Attn: Prof Long Cheng

AI6128 Asssignment 2

Group member names	Contribution
Tian Mingchuan	Strive for completion of Report
Wang Duosheng	Strive for completion of Report
Teng Jie	Strive for completion of Report
Teng Guang Way	Strive for completion of Report

Each member in the group contributed to assignment equally and is therefore appropriate to assume that each member has equal contribution.

Task 2 Visualizations of the GPS points (2 Pages)

Figure 1 - Raw GPS point of TrajectoryID 00 to 04 (First 5), with annotations. All coordinates in (Longitude, Latitude) format. Trajectory 2 and 3 contains some noise points resulting in a bizarre looking visualization.

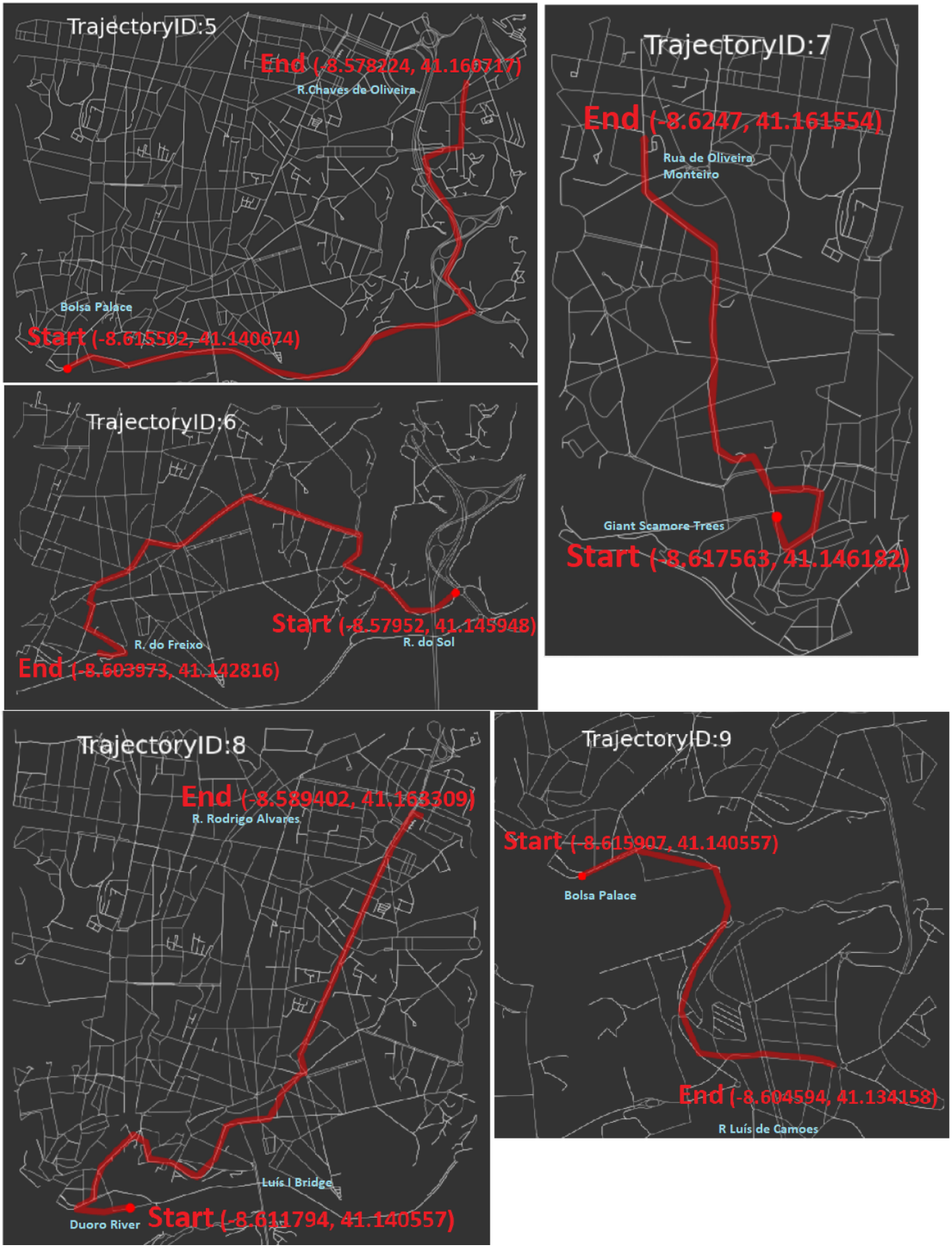

Figure 2 - Raw GPS point of TrajectoryID 05 to 09 (next 5), with annotations. All coordinates in (Longitude, Latitude) format.

