GPS Probe Generation

This program reads a csv file which contains GPS points along the road. Sometimes, such data may be sparse and smooth visualization along the road may not be guaranteed. This program generates intermediate GPS points along the road using OpenStreetMap data. The sample csv file (gps_probe_Nepal.csv) is provided for demo purpose. The csv file assumes four fields ap_id (i.e. gPS_device ID), timestamp, lat (i.e. latitude) and lon (i.e. longitude). Based on the sequence of latitude and longitude points and the timestamps, intermediate points and corresponding timestamp are generated for visualization along the road.

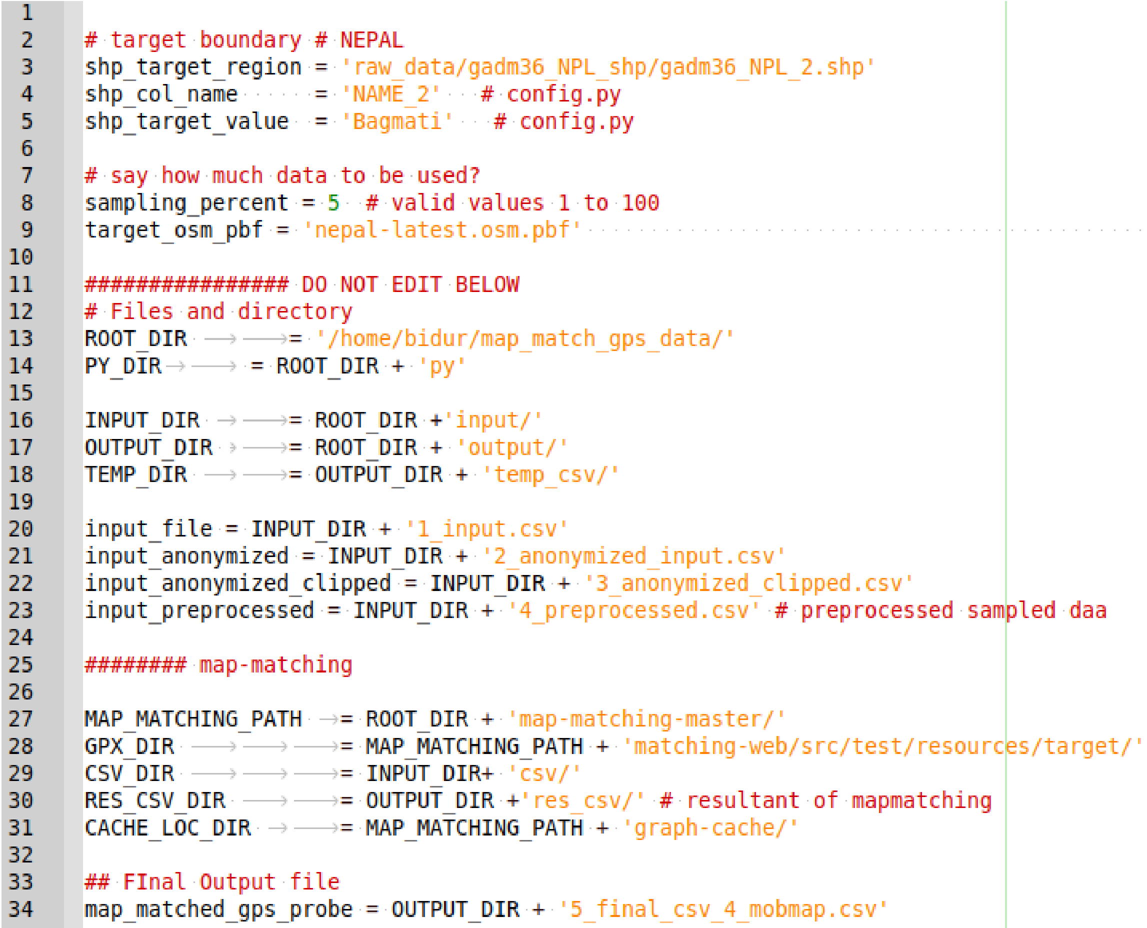
The program works for generating routes that can be visualized in Mobmap. However, this code is an initial version which have to be improved for differnt aspects. This is published here in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Main Modules used:

