# Project 2 Trajectory and Road Network Data Analysis

**Long Cheng**

**Assistant Professor**

**c.long@ntu.edu.sg**

# Project Background

Many vehicles such as taxies that are moving in an urban space have their locations sensed in real time nowadays. The sensed data corresponds to sequences of time-stamped points and is called trajectory data. Raw trajectory data usually involves noises (including sensor ones and measurement ones). As a result, the locations of a vehicle as indicated by the raw coordinates are often not on the road networks. A common practice is to map the trajectory data (of GPS points) to road networks before it is visualized and analyzed. This project is to conduct several tasks of preprocessing, visualizing and query processing trajectory data, where one important process is to map the trajectory data to a road network. The tasks are described with more details next.

# Tasks of the Project

- Task 1: Data Preparation
- Task 2: GPS Point Visualization
- Task 3: Map Matching
- Task 4: Route Visualization
- Task 5: Route Analysis
- Task 6 (bonus and optional): Case studies

# Tasks of the Project: Detailed Description

- **Task 1.** Download a road network of the Porto City, Portugal and download some trajectory data of taxies in the city from a Kaggle competition.
- **Task 2.** Visualize the raw GPS points of **the first 10 trips** in the trajectory data dataset on a map with the road network of Porto (together with different colors or separately).
- **Task 3.** Perform the map matching task to map the trajectory data to the road network using an existing open-source tool. Each trajectory will be mapped to a sequence of road segments and we call a sequence of road segments a route.
- **Task 4.** Visualize the routes that are mapped from the trajectories of the **first 10 trips** on the road network (together with different colors or separately).
- **Task 5.** Perform some analysis on the mapped routes. This includes (1) to return **5 road segments** that are traversed the most often as indicated by the trajectory data; (2) to return **5 road segments**, whose average travelling time as indicated by the trajectory data are the largest (those road segments that are traversed by no trajectories can be ignored); and (3) and visualize the road segments returned in (1) and (2) on the road network separately.
- **Task 6 (bonus and optional).** If possible, identify those cases where the map matching algorithm do not work well, think of some solutions for the identified cases and implement the new ideas into the open source tools and perform the map matching task on the trajectory data again and compare the mapped routes before and after the changes are made.

# Task 1 (Data Preparation)

- **Task 1. Download a road network of the Porto City, Portugal and download some trajectory data of taxies in the city.**
- The road network data can be downloaded from OpenStreetMap (https://www.openstreetmap.org/). Some Python code that can be used for downloading road network from OpenStreetMap can be found here: https://github.com/cyang-kth/fmm/blob/master/example/osmnx_example/download_network.ipynb
- The trajectory data can be downloaded from Kaggle (https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data) and the data description is available there as well. You need to create a Kaggle account in order to download the data. Please retrieve **the first 1000 trips in the train.csv dataset** for this project. We call the dataset **train-1000**.

# Task 2 (GPS Point Visualization)

- **Task 2. Visualize the raw GPS points of the first 10 trips in the trajectory data dataset on a map with the road network of Porto (together with different colors or separately).**
- You may use the OSMnx, which is an open-source tool (I believe this is the tool that is usable, but in case not, please find one by yourself).
  - Documentation **(on plots):** https://osmnx.readthedocs.io/en/stable/osmnx.html?highlight=ox.graph_from_place#module-osmnx.plot
  - GitHub: https://github.com/gboeing/osmnx/tree/243216b1f4574481dfe1b663d8e8dedac1acb419
- Alternatively, you can search for some other tools using Google and/or GitHub or even implement one by yourself for the task.

# Task 3 (Map Matching)

- **Task 3. Perform the map matching task to map the trajectory data to the road network using an existing open-source tool. Each trajectory will be mapped to a sequence of road segments and we call a sequence of road segment a route.**
- The open-source tool that is recommended for the map matching task can be found here.
  - https://github.com/cyang-kth/fmm
- Alternatively, you can search for some other tools using Google and/or GitHub or even implement one by yourself for the task.

# Task 4 (Route Visualization)

- **Task 4. Visualize the routes that are mapped from the trajectories of the first 10 trips on the road network (together with different colors or separately).**
- You may use the OSMnx tool as described before.
- Alternatively, you can search for some other tools using Google and/or GitHub or even implement one by yourself for the task.

# Task 5 (Route Analysis)

- **Task 5: Perform some analysis on the mapped routes. This includes (1) to return 5 road segments that are traversed the most often as indicated by the trajectory data; (2) to return 5 road segments, whose average travelling time as indicated by the trajectory data are the largest (those road segments that are traversed by no trajectories can be ignored); and (3) and visualize the road segments returned in (1) and (2) on the road network separately.**
- You need write some Python code for (1) and (2) for this task.
- For (3), you may use the OSMnx tool as described before.

# Task 6 (bonus and optional)

- **Task 6 (bonus and optional): If possible, identify those cases where the map matching algorithm do not work well, think of some solutions for the identified cases and implement the new ideas into the open source tools and perform the map matching task on the trajectory data again and compare the mapped routes before and after the changes are made.**
- If you use the map matching algorithm recommended, you may check the following paper for this task.
  - Can Yang & Gyozo Gidofalvi (2018) Fast map matching, an algorithm integrating hidden Markov model with precomputation, International Journal of Geographical Information Science, 32:3, 547-570, DOI: 10.1080/13658816.2017.1400548
- If you use other map matching algorithms, you may check their corresponding papers.

# Tasks: A Few Notes

- You can use any programming language.
- Note that the recommended open-source tools are written in Python. You are recommended to use Python for this project.

# Materials to Submit

1. A report consisting of
   - **A description of the contribution of each team member on the first page (it is assumed that each member has equal contribution if not submitted)**
   - (For Task 2) visualizations of the GPS points (first 10 trips only), together in one figure or separately in 10 figures (at most two A4 pages)
   - (For Task 4) visualizations of the mapped routes (first 10 trips only), together in one figure or separately in 10 figures (at most two A4 pages)
   - (For Task 5) visualizations of 5 road segments for 5(1) and another 5 road segments for 5(2), together in one figure or separately in 2 figures (at most one A4 page)
   - (For Task 6) anything you deem related (no page limit)
2. A CSV file storing
   - (For Task 3) the mapped routes of all 1000 trajectories in the train-1000 dataset (each row for one trip and in a format of a sequence of road segment ID's)
3. Codes for each task, in different files named based on the tasks. The codes should be executable for assessment purpose

# Submission Policy

- Format
  - Report in **PDF** format
  - Report, result file and codes put in a **zip** file
- Where to submit
  - The “Content\Project 2\Project 2 Submission” tab of NTUlearn
- **Deadline: 11:59pm 25 Nov 2022 (Fri)**
- One-week grace period for late submissions
  - No penalty if a valid excuse provided; otherwise, **a penalty of 5% per day for at most 7 days**
  - Zero mark for submissions after the grace period
- Policy on plagiarism
  - Open-source tools can be used
  - Codes cannot be shared across groups (if found, all groups that are involved will be subject to disciplinary action)
  - Discussions across groups are in general not allowed (unless pre-approval with justifications is sought first)

# Assessment Rubrics

- Task 1: 10
- Task 2: 15
- Task 3: 40
- Task 4: 15
- Task 5: 20
- Task 6 (bonus): 20
- Bounded by 100