Course Project 2 – Trajectory and Road Network Data Analysis

AI6128 Urban Computing

Kishore Rajasekar
Nanyang Technological University
G2101949G
Singapore
kishore004@e.ntu.edu.sg

Seah Xiu Xuan Sherbelle
Nanyang Technological University
G2102418J
Singapore
sseah013@e.ntu.edu.sg

Sean Goh Ann Ray
Nanyang Technological University
G2202190G
Singapore
sean0057@e.ntu.edu.sg

Sita Rajagopal
Nanyang Technological University
G2202266H
Singapore
sita001@e.ntu.edu.sg

***Abstract*— Trajectory and road network analysis can aid in optimizing electronic dispatch systems. In this project, the team analyzes trajectories derived from taxi trips and the road network data of Porto City, Portugal. The raw GPS points collected by the mobile data terminals in the vehicles will be visualized to point out possible misalignments with the road network. Thereafter, map matching is performed with FastMapMatching [1], a python library based off Hidden Markov Models (HMM). The corrected trajectories are analyzed and possible improvements for map matching are listed out by the team.**

I. Introduction

With the prominence of Google Maps and other web mapping platforms, many vehicles inadvertently have their movements sensed in real time when travelling. Such spatiotemporal data is called a trajectory. Raw trajectories collected from vehicles involve noise and their accuracy highly depends on the instrument measuring and calculating them. A common practice to overcome such inaccuracies is to map trajectories to road networks before they are analyzed.

Map Matching is a technique that assigns points to a location on a digital map. A common framework used for Map Matching is the Hidden Markov Model (HMM), where the most likely road route is found given a time-stamped sequence of latitude and longitude pairs [2].

This report contains 6 tasks and outlines how each task was performed. Trajectory and road network analysis is performed using taxi trip information and the road network of Porto City, gathered using crowdsourcing efforts by OpenStreetMaps. Thereafter, map matching is performed before deriving conclusions about the most prominent road segments that are travelled by taxis, as well the road segments that may cause delays due to a longer travel time on average. Such conclusions can ultimately improve the efficiency of electronic dispatching systems adopted by taxi companies, where taxi fleets can be managed more easily, and customers can avoid delays when they are assigned to taxis that are nearest to them during peak hours.

II. Individual Contributions

For this report, Kishore worked on the process of installing FMM into a Google Colab notebook instance, as well as the route analysis of most traversed road segments (Task 5.1). Sherbelle worked on the map matching and subsequent road visualization with FastMapMatch (Task 3 and 4), as well as the route analysis of most traversed road segments (Task 5.1) and road segments with the longest average travel time (Task 5.2). Sean worked on the data preparation (Task 1), GPS Point visualization (Task 2), route analysis of road segments with the longest average travel time (Task 5.2), and the bonus task. Sita worked on the map matching and subsequent road visualization with Leuven MapMatching (Task 3 and 4), as well as the route analysis of road segments with the longest average travel time (Task 5.2).

III. Dataset Preparation

A. Road Network of Porto City, Portugal

The road network data of Porto City, Portugal was downloaded using an open-source tool called OSMnx [3]. OSMnx is a powerful tool which can download different network types from OpenStreetMap. A bounding box of the city was first defined and using OSMnx to download the 'driveable' network, the network shown in Fig. 1 was produced.

Fig. 1. Road Network of Porto City, Portugal

B. *Taxi Trajectory Data*

The downloaded taxi trajectory data from [4] comes as a table with 1000 rows of data with 9 columns and they are described in TABLE I.

TABLE I. TAXI TRAJECTORY DATA DESCRIPTION

Column Name	Description
TRIP_ID	A unique identification number for each trip
CALL_TYPE	How the taxi was requested. 'A' refers to phone, 'B' refers to taxi stand, and 'C' refers to flag down
ORIGIN_CALL	A unique identification number for each phone number used for a taxi request (CALL_TYPE 'A' only)
ORIGIN_STAND	A unique identification number for each taxi stand (CALL_TYPE 'B' only)
TAXI_ID	A unique identification number for each taxi driver
TIMESTAMP	The start off each trip, in Unix time standard
DAY_TYPE	The type of day. 'A' refers to normal weekdays and weekends, 'B' refers to holidays, and 'C' refers to the eve of the holidays
MISSING_DATA	Boolean. 'False' means all GPS data is present while 'True' means one or more GPS points are missing
POLYLINE	A list of raw GPS coordinates

The 'POLYLINE' column is a list of raw GPS coordinate pairs, where each pair of coordinates is the longitude and latitude, in a form of a string. These pairs were recorded on a 15-second interval, with the first pair being the start of the trip and the last pair being the destination.

