Replication package

Manuscript title: *On the simulation of shared autonomous micro-mobility*

Manuscript ID: COMMTR-D-22-00020R1

Authors: Naroa Coretti Sanchez [*,a], Iñigo Martinez[b], Luis Alonso[a], Kent Larson[a]

[a] *MIT Media Lab, Cambridge, MA, USA*

[b] *University of Navarra, San Sebastian, Spain*

* Corresponding author, *email address:* naroa@mit.edu

Replication package access:

The source code for the simulation tool presented in this paper is available on GitHub https://github.com/CityScope/AutonomousMicroMobility

In addition, we have also put together a website with a comprehensive guide on how to use it https://micro-mobility-abm.netlify.app/

Data description:

The data used for the results shown in Section 6 is available in the same GitHub repository, under the data folder: https://github.com/NaroaCS/AutonomousBicycleSimulation/tree/master/data

This data includes:

1. The user demand 'user_trips_0.csv' https://github.com/NaroaCS/AutonomousBicycleSimulation/blob/master/data/user_trips_0.csv
2. The station data:

   https://github.com/NaroaCS/AutonomousBicycleSimulation/blob/master/data/bluebikes_stations_07_2020.csv

3. The GIS data:

   3.1 Shapefile containing the buildings 'buildings.shp':

   https://github.com/NaroaCS/AutonomousBicycleSimulation/blob/master/data/buildings/buildings.shp

   3.2 Graph containing the roads 'greater_boston_road.graphml':

   https://github.com/NaroaCS/AutonomousBicycleSimulation/blob/master/data/graph/greater_boston_road.graphml

4. The output of the demand prediction module 'demand_grid.csv'

   https://github.com/NaroaCS/AutonomousBicycleSimulation/blob/master/data/demand_grid.csv

The dataset that was used for training the demand prediction module can be obtained directly from the Bluebikes website, going to https://www.bluebikes.com/system-data and then clicking on 'Download Bluebikes trip history data' or directly in https://s3.amazonaws.com/hubway-data/index.html

Code description:

The environment needed to run the scripts is Python with a version greater than 3.6 and the following packages:

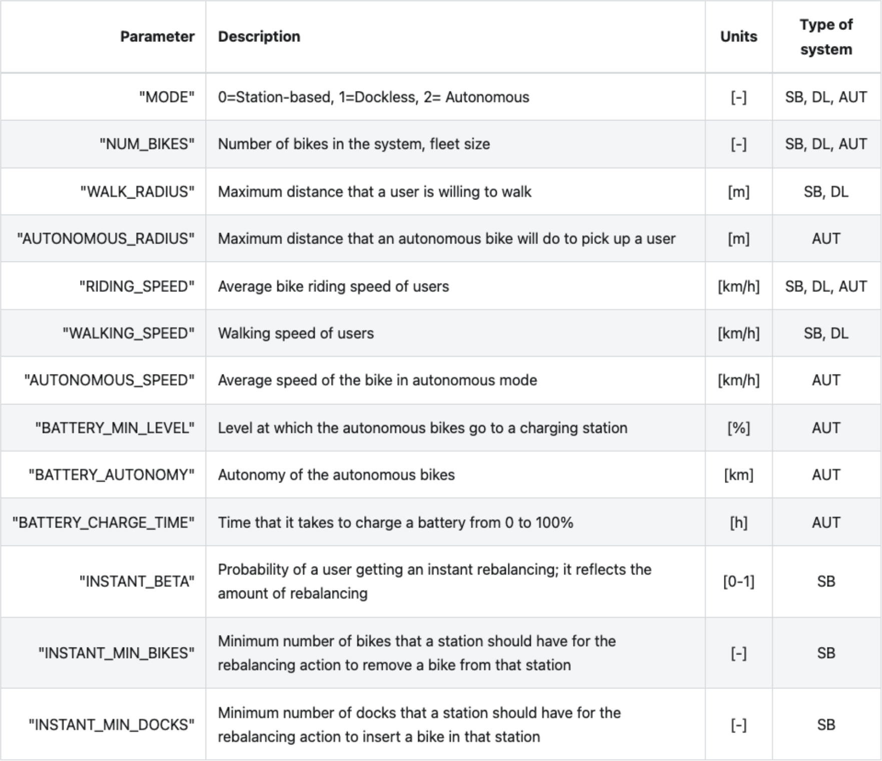
- simpy
- pandas
- numpy
- matplotlib
- networkx
- scipy==1.6.0
- geopandas
- scikit-learn
- pyproj
- tqdm
- tensorflow==1.15
- git+git://github.com/imartinezl/pandana.git@master