Task 4 Visualization of Mapped Routes (2 Pages)

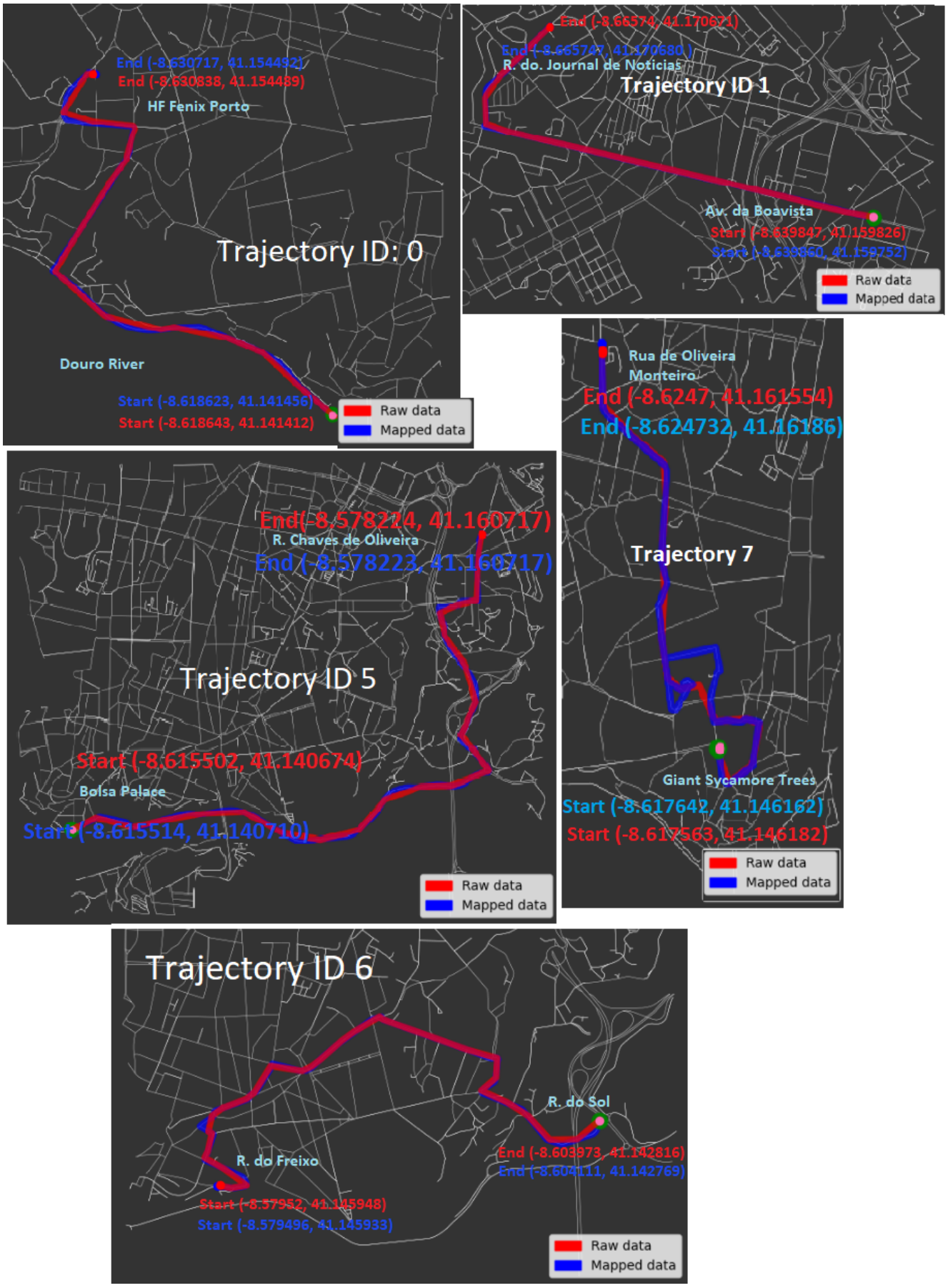

Figure 3 - Trajectory 0, 1, 5, 6, 7 mapped routes visualization (First 5 out of 10), with annotations. All coordinates in (Longitude, Latitude) format. Well mapped trajectory (0,1,5,6) looks very similar to the GPS trajectory when zoomed-out at a high level view.

Figure 4 - Trajectory 8, 10, 12, 13, 14 mapped routes visualization (Last 5 out of 10), with annotations. All coordinates in (Longitude, Latitude) format. All the routes in this figure appear reasonably routed.

TASK 5(1) and 5(2) (1 Page)

Algorithm Explanation

Task 5(1): After task 3, we can get the edge indice sequence of each trajectory. Each indice sequence is a list containing unique indexes . In order to get 5 road segments that are traversed the most often, we count the number of occurrences of each index in all trajectory data and return the indices with the top five occurrences.

Task 5(2): For each trajectory data, we know the raw GPS point sequence and the edge indice sequence. If we want to compute the average traveling time of an edge, we need to first compute the traveling time of this edge in each trajectory data in which it appears and then do the average. An algorithm is used to associate each of the trajectory GPS points to a relevant edge. At the beginning, we set the first edge in the edge indice sequence as the **active edge**. For each GPS point in the trajectory data, we compute two harversine distances: 1) the distance between the 'origin u' and 'destination v' of the **active edge**. 2) the distance between the current GPS point and the 'origin u' of the **active edge**. If the distance 1) is longer than distance 2), we can assume this GPS point lies on the **active edge** and assign +1 point to it and move to the next GPS point. If distance 1) is shorter than distance 2), we can assume this GPS point lies on the next edge. Therefore, we can assign +1 point to the incoming edge as per the edge indice sequence. We then update the **active edge** as the incoming edge and move to the next GPS point. After all the trajectory data has been traversed, each edge has a total score and we can compute the average score of each edge. The average approximate time of an edge can be calculated as 15s * average score of the edge since it was mentioned in the Kaggle dataset that each GPS point was taken at 15s interval.