This string of GPS coordinates was first converted to a float array, with 2 columns and number or rows being equal to the number of GPS coordinate pairs. This allows an easier manipulation of the data, mainly for the GPS point visualization and for the open-source tools.

IV. GPS POINT VISUALISATION

Using OSMnx, the first 10 trips are plotted on the map by looping through the 'POLYLINE' data of each trajectory and is shown in Fig. 2 below.

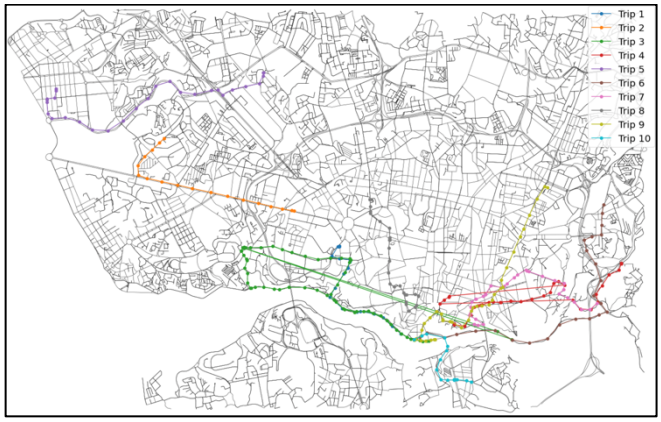


Fig. 2. Visualization of First 10 Trips

A quick observation of Fig. 2 shows that trips 3 (green) and 4 (red) have slightly corrupted 'POLYLINE' data due to the sudden change in their trajectories because of 2 or 3 corrupted GPS points.

V. MAP MATCHING

This section will cover Tasks 3 and 4, to perform the mapping of the taxi trajectory data to the road network, and then to visual the routes of the first 10 trips. Two different open-source tools are explored, Fast Map Matching (FMM), and Leuven Map Matching.

A. *Fast Map Matching (FMM)*

FMM is an open-source algorithm that integrates a Hidden Markov with pre-computation [2]. The precomputation stage stores all pairs of shortest paths (within a fixed threshold) in an upper bounded origin-destination table. Thereafter, the map matching stage uses a hash table search, instead of computationally expensive routing queries. This allows FMM to be able to match 25,000 – 24,500 points within a second, depending on the output mode.

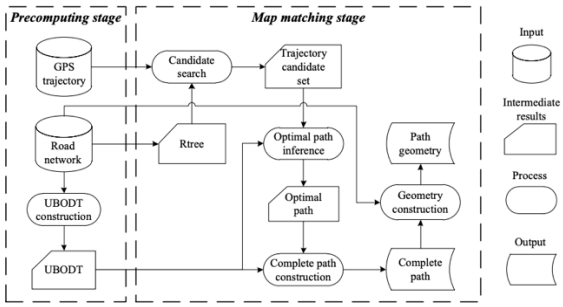


Fig. 3. Process of Fast Map Matching Algorithm

1) *Using FastMapMatch*

In the pre-computation stage, the extracted Porto's drivable road graph network is passed into `UBODTGenAlgorithm` in FMM, which generates an Upper Bounded Origin Destination Table (UBODT) file. This file holds pairs of shortest paths in the graph network that has the length that is under a delta value of 4. This generated table, along with the graph network, is passed into `FastMapMatch` to generate a FMM model.

Preprocessing of the taxi trajectory data was done as well. The POLYLINE column in the taxi trajectory data was first converted into this format: "LINESTRING(lon_1 lat_1, lon_2 lat_2, …, lon_x lat_x)" where lon_n refers to the longitude of the nth point and lat_n refers to the latitude of the nth point.

Each preprocessed taxi trajectory data record is then passed into the `match_wkt` method of the FMM model, along with the following configuration values: 4 candidates, a search radius of 0.4, and a gps error of 0.5.

Two outputs of this map matching processed is saved for each mapped trajectory. The first is cpath, which is the path traversed by the trajectory. This comprises of a list of integers, each of which maps to the fid of the graph network's edges. The other is mgeom, which holds the geometry of the traversed path, in the same format as the pre-processed trajectory data.

a) *Visualisation of First 10 Trips*

As the geometry of the traversed path holds the latitude and longitude of each mapped point, this can be easily plotted. The result is shown in Fig. 4. The mapped trajectories

introduce more points that can be plotted, as shown by the increased number of markers for each trajectory.

