data types such as time and distance which are visualized in histograms, the user may specify a value range by which to filter the dataset. Once a range has been provided, only data points whose value falls within the specified range are included in the selection. The value range window is specified by providing a minimum and maximum value, by clicking and dragging on the histogram. Once a window has been specified on the chart, it may be moved (changing the limit values while maintaining the range) or stretched (modifying the range by dragging one of the limit values). Only one filter window may be active at any time.

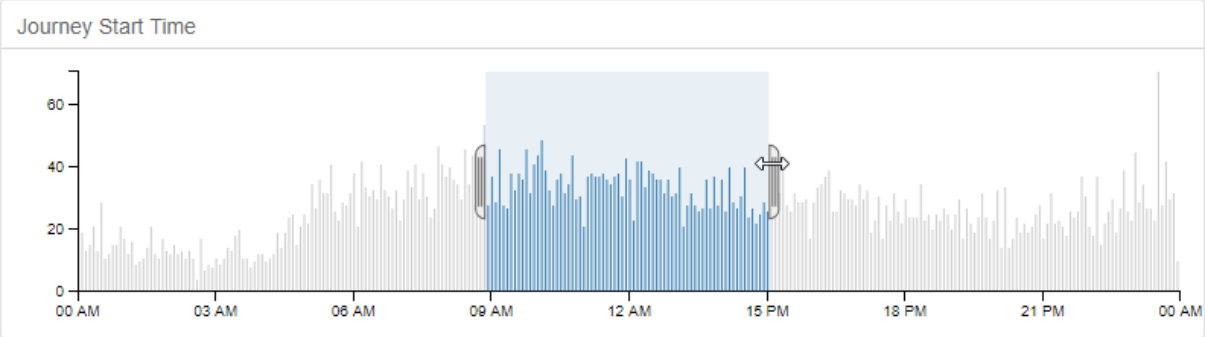

Figure 5: Filtering on a histogram

2.2.2 Maps

Users are able to apply spatial filters to the dataset as well, to identify activity relevant to a particular area. This is done by specifying a boundary and selecting trips that pass through the specified polygon. Spatial filters can be applied through the webmap, by using the shape drawing tools available in the map interface. The filter polygon may be one of two geometry types: a rectangle, or a non-self-intersecting polygon with vertices specified by the user. Multiple filter polygons may be provided; in the case when more than one filter polygon is provided, filter polygons are applied in conjunction, i.e. a given trip is included in the filtered set if and only if it passes through all the filter polygons.

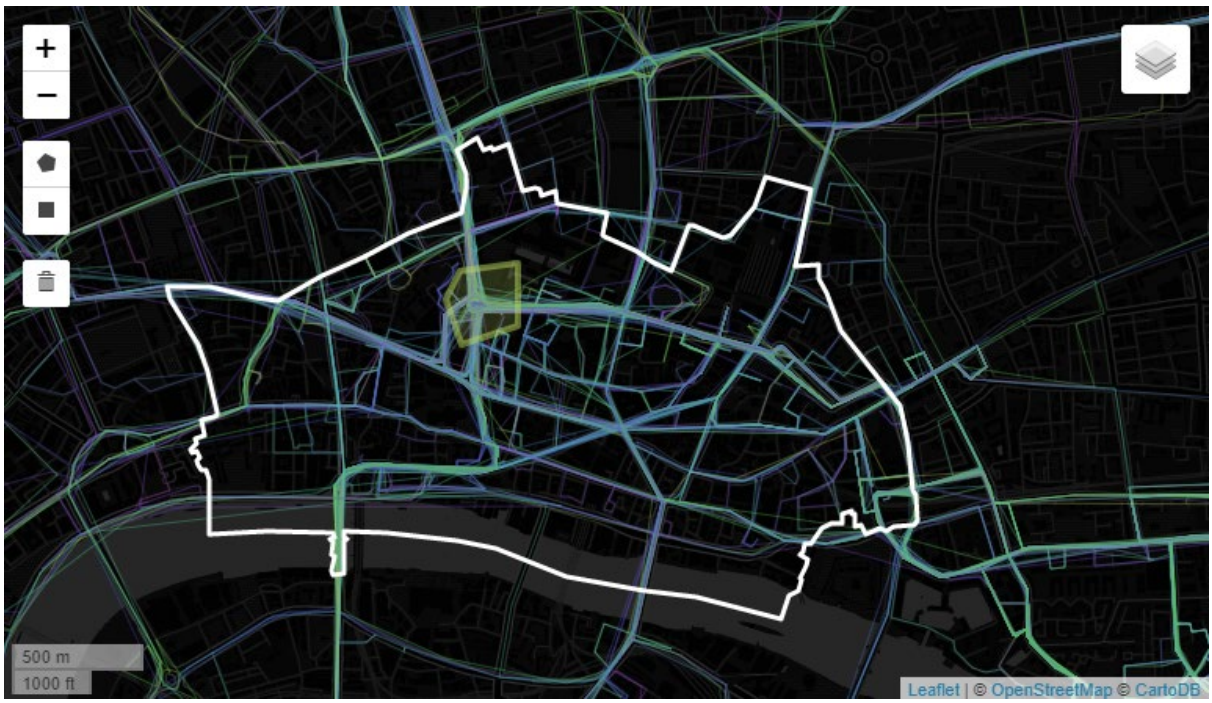

Figure 6: Spatial filtering. By providing a filter polygon, a user can select all trips passing through the area.