Source code of above algorithm provided in *Task5_12_src.ipynb*

TASK 5(3), Single Figure annotated Figure 5

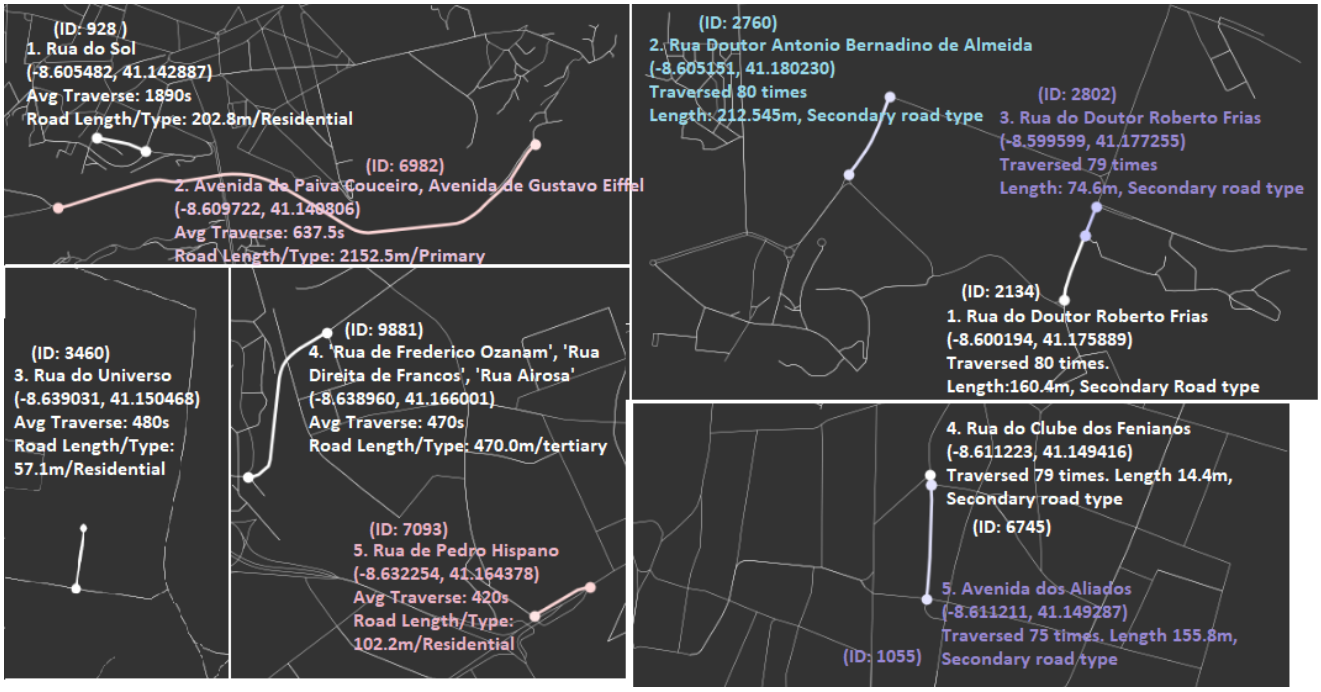

Figure 5 The segments annotated pink on the left shows the visualization result of road segments having the longest travel time for task 5(2). The segments annotated purple on the right shows the visualization of road segments with the most visits for task 5(1). Ranking numbers are indicated beside the road name. All coordinates in (Longitude, Latitude) format.

TASK 6 (5 Pages)

After the work in Task 3, 661 out of 1000 trajectories were successfully matched through the recommended Fast Map Matching Algorithm. The remaining 339 trajectories were not mapped successfully. In this part of the assignment we identified 2 main causes that resulted in the failure to map match some of these failed instances. We then conduct corrections to the map matching procedures to address this problem. The report uses the trajectory of *trajectoryID 4* as an example for illustrating how *main causes 1* were rectified by applying its respective fix, and trajectory of *trajectoryID 2* as a similar example to illustrate how *main causes 2* were also rectified by its respective fix. At the end of the assignment, 977 out of 1000 trajectories were map matched successfully. The remaining 23 trajectories which weren't mapped successfully were caused by many individual level problems. Some of them were discussed at the conclusion part of Task 6's report.

Cause 1: The map matching task uses the network map downloaded from task 1 using osmnx.

The map matching task (based on FMM algorithm) is based on Task 1 Porto city road network. This road network covering just the boundary area of Porto City is not sufficiently wide enough to contain all the GPS points in the trajectory dataset. There are many cases when trajectory instances actually contain GPS points beyond the boundary of Porto City.

For example using *trajectory ID 4*, as shown in the **Figure 6** below, the taxi origin (in green) may start from a road segment located at Rua do Viso located within osmnx package's Porto City boundary graph network. However, the destination of the trip ended at a location known as Rua de Dom Joao (in red) which was outside of the defined Porto City boundary graph network. As the map matching task uses the very same road network, as defined in Task 1, destination points describing the taxi road segment location outside the boundary are not recognized with any edges or nodes within the network graph. Therefore the map matching algorithm is not able to match any points outside of the defined graph, resulting in the failure to map match the given trajectory.

Figure 6 - Raw GPS Points trajectory of *TrajectoryID 4*. Start Point marked with Cyan color and End Point marked with Red. At some point (marked orange) and beyond, the trajectory went outside of the coverage of Porto City network defined by osmnx tool. All coordinates in (Longitude, Latitude) format.

***Cause 1* Fix:**

A new network was downloaded to rerun the FMM algorithm. This network was extracted within a predefined rectangle boundary. The boundary range is set such as the min X, max X, min Y and max Y encompass well the smallest Longitude, biggest Longitude, smallest Latitude and biggest Latitude of all the GPS points in the trajectory dataset. **Figure 7** below is an illustration of the map matched trajectory of *trajectoryID 4* which was illustrated in **Figure 6** earlier for not being able to map matches before applying fix.