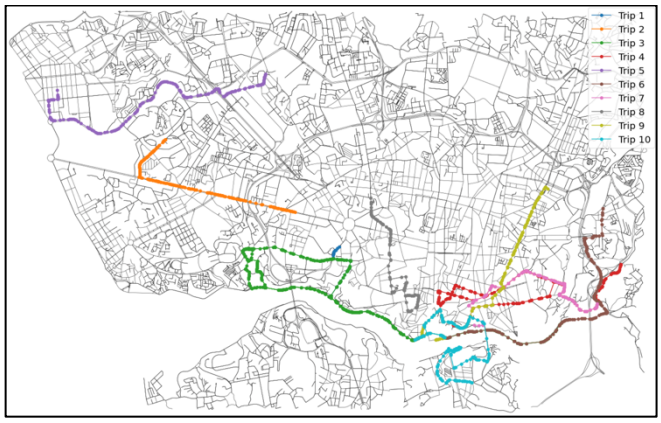


Fig. 4. Visualization of First 10 Trips Trajectory using FastMapMatch

When comparing against Fig. 2, most trajectories tend to match the GPS locations well, such as Trips 1, 2, 5, 6, 7, 8 and 9. For Trips 3 and 4, as the 'POLYLINE' data seems to be corrupted, it is also expected that the mapped trajectory would not have a good match as well.

For Trip 10, however, the mapped trajectory is not as expected. Fig. 5 shows both the original GPS coordinates (blue dashed line with markers) and the matched trajectory (orange dashed line). Instead of taking the bridge on the left-hand side, the mapped trajectory routes around to take the bridge on the right-hand side. The grey lines in the map indicate the road network for driving. As seen in Fig. 5, a grey line segment is not present in the taxi path for the bridge on the left-hand side. Upon further inspection, the team had identified that the bridge in question as the *Dom Luís I* bridge, that has both pedestrian and vehicle crossings. It was not listed as part of the road network downloaded from OSMnx. Such an error may have occurred due to the crowdsourcing nature in which the network data is collected by OpenStreetMaps.

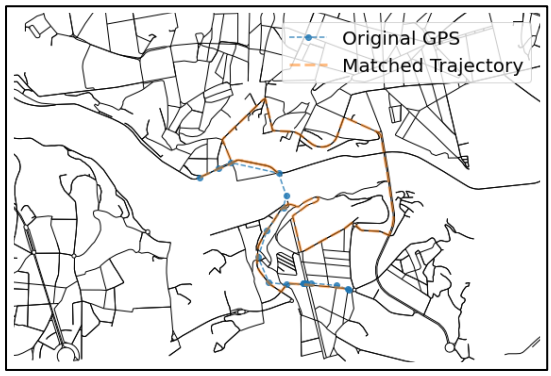


Fig. 5. Original GPS Points and Matched Trajectory of Trip 10

2) Using STMatch

The map matching process is also done using the other algorithm provided in FMM, `STMatch`. According to the documentation, `FastMapMatch` is recommended for small and middle scale networks, while `STMatch` is recommended for large scale networks.

The STMatch model does not make use of the UBODT file, instead only taking in the graph network as its inputs for the initialization. Like the FMM model, each preprocessed taxi trajectory data record is then passed into the `match_wkt` method of the STMatch model, along with the same configuration values, in addition to using a maximum speed of 30 and a scale factor of 1.5.

a) Visualisation of First 10 Trips

The mapped trajectories of the first 10 trips are plotted and shown in Fig. 6. The mapped results look like what was plotted in Fig. 4, which indicates no additional advantage of using the STMatch model over the FMM model specifically for Porto City and this dataset.

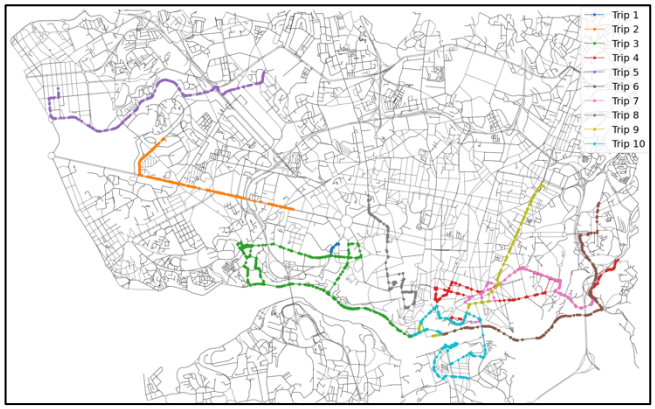


Fig. 6. Visualization of First 10 Trips Trajectory using STMatch

B. Leuven MapMatching

An alternative tool, called Leuven MapMatching, was also used to compare against FMM. Leuven MapMatching is based on a Hidden Markov Model (HMM) with non-emitting states [5]. It aligns a trace of GPS coordinates to a map or road segments.