Simulation software description:

1. Clone the git repository:

   git clone https://github.com/NaroaCS/AutonomousBicycleSimulation.git

2. You can then customize the inputs as desired:
   a. To change the city you will need to obtain a shapefile containing the buildings (https://osmbuildings.org/) and a graph containing the road network (https://overpass-turbo.eu/).
   b. You can customize the parameters in the configuration file ‘config.json’

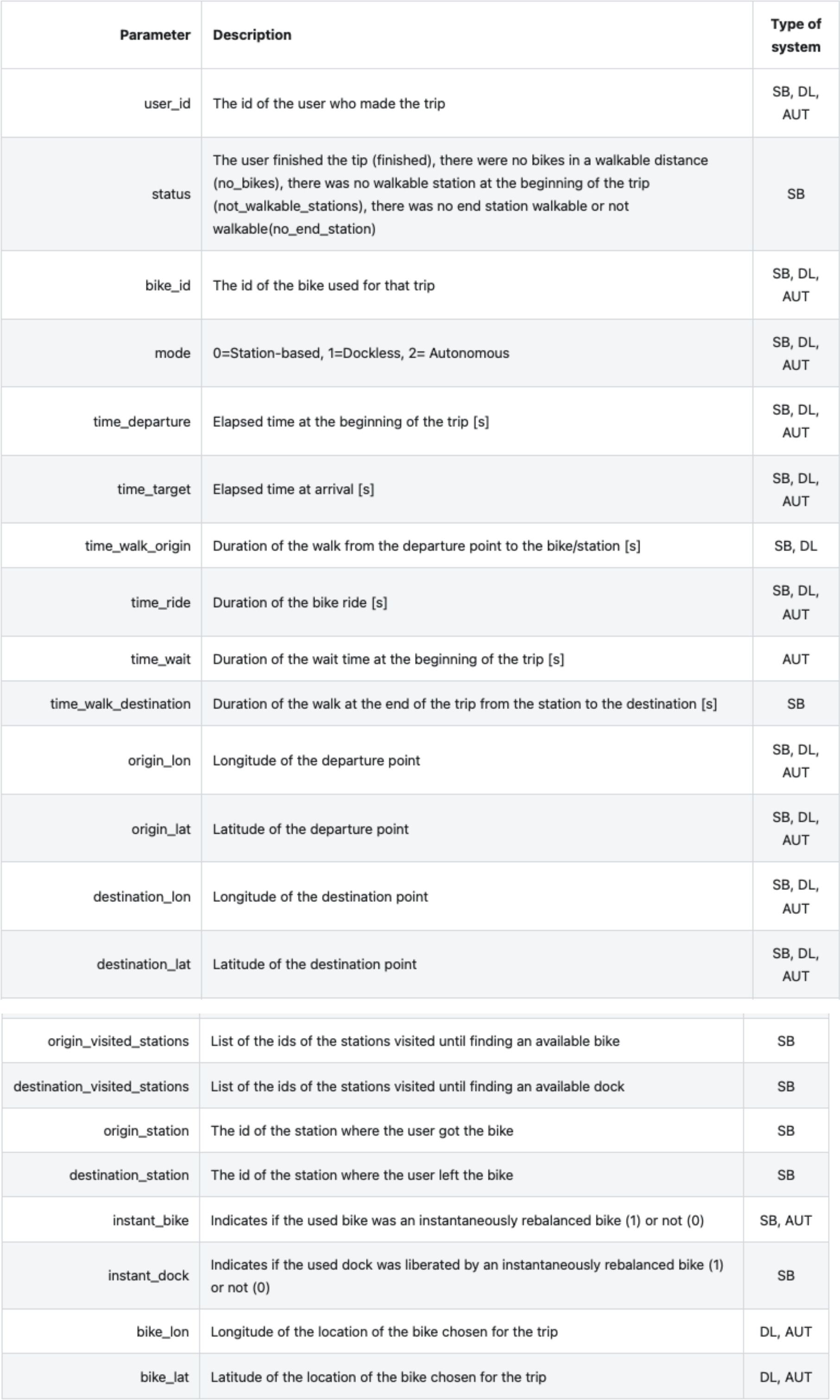


Parameter	Description	Units	Type of system
"MODE"	0=Station-based, 1=Dockless, 2= Autonomous	[-]	SB, DL, AUT
"NUM_BIKES"	Number of bikes in the system, fleet size	[-]	SB, DL, AUT
"WALK_RADIUS"	Maximum distance that a user is willing to walk	[m]	SB, DL
"AUTONOMOUS_RADIUS"	Maximum distance that an autonomous bike will do to pick up a user	[m]	AUT
"RIDING_SPEED"	Average bike riding speed of users	[km/h]	SB, DL, AUT
"WALKING_SPEED"	Walking speed of users	[km/h]	SB, DL
"AUTONOMOUS_SPEED"	Average speed of the bike in autonomous mode	[km/h]	AUT
"BATTERY_MIN_LEVEL"	Level at which the autonomous bikes go to a charging station	[%]	AUT