The source code to provide all the route sequences in WSG84 points for visualizations were also submitted through the *Task6_src.ipynb* source code file. After applying the first fix, 944 out of 1000 trajectories were matched successfully by the FMM algorithm. The map matched routes are provided in the filename, *postprocessed_output03.csv*

Figure 7 - Map matched trajectory of *TrajectoryID 4* of FMM algorithm after applying fix. The extended map network allows proper map match of the entire route.

***Cause 2*: Wide distance between interval GPS points kaggle's trajectory dataset**

Some neighboring GPS points are situated too far apart. The FMM algorithm used in this project assignment requires the user to provide a hyperparameter input called Shortest Path(SP) distance thresholds. This hyperparameter would define the algorithm's search range of all pairs of shortest paths in the graph to form a list of possible solutions to connect the 2 points. If the search range is less than the situated distance between the 2 GPS points, then the FMM algorithm will not be able to provide any possible solution to match the 2 points, and therefore cannot match the entire route successfully. [1] Applying the map matching task will result in the map matching tool kit used in this project to prompt a warning message highlighting how certain trajectories have points that cannot be matched as shown in **Figure 8**.

Further investigating, **Figure 9** using *trajectoryID 2* as an example, we can note that there exist 2 unusual neighboring GPS points positioned at 33th and 34th, where they are situated far apart from each other. This means that the 2 points are very unlikely to be taken within 15s interval as mentioned in the kaggle dataset introduction due to certain unexpected events. For instance, the mobile data terminals installed in the taxis were switched off by the drivers or certain technical issues resulted in the occurrence of noise data that are not relevant to the vehicle's position. This situation will cause problems to the FMM algorithm if the SP distance threshold's hyperparameter is set below the distance between the 2 GPS points.[1] In task 3, the SP distance threshold was defined at 3.33km which was not enough to cover the two points which were situated 5.783km apart.

```
[2022-11-16 19:45:22.200] [warning] [fmm_algorithm.cpp:289] Traj 3 unmatched as point 33 and 34 not connected
```

Figure 8 - The FMM algorithm prompted a warning when running the map matching task highlighting point 33 and 34 of the trajectory are not connected causing the map making to fail for the particular trajectory.

```
Final Lats and Long List processed finished
comparison of trajectoryId:3 gps point pt33 and p34 [-8.6499, 41.154264] [-8.599383, 41.141736]
harv dist of 2 pts(m) 5783
```

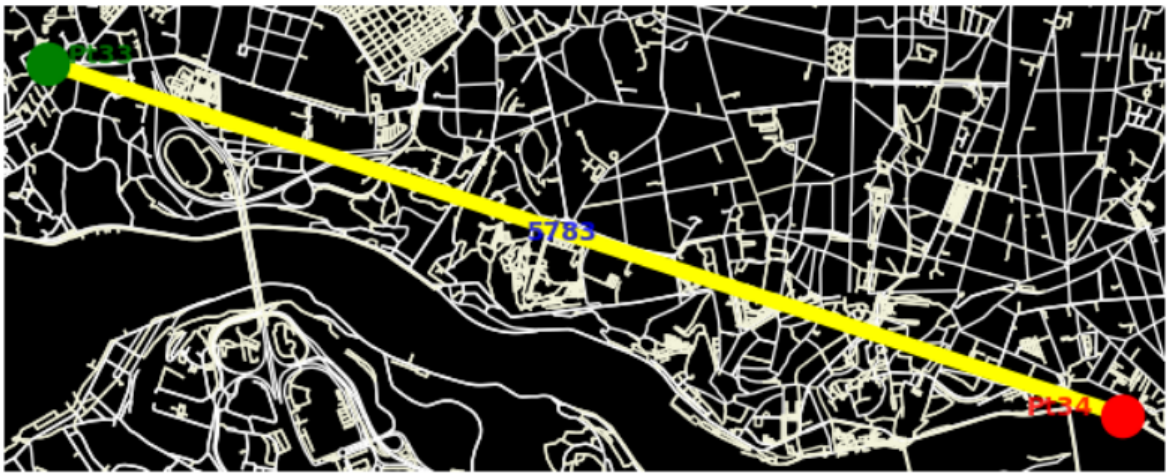

Figure 9 - 33th(Green label) and 34th(Red label) Raw GPS Points *TrajectoryID 2*. The 33th Raw GPS point is situated at Longitude=-8.6499 and Latitude = 41.154264 while the 34th Raw GPS point is situated at Longitude =-8.599383 and Latitude = 41.141736. It can be noted that the harversine distance between the 2 points are situated 5783 meters from each other.