The road network data of Porto City from OSMnx is provided as an input to the `InMemMap` object. It is an in-memory representation of a map that is a database-like object to perform experiments with map matching. It contains a parameter called `graph`, which allows for the initialization of a graph using the Porto City Road Network.

The polylines recorded in the taxi trajectory data are individually parsed into an object known as a `DistanceMatcher`. It performs map matching, considering the distance between the observations found in the above-mentioned dataset and the nodes in the `InMemMap` graph. The parameters listed in TABLE II. were set for the `DistanceMatcher`.

TABLE II. PARAMETER DESCRIPTION FOR DISTANCEMATCHER

Name	Description	Value
max_dist	Hard-cut maximum distance from path. This ensures that not all nodes in the graph are visited for matching.	200
min_prob_norm	Minimum normalized probability of observations (EMA)	0.1
non_emitting_length_factor	Reduce the probability of a sequence of non-emitting states the longer it is. This can be used to prefer shorter paths.	0.95
non_emitting_states	Allow non-emitting states. A non-emitting state is a state that is not associated with an observation. Here, it is assumed a state can be associated with a location in between two observations to allow for pruning.	True

Each trajectory is visualized individually using the plotting function provided by the library and sample visualizations can be found in Fig. 7 and Fig. 8. The cyan polyline connects the matched nodes. The blue line and dots are the initial trajectory inputted to the matcher. As seen in Fig. 7, a matching node can be found for every coordinate in the trajectory, leading to a completely matched trajectory. For Fig. 8, however, matching nodes are only found up to the *Dom Luis I* bridge and the remaining path cannot be matched to a continuous line segment within the constraints provided to the `DistanceMatcher`. Such an occurrence, once again, may be attributed to the sparsity of the nodes in the road network. Fig. 9 shows the network, in purple, overlayed against the map and trajectory of Trip 10. Once again, the *Dom Luis I* bridge is not contained within the purple dots, indicating the nodes in the network. Therefore, the continuous line segment is broken, and the remaining path cannot be mapped. Individual visualizations for the first 10 trips can be found in Appendix A. The algorithm in Leuven MapMatching is far more rigid than FMM in yielding a trajectory and is prone to providing partial trajectories, given sparse road driving networks.

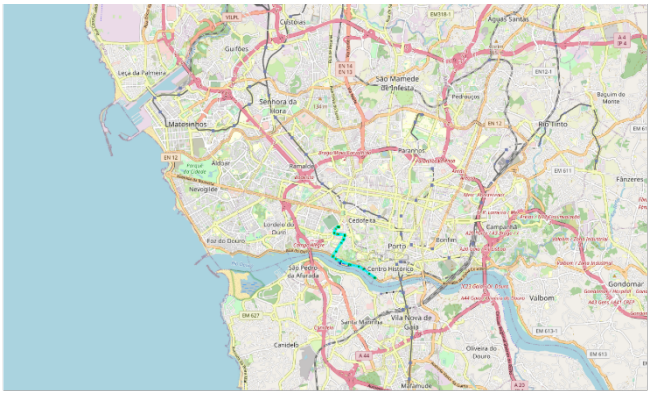


Fig. 7. Visualization of Trip 1 using Leuven MapMatching

Fig. 8. Visualization of Trip 10 using Leuven MapMatching

Fig. 9. Visualization of Trip 10 using Leuven MapMatching and Network Overlay

VI. Route Analysis

A. 5 Most Traversed Road Segments

The fid from the matched paths are retrieved and then utilized the mapping of fid to osmid to retrieve the count of the top 5 most frequently occurring osmids. The trajectory fid is proceeded to be mapped with the edges file. The top 5 road segments counts are extracted. Fig. 10 represents the top 5 road segments which are traversed the most often.