"BATTERY_AUTONOMY"	Autonomy of the autonomous bikes	[km]	AUT
"BATTERY_CHARGE_TIME"	Time that it takes to charge a battery from 0 to 100%	[h]	AUT
"INSTANT_BETA"	Probability of a user getting an instant rebalancing; it reflects the amount of rebalancing	[0-1]	SB
"INSTANT_MIN_BIKES"	Minimum number of bikes that a station should have for the rebalancing action to remove a bike from that station	[-]	SB
"INSTANT_MIN_DOCKS"	Minimum number of docks that a station should have for the rebalancing action to insert a bike in that station	[-]	SB

    c. Update the file names accordingly and run UserGeneration.py to generate the new demand files

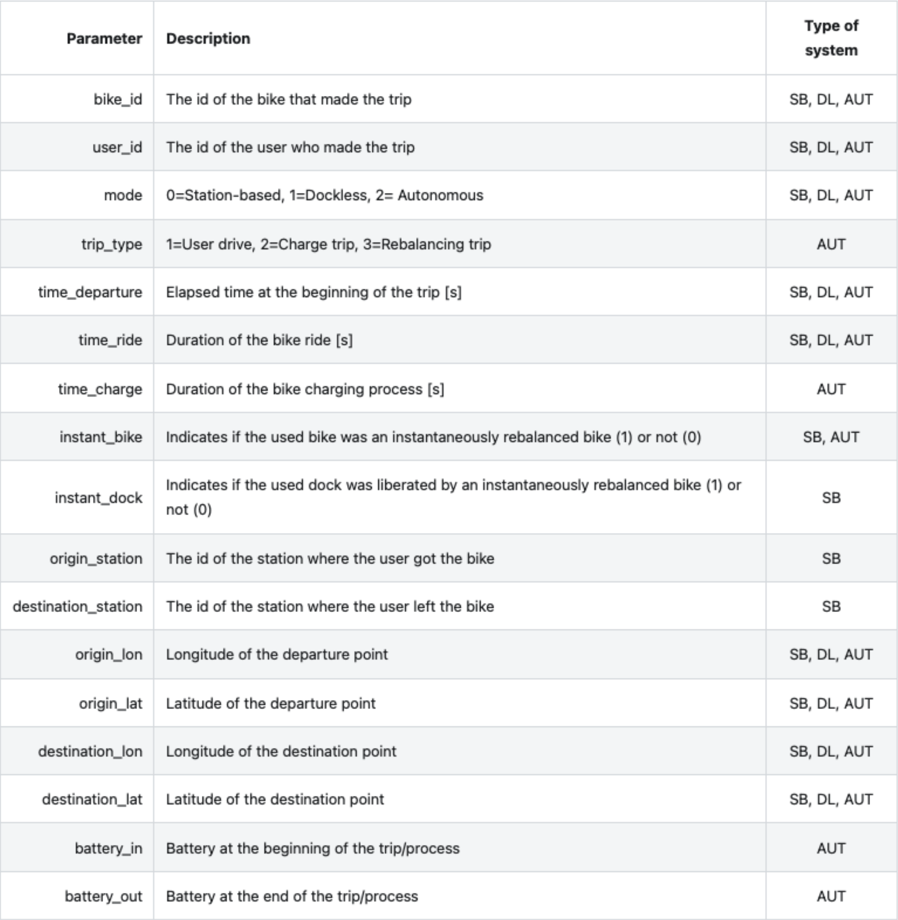
3. Run main.py
4. The run times will be printed at the end of the simulation and the results will be saved in the ‘results’ folder in a subfolder with the filename being the timestamp of the simulation launch time. This folder will contain the configuration file that was used to launch it and the two main output files: ‘user_trips.csv’ and ‘bike_trips.csv’.