***Cause 2* Fix:**

The float variable d defined in the source code refers to the SP distance threshold in the FMM algorithm. This was increased from 0.03 degrees to 0.06 degrees. 0.03 degrees in WSG84 unit would translate to approximately 3.33km (0.03 * 111km), while 0.06 degrees translate to 6.66km. After applying the second fix, we were able to approximate a connected route between the 2 points as shown in **Figure 10** on TrajectoryID 2. **Figure 11** below is an illustration of the entire map matched route of trajectoryID 2. Even though the map matching algorithm was able to approximate a complete route, it may not be exactly the trajectory which was actually traversed by the vehicle. To solve the root problem taxi drivers must keep their mobile data terminal functional at all times such that GPS points of the journey's trajectory are properly tracked throughout the journey.

The source code to provide all the route sequences in WSG84 points for visualizations were also submitted through the *Task6_src.ipynb* source code file. In total 977 out of 1000 trajectories were matched successfully by the FMM algorithm after applying *Cause 1* and *Cause 2* fix. The complete map matched routes output given all 1000 trajectories are provided in the filename, *postprocessed_output06.csv*

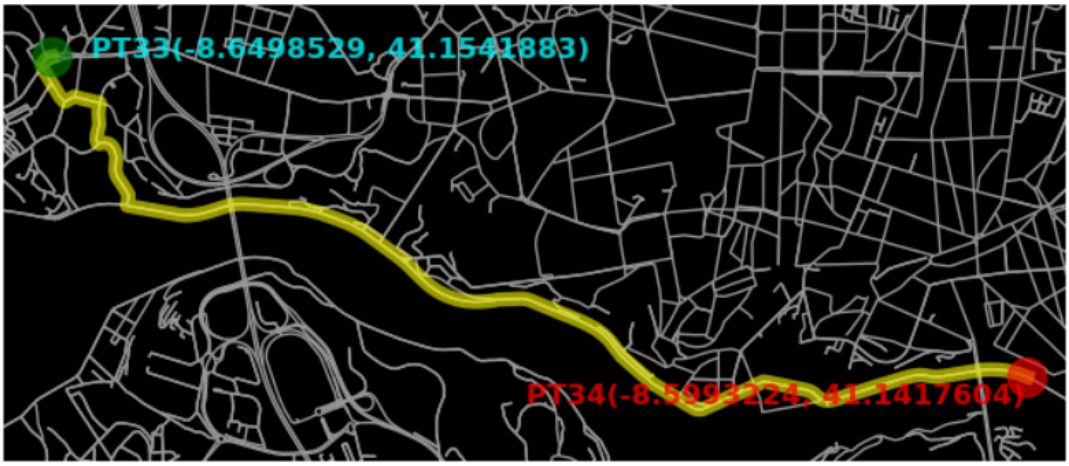

Figure 10 - 33th(Green label) and 34th(Red label) Raw GPS Points *TrajectoryID 2*. The 33th Raw GPS point is situated at Longitude=-8.6499 and Latitude = 41.154264 while the 34th Raw GPS point is situated at Longitude =-8.599383 and Latitude = 41.141736. By increasing the SP distance hyperparameters to approximately 6.66km, the FMM algorithm is able to connect the 2 points to form a complete map matched route.

Figure 11 - Visualization of the complete trajectory path of *TrajectoryID 2* (from *postprocessed_output06.csv*) after applying the 2 fixes are shown above. Start Point marked green and End Point marked red. It can be noted that the trajectory produced actually described how the vehicle was going about a small area around *Bolsa Palace*(**Figure 12**), before returning to the same location.