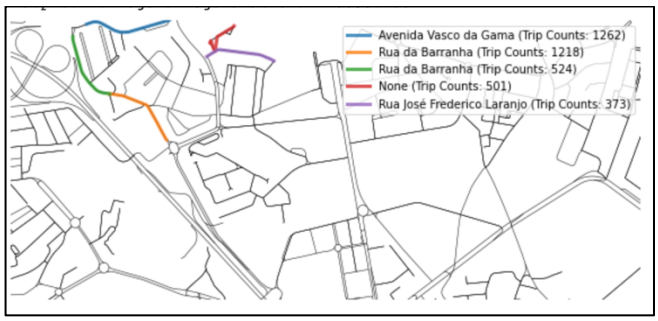


Fig. 10. Visualization of Top 5 road segments that are traversed the most

B. 5 Road Segments with Largest Average Travelling Time

Based on the number of GPS points collected in the raw trajectory data, the total time taken for each trip can be calculated as:

$$N_{GPS\ Points} - 1 \times\ 15\ seconds$$

where $N_{GPS\ Points}$ refer to the total number of GPS points.

Based on the assumption that the taxi travels at a constant speed across the entirety of the journey, the average travel time of each mapped fid segment can be found by taking the total time divided by total road segment length, divided by the fid segment length.

Thereafter, the average time per unit length of each fid is calculated, before mapping the fid average travel time back to its osmid. Now, for each osmid, the average of the average time per unit length, as well as the total length of all fid segments, are found.

Lastly, the average time per unit length is multiplied against the total length of all fid segments to get the average travelling time for each road segment.

The top 5 road segments with the longest average travelling time are then identified and plotted into Fig. 11. The

route plotted by the blue line has the longest average travel time of 6.51 minutes. However, this may also be because the road segment is very long, stretching for 4.3km (based on the calculation approach above).

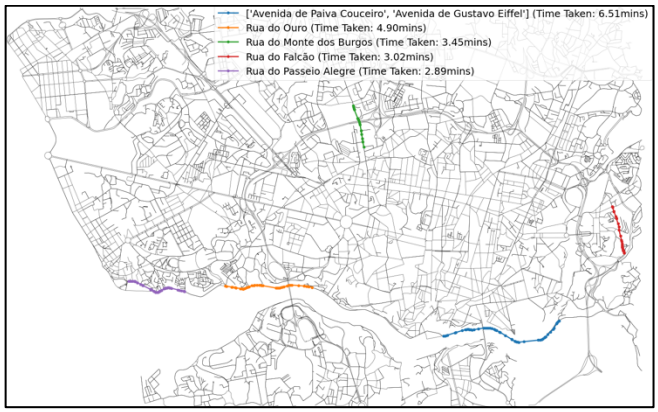


Fig. 11. Visualization of Top 5 Road Segments with Largest Average Travelling Time

However, there are limitations to this analysis due to the assumptions made, chiefly the one that all fids in each osmid are non-overlapping. As can be seen from Fig. 12 and Fig. 13, the same road segment of Avenida de Paiva Couceiro and Avenida de Gustavo Eiffel was calculated to be 4.3km, when in actuality it is only 2.2km. This may dramatically increase the amount of travel time attributed to this road segment due to the overestimated length.

Fig. 12. Mapped road segment of Avenida de Paiva Couceiro and Avenida de Gustavo Eiffel (4.3km).

Fig. 13. Actual length of Avenida de Paiva Couceiro and Avenida de Gustavo Eiffel from Google Maps (2.2km).

VII. Bonus Task

There are cases where the raw GPS coordinates from the 'POLYLINE' data seems to be corrupted, which thus resulted in the map matching algorithm not working very well. Fig. 14 below shows 4 such cases of the corrupted raw GPS trajectories, including trips 3 and 4 mentioned earlier.

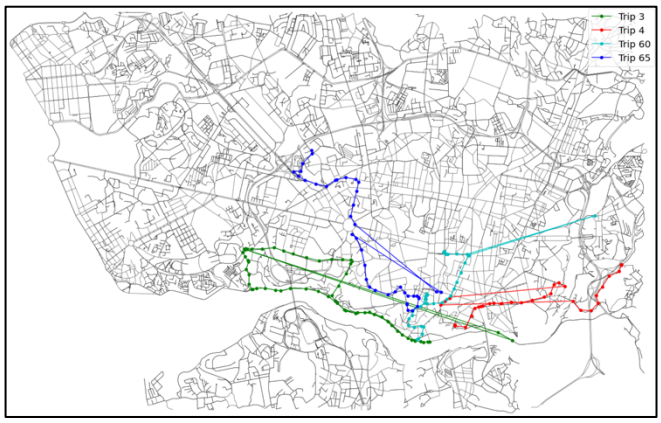


Fig. 14. Visualization of 4 Trips 'Corrupted' Trajectory

The approach is to remove the outlying GPS coordinates. For every trajectory, the distance between one GPS point to the next is first computed, followed by getting the average of those distances. For example, by setting a threshold of 5, if the distance between point 'P' and the next point 'Q' is 5 times more than the average, then that next point 'Q' is recognized as an outlier.