The output file ‘user_trips.csv’ has the following columns:



Parameter	Description	Type of system
user_id	The id of the user who made the trip	SB, DL, AUT
status	The user finished the tip (finished), there were no bikes in a walkable distance (no_bikes), there was no walkable station at the beginning of the trip (not_walkable_stations), there was no end station walkable or not walkable(no_end_station)	SB
bike_id	The id of the bike used for that trip	SB, DL, AUT
mode	0=Station-based, 1=Dockless, 2= Autonomous	SB, DL, AUT
time_departure	Elapsed time at the beginning of the trip [s]	SB, DL, AUT
time_target	Elapsed time at arrival [s]	SB, DL, AUT
time_walk_origin	Duration of the walk from the departure point to the bike/station [s]	SB, DL
time_ride	Duration of the bike ride [s]	SB, DL, AUT
time_wait	Duration of the wait time at the beginning of the trip [s]	AUT
time_walk_destination	Duration of the walk at the end of the trip from the station to the destination [s]	SB
origin_lon	Longitude of the departure point	SB, DL, AUT
origin_lat	Latitude of the departure point	SB, DL, AUT
destination_lon	Longitude of the destination point	SB, DL, AUT
destination_lat	Latitude of the destination point	SB, DL, AUT
origin_visited_stations	List of the ids of the stations visited until finding an available bike	SB
destination_visited_stations	List of the ids of the stations visited until finding an available dock	SB
origin_station	The id of the station where the user got the bike	SB
destination_station	The id of the station where the user left the bike	SB
instant_bike	Indicates if the used bike was an instantaneously rebalanced bike (1) or not (0)	SB, AUT
instant_dock	Indicates if the used dock was liberated by an instantaneously rebalanced bike (1) or not (0)	SB
bike_lon	Longitude of the location of the bike chosen for the trip	DL, AUT
bike_lat	Latitude of the location of the bike chosen for the trip	DL, AUT

The ‘bike_trips.csv’ has the following columns:



Parameter	Description	Type of system
bike_id	The id of the bike that made the trip	SB, DL, AUT
user_id	The id of the user who made the trip	SB, DL, AUT
mode	0=Station-based, 1=Dockless, 2= Autonomous	SB, DL, AUT
trip_type	1=User drive, 2=Charge trip, 3=Rebalancing trip	AUT
time_departure	Elapsed time at the beginning of the trip [s]	SB, DL, AUT
time_ride	Duration of the bike ride [s]	SB, DL, AUT
time_charge	Duration of the bike charging process [s]	AUT
instant_bike	Indicates if the used bike was an instantaneously rebalanced bike (1) or not (0)	SB, AUT
instant_dock	Indicates if the used dock was liberated by an instantaneously rebalanced bike (1) or not (0)	SB
origin_station	The id of the station where the user got the bike	SB
destination_station	The id of the station where the user left the bike	SB
origin_lon	Longitude of the departure point	SB, DL, AUT
origin_lat	Latitude of the departure point	SB, DL, AUT
destination_lon	Longitude of the destination point	SB, DL, AUT
destination_lat	Latitude of the destination point	SB, DL, AUT
battery_in	Battery at the beginning of the trip/process	AUT
battery_out	Battery at the end of the trip/process	AUT

5. Lastly, while the results of the demand prediction for the case presented in this article have been provided as data (demand_grid.csv), we also detail here the procedure that was followed to generate such files:
    1. Download the historical trip data from Buebikes (see Data section in this replication package) and save it in a folder ‘bluebikes_data’ within the Preprocessing folder:
    2. Rscript training_data.R (input bluebikes_data → outputs training_data.csv)
    3. python3 gccn_ddgf.py (input training_data.csv → outputs testing_data.csv)
    4. Rscript testing_data.R (input testing_data.csv → outputs demand_grid.csv)