3 Data Formatting

The majority of data sets used in the deployed dashboard are secondary data, generated by analysing raw data. Primary data come in the form of a waypoint sequence, consisting of a series of

geocoordinates arranged chronologically as reported by vehicle GPS loggers, with some additional meta-data attached including vehicle id, vehicle weight class, etc. The raw data is then filtered into trips, by identifying stop locations and splitting the combined waypoint series into individual journeys done by each individual vehicle. A second-pass analysis is then applied to the trip data, to generate information about each trip, including the total trip length, information regarding the origin and destination of the trip, start and end times, etc. to be visualised in the dashboard.

A further spatial clean-up step is applied to the trip data, by performing a map matching pass on the path geometry, in order to snap the trip to the street network. The map matching pass is performed via the Open Source Routing Machine's[4] 'Match' service, which uses OpenStreetMap[5] data to represent street networks. The output is a new path geometry that is guaranteed to be on the street network, therefore removing any error. Additionally, by tracing the route on OpenStreetMap data, the ids for each street segment and node visited during the trip are able to be extracted, therefore allowing the subsequent spatial filtering in the dashboard.

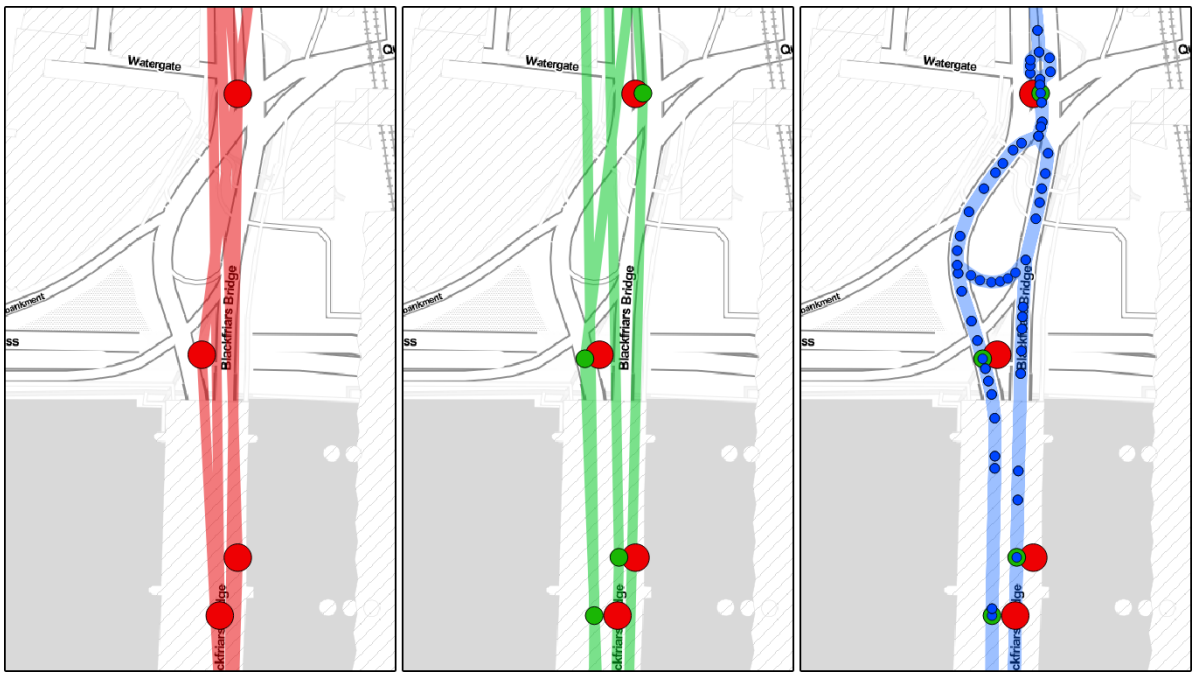

Figure 7: Map matching. Left: Raw waypoint data. Middle: Waypoints snapped to the street network. Right: Interpolation between known visited points, to reconstruct the full path.

The final data used in the dashboard is in three formats:

1. A dataset containing general information about each trip, including trip start and end time, trip length, information about origin and destination, etc.
2. A dataset containing the raw waypoints for each trip, sorted chronologically.
3. The reconstructed (map matched) trip path geometry, in GeoJSON[6] format.

[4] http://project-osrm.org/
[5] https://www.openstreetmap.org
[6] http://geojson.org/

4 Deployment

The deployment of the dashboard is done in two parts: The backend (server-side) containing the database and a server for handling requests and data queries, and the frontend (client-side) which contains the visualisation and interaction elements. The dashboard is accessed through any modern web-browser, by pointing to the relevant URL.

The database containing all the datasets discussed in the previous section (section 3) is a MongoDB instance[7], running version 3.6.4. Request handling and queries to the database are done through a server application written in JavaScript using the node.js environment[8]. The dashboard backend (server and database) is stored locally at the current stage, however the technologies used for each of the components (node.js and MongoDb) are designed primarily for web applications and therefore the dashboard can be easily deployed online.

The frontend part of the dashboard (the user interface) is written in HTML and JavaScript, combining multiple visualisation, charting, and mapping libraries. The layout and overall User Interface uses an HTML dashboard template published by Keen[9], modified to accommodate the required elements. The visualisation and filtering functionality is implemented in JavaScript. Specifically, overall data manipulation is handled using the dc.js library[10], which in itself combines more fundamental libraries: d3.js[11] is used for data visualisation in the form of charts, and Crossfilter[12] for filtering the data over multiple attributes. Finally, mapping and spatial filtering is implemented through the use of the Leaflet[13] library.

5 Appendix

The code used for the development of the dashboard will be made available at:

https://github.com/cheliotk/ftc2050-dashboard

[7] https://www.mongodb.com/
[8] https://nodejs.org/
[9] https://github.com/keen/dashboards
[10] https://dc-js.github.io/dc.js/
[11] https://d3js.org/
[12] http://crossfilter.github.io/crossfilter/
[13] https://leafletjs.com/