The outlier point 'Q' is then removed from the trajectory, and the process continues by comparing the same point 'P' with point 'R', which is the point after the removed point 'Q'. This process repeats until all the outliers are removed from the trajectory. Fig. 15 shows the results of the outlier removal.

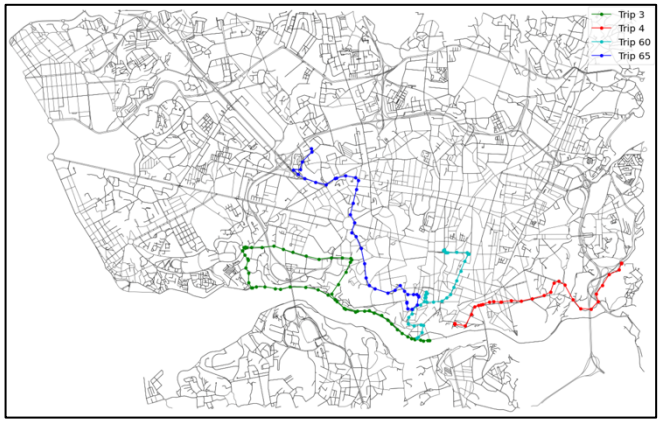


Fig. 15. Visualization of 4 Trips 'Corrupted' Trajectory After Outlier Removal

The set of data without outliers is then passed through the same FMM algorithm mentioned in Section V for the map matching. Fig. 16 shows the 4 trips trajectory before outlier removal and after map matching, while Fig. 17 shows the 4 trips trajectory after outlier removal and after map matching.

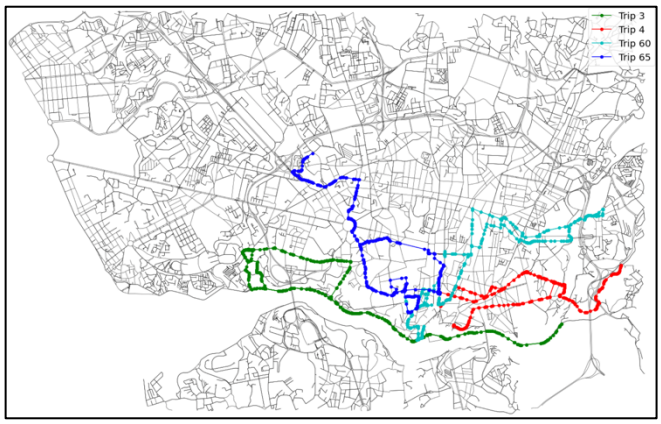


Fig. 16. Visualization of 4 Trips 'Corrupted' Trajectory Before Outlier Removal using FastMapMatch

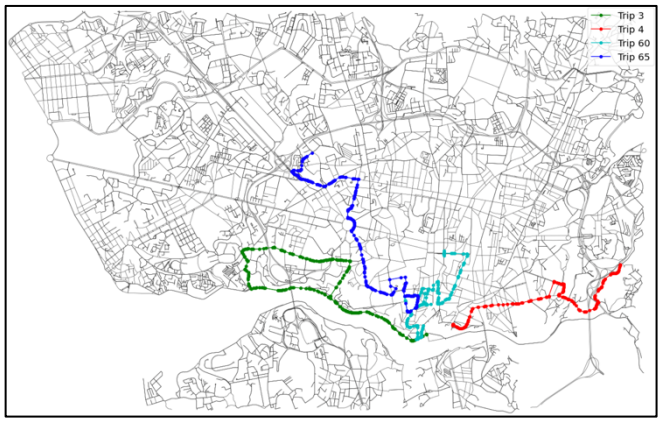


Fig. 17. Visualization of 4 Trips 'Corrupted' Trajectory After Outlier Removal using FastMapMatch

Clear improvements in the map matching can be seen on all the 4 trips after removal of the GPS outliers, where the unnecessary 'detours' are gone, most obviously seen in trips 4 (red) and 60 (cyan).