- Python 3 and its modules like pandas, geopandas, etc.
- Faker for Data anonymization
  - https://pypi.org/project/Faker/
- GraphHopper Map-matching tool
  - https://github.com/graphhopper/map-matching

1. Edit config.py

- Specify
  - target region’s shapefile location (line 3)
  - shapefile’s target attribute column (line 4)
  - target region’s name in shapefile (line 5)
- Specify the map of the target region obtained from openstreetmap (line 9)
  - the input GPS points in the input .csv file must be within this map region.
- Update ROOT_DIR (line 13)
  - Give the location of ‘gpsProbeMatching.ipynb’ as ROOT_DIR

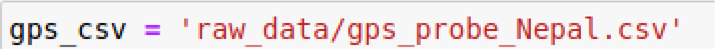


```
1
2   # target boundary # NEPAL
3   shp_target_region = 'raw_data/gadm36_NPL_shp/gadm36_NPL_2.shp'
4   shp_col_name      = 'NAME_2'   # config.py
5   shp_target_value  = 'Bagmati'   # config.py
6
7   # say how much data to be used?
8   sampling_percent = 5  # valid values 1 to 100
9   target_osm_pbf = 'nepal-latest.osm.pbf'
10
11  ################ DO NOT EDIT BELOW
12  # Files and directory
13  ROOT_DIR        = '/home/bidur/map_match_gps_data/'
14  PY_DIR          = ROOT_DIR + 'py'
15
16  INPUT_DIR       = ROOT_DIR +'input/'
17  OUTPUT_DIR      = ROOT_DIR + 'output/'
18  TEMP_DIR        = OUTPUT_DIR + 'temp_csv/'
19
20  input_file = INPUT_DIR + '1_input.csv'
21  input_anonymized = INPUT_DIR + '2_anonymized_input.csv'
22  input_anonymized_clipped = INPUT_DIR + '3_anonymized_clipped.csv'
23  input_preprocessed = INPUT_DIR + '4_preprocessed.csv' # preprocessed sampled daa
24
25  ######## map-matching
26
27  MAP_MATCHING_PATH   = ROOT_DIR + 'map-matching-master/'
28  GPX_DIR             = MAP_MATCHING_PATH + 'matching-web/src/test/resources/target/'
29  CSV_DIR             = INPUT_DIR+ 'csv/'
30  RES_CSV_DIR         = OUTPUT_DIR +'res_csv/' # resultant of mapmatching
31  CACHE_LOC_DIR       = MAP_MATCHING_PATH + 'graph-cache/'
32
33  ## FInal Output file
34  map_matched_gps_probe = OUTPUT_DIR + '5_final_csv_4_mobmap.csv'
```

Figure 1: ./py/config.py

2. Provide input csv file in gpsProbeMatching.ipynb

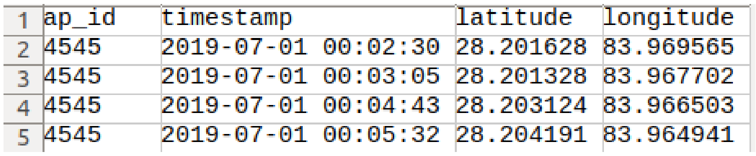
- Specify input **gps_probe** data in notebook



```
gps_csv = 'raw_data/gps_probe_Nepal.csv'
```

Figure 2: gpsProbeMatching.ipynb

- The contents of the csv file are expected as follows:(For e.g. ap_id indicates car_id)



ap_id	timestamp	latitude	longitude
4545	2019-07-01 00:02:30	28.201628	83.969565
4545	2019-07-01 00:03:05	28.201328	83.967702
4545	2019-07-01 00:04:43	28.203124	83.966503
4545	2019-07-01 00:05:32	28.204191	83.964941

