
osm2gmns

Release 0.7.3

Jiawei Lu, Xuesong (Simon) Zhou

Jul 19, 2023

CONTENTS

1	Publication	3
2	Main Features	5
3	Contents	7
3.1	Installation	7
3.2	GMNS	7
3.3	Multi-Resolution Modeling	8
3.4	Quick Start	13
3.5	Functions	22
3.6	Sample Networks	27
3.7	Acknowledgement	28
	Index	31

Authors: Jiawei Lu, Xuesong (Simon) Zhou

Email: jiawei19@asu.edu, xzhou74@asu.edu

osm2gmns is an open-source Python package that enables users to conveniently obtain and manipulate any networks from [OpenStreetMap](#) (OSM). With a single line of Python code, users can obtain and model drivable, bikeable, walkable, railway, and aeroway networks for any region in the world and output networks to CSV files in [GMNS](#) format for seamless data sharing and research collaboration. osm2gmns mainly focuses on providing researchers and practitioners with