Figure 12 - Image of *Bolsa Palace*. *Image provided by google map*

Conclusion:
Prior to the fixing task of *Task 6*, 339 trajectories were not mapped successfully. After applying the 2 fixes, there remain 23 routes still not successfully matched. For the majority of the remaining failed trajectories, it can be noted that similar problems as reported in *Cause 2* were the main causes but with much longer distance between 2 interval GPS points such that setting the SP distance hyperparameters to approximate 6.66km is still not sufficient. **Figure 13** shows one classical example where the 34th and 35th GPS points of trajectoryID 978 were situated about 11.8km apart. Therefore a search range of 6.66km is definitely not sufficient. Due to limited computing resources, further increasing the SP distance is difficult for us as this would exponentially increase the number of possible routes between pairs of points to stores. If we were to increase the SP distance from 6.66km to 15.66km, the final precomputed UBODT file turns out to be many times larger than it was after applying *Cause 2* fix. Loading the enormously sized file during map matching would crash the 8GB ram Ubuntu system we used to conduct the experiment. Therefore we did not attempt to solve the remaining trajectories.

It is also well noted that the root causes of such problems is due to the vehicle's tendency to on/off their mobile data terminal or system malfunctioning in the midst of trajectory tracking. Therefore even though we can still force an approximated route between 2 neighboring points situated far from each other using the map matching algorithm, the final map matched results may look bizarre or are very likely not exactly how the vehicle traversed. To optimally resolve the root cause, taxi drivers must ensure that the mobile data terminal is functional at all times during trajectory tracking.

```
comparison of trajectoryId:978 gps point pt34 and p35 [-8.532585, 41.137515] [-8.639334, 41.139018]
harv dist of 2 pts(m) 11871
```

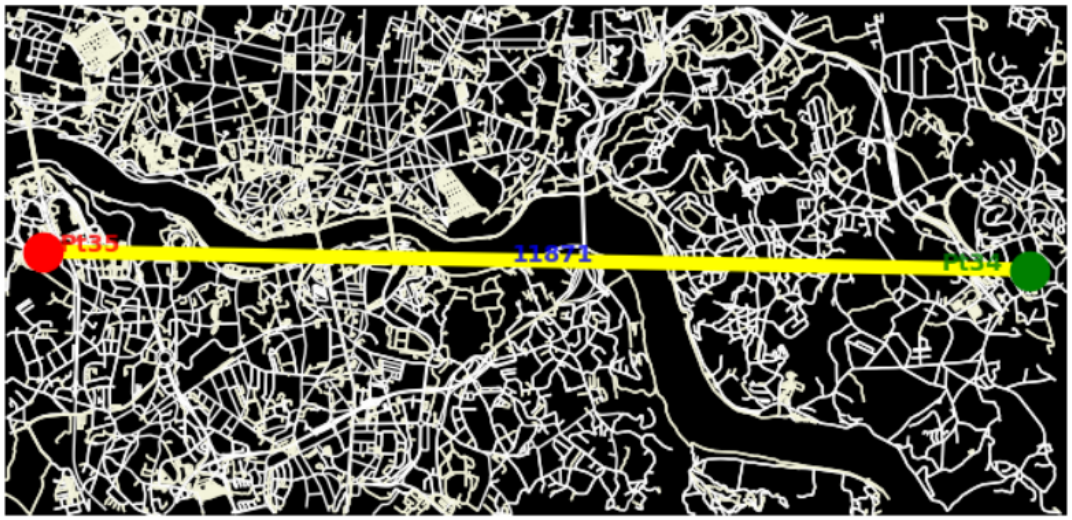

Figure 13 - 34th(Green label) and 35th(Red label) Raw GPS Points *TrajectoryID 978*. The 34th Raw GPS point is situated at Longitude=-8.532585 and Latitude = 41.137515 while the 35th Raw GPS point is situated at Longitude =-8.639334 and Latitude = 41.139018. It can be noted that the harversine distance between the 2 points are situated at about 11.8km (11871 meters) from each other.

Reference:

[1] C. Yang and G. Gidófalvi, “Fast map matching, an algorithm integrating hidden Markov model with precomputation,” *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, 2017.