Figure 3: Input File Fields

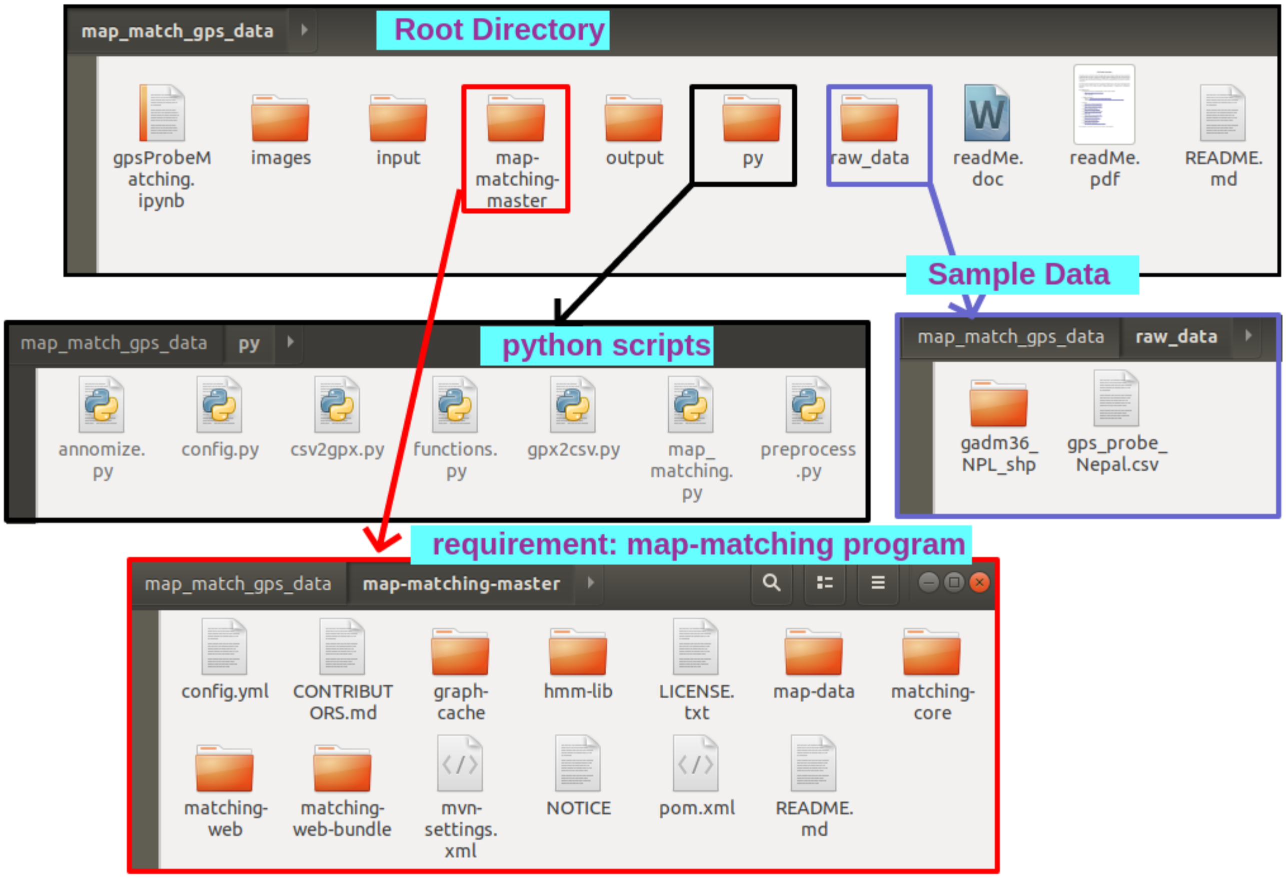


Figure 4: Directory Structure

The instructions below are to be done ONE TIME ONLY

3. Align GPS probe data along the road network (using Graphhopper map-matching)

3.1. If Java and Maven are not already installed then install them. Java 8 and Maven >=3.3.

- **JDK** (https://www.java.com/en/download/)
- **OpenJDK** (https://jdk.java.net/archive/)
- https://mkyong.com/maven/how-to-install-maven-in-windows/

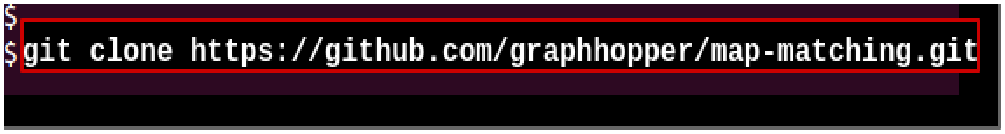
NOTE: Install JDK or OpenJDK, any of them works.