VIII. Conclusion

In this project, the road network of Porto City, Portugal from OpenStreetMap was used in combination with the taxi trajectory data from Kaggle to understand the end-to-end process of mapping GPS points to road networks. The GPS points of the first 10 trips were visualized and issues with the dataset, such as corrupted GPS locations, was noted. The GPS points were then fed into two open-source tools, Fast Map Matching and Leuven MapMatching, and the corresponding trajectories of the first 10 trips were visualized. Limitations of both tools were noted. Then, route analysis was performed to find the top 5 most traversed road segments, and the top 5 road segments with the longest average travelling time. Lastly, some efforts were made to improve on the map matching algorithm, with the pre- and post-improvement trajectories being compared.

References

[1] C. Yang and G. Gidofalvi, "Fast Map Matching, an Algorithm Integrating Hidden Markov Model with Precomputation," International Journal of Geographical Information Science, vol. 32, no. 3, pp. 547-570, 2018.

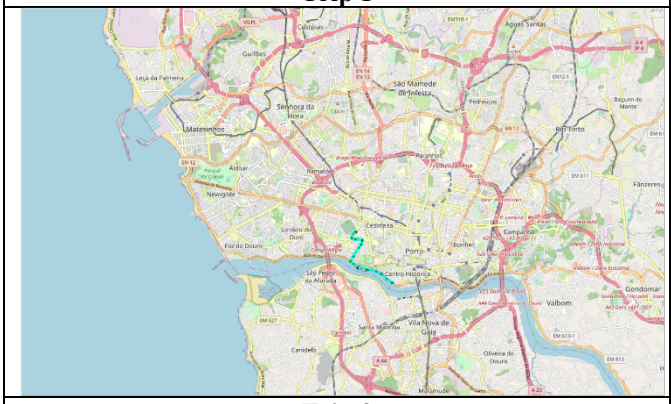
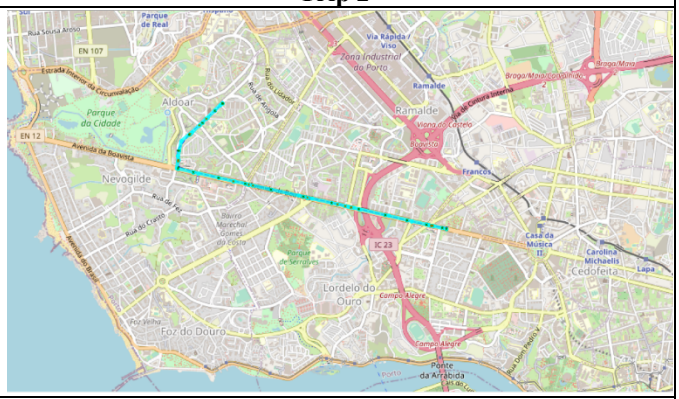
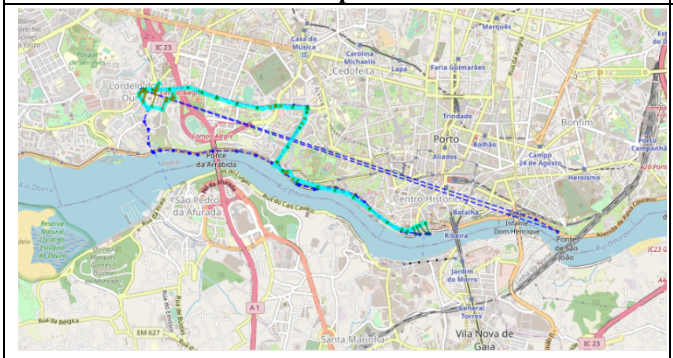
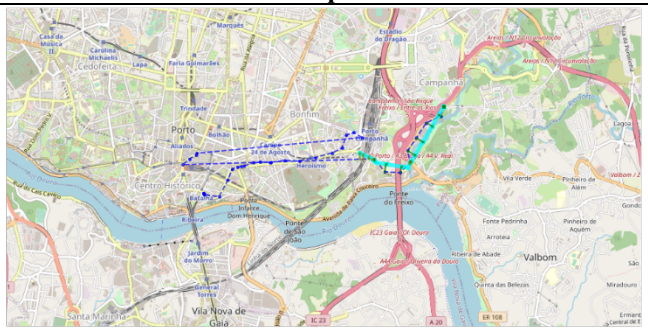
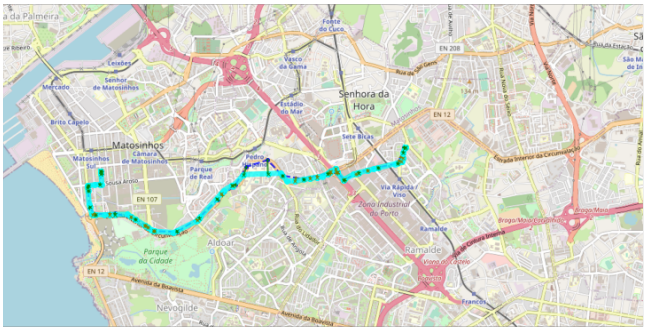
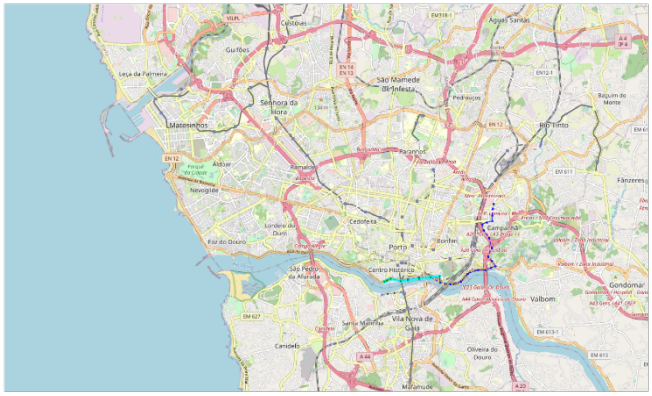
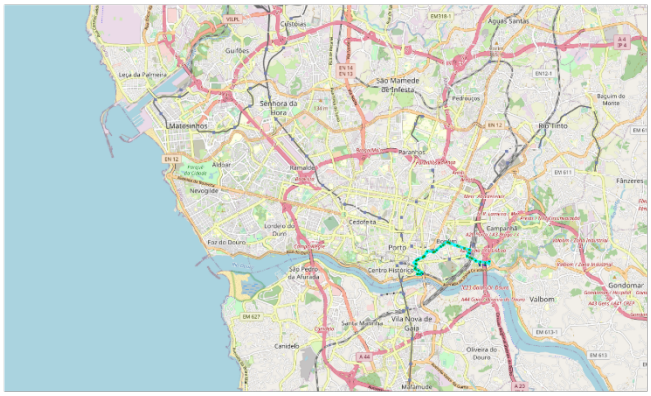
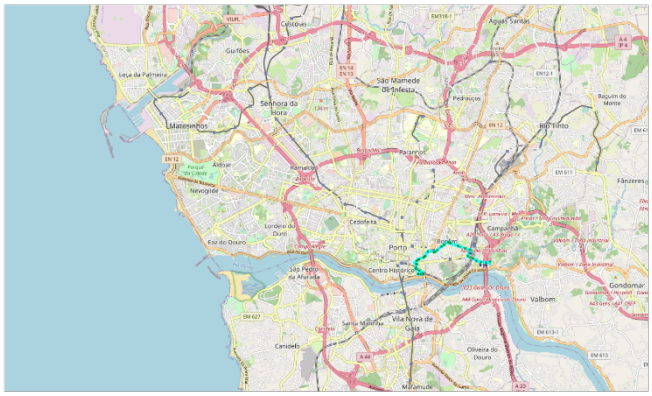
[2] Newson, P., and Krumm, J. (2009). Hidden markov map matching through noise and sparseness. Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '09. https://doi.org/10.1145/1653771.1653818

[3] OSMnx 1.2.2 - OSMnx 1.2.2 documentation. [Online]. Available: https://osmnx.readthedocs.io/en/stable/. [Accessed: 10-Nov-2022].

[4] ECML/PKDD 15: Taxi Trajectory Prediction (I). [online]. Available: https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data. [Accessed: 10-Nov-2022].

[5] M. Wannes, M. Verbeke, "HMM with Non-Emitting States for Map Matching", European Conference on Data Analysis (ECDA), Paderborn, Germany, 2018.

APPENDIX A – VISUALIZATION OF LEUVEN MAP MATCHING

Trip 1	Trip 2
	
Trip 3	**Trip 4**
	
Trip 5	**Trip 6**
	
Trip 7	**Trip 8**
	
Trip 9	**Trip 10**