3.2. Download and install GraphHopper Map-matching tool

Download map-matching to **map_match_gps_data** directory (with option 1 **OR** option 2)

- **Option 1:** Download: https://github.com/graphhopper/map-matching and copy the code in the appropriate directory as shown in the image in *Figure 4* above.

- **Option 2:** Use git clone comman:
  - Change directory to <map_match_gps_data>
  - git clone https://github.com/graphhopper/map-matching.git



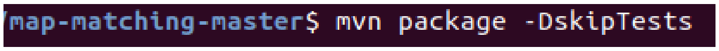
```
$
$git clone https://github.com/graphhopper/map-matching.git
```

Fig 5: Git clone command to download map-matching

Please confirm that **map-matching-master** is inside **map_match_gps_data** as shown in the *Figure 4* above.

3.2 Build (install) Graphhopper

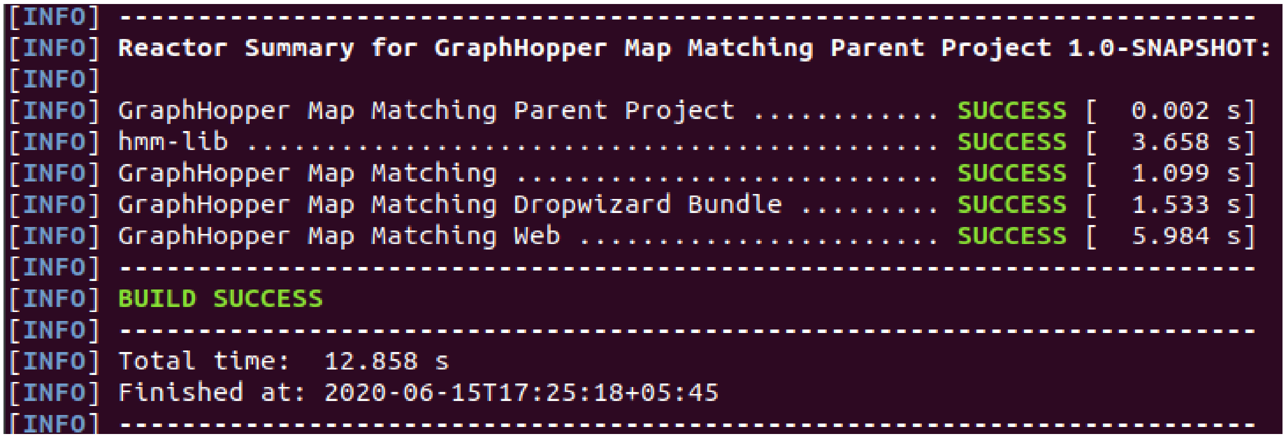
- Change directory to <map-matching-master>
- Command: `mvn package -DskipTests`



```
/map-matching-master$ mvn package -DskipTests
```

Fig 6: Build Command

The build command executes for some time and the successful completion is displayed as follows:

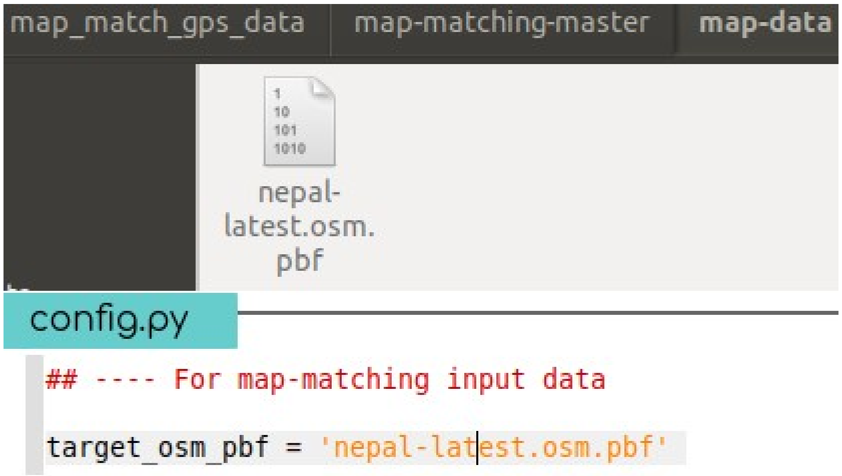


```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for GraphHopper Map Matching Parent Project 1.0-SNAPSHOT:
[INFO]
[INFO] GraphHopper Map Matching Parent Project ............ SUCCESS [  0.002 s]
[INFO] hmm-lib ............................................ SUCCESS [  3.658 s]
[INFO] GraphHopper Map Matching ........................... SUCCESS [  1.099 s]
[INFO] GraphHopper Map Matching Dropwizard Bundle ......... SUCCESS [  1.533 s]
[INFO] GraphHopper Map Matching Web ....................... SUCCESS [  5.984 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  12.858 s
[INFO] Finished at: 2020-06-15T17:25:18+05:45
[INFO] ------------------------------------------------------------------------
```

Fig 7: Successful Built

3.3 Import OSM map for target geography (e.g. Nepal)

- Download: https://download.geofabrik.de/asia/nepal-latest.osm.pbf
- Copy to map-data directory.
- Update config.py with the name of the map file **target_osm_pbf = nepal-latest.osm.pbf**
- NOTE: make sure the GPS Probe is within the map region, otherwise map-matching may encounter exceptions.



```
## ---- For map-matching input data

target_osm_pbf = 'nepal-latest.osm.pbf'
```

Fig 8: Download and Save nepal-latest.osm.pbf

4. Run notebook code in browser <gpsProbeMatching.ipynb>

4.3 Generated Routes:

- The **output/** directory will contain the output files.
- *‘5_final_csv_4_mobmap.csv’* contains the generated routes for all ap_ids combined in a single file.
- MobMap visualization can be done using this file
- res_csv/ contains generated route for individual ap_ids separately.

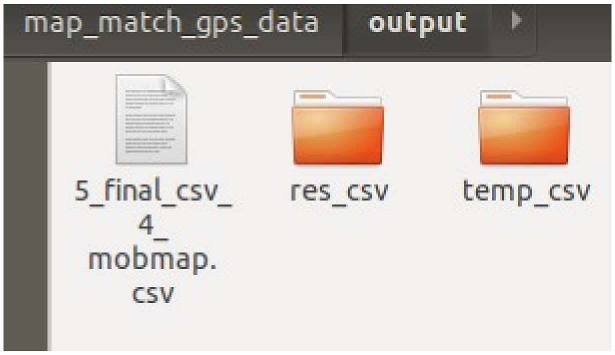


Figure 9: Output Files

4.4 Input and Intermediate Files:

- The input/ directory contains original as well as intermediate probe data.
  - 1_input.csv : original input file.
  - 2_anonymized_input.csv : ap_id field is anonymized
  - 3_anonymized_clipped.csv: Input data which lies beyond the desired geographic boundary is removed.
  - 4_preprocessed.csv: Preprocessed file (remove duplicates, corrected same timestamp and multiple location issue) for all the data.
  - csv/ folder : input separated into individual .csv file for each ap_id (i.e. each car)

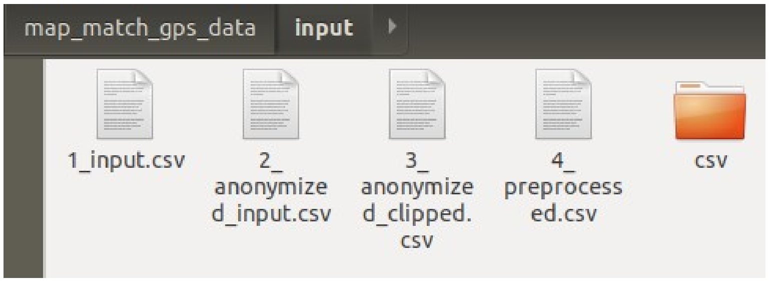
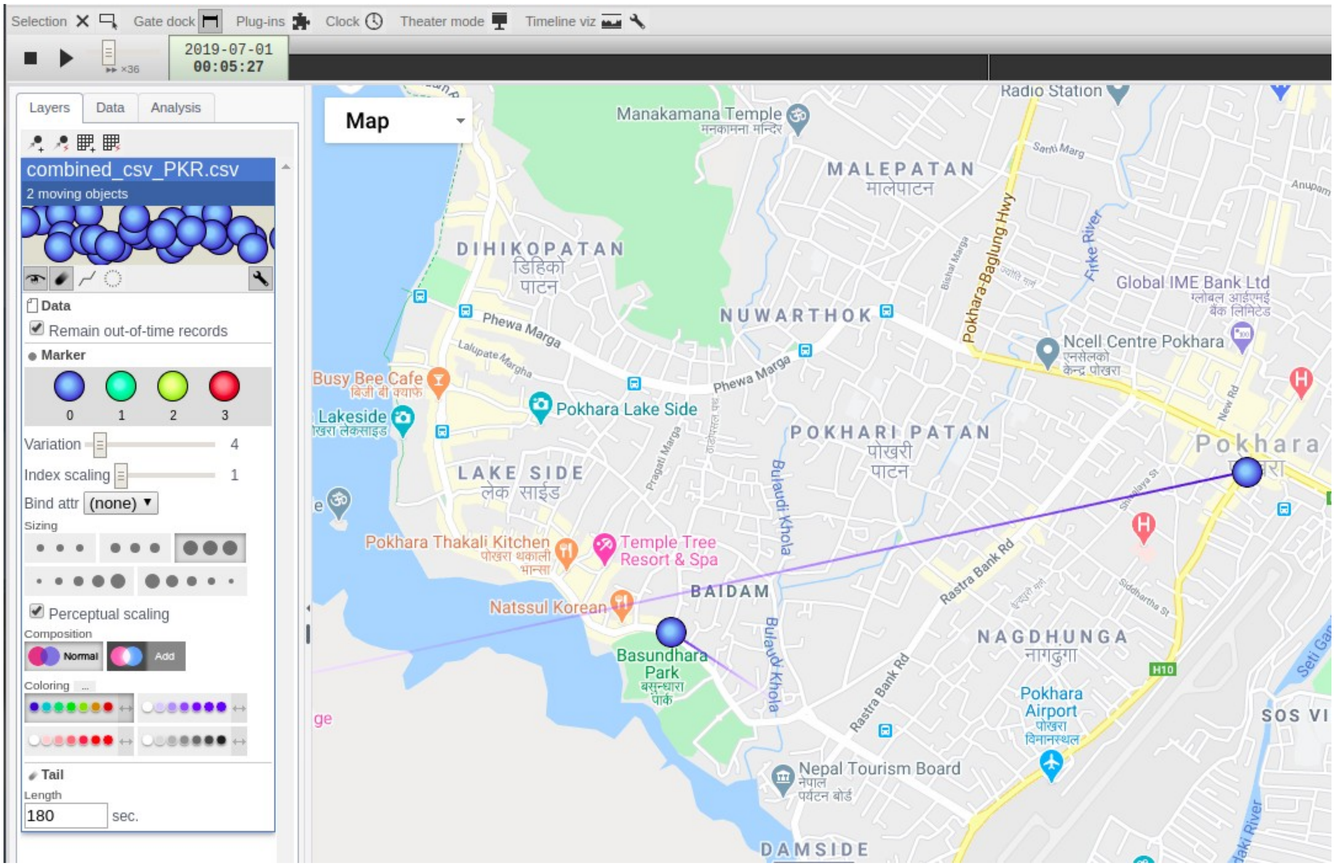


Figure 10: Input Folder with Intermediate files

4.5 MobMap visualization <https://shiba.iis.u-tokyo.ac.jp/member/ueyama/mm/app/>



2019-07-01
00:05:27

combined_csv_PKR.csv
2 moving objects

Data
☑ Remain out-of-time records

Variation 4
Index scaling 1
Bind attr (none) ▾

☑ Perceptual scaling

Length
180 sec.

Figure 11: Visualization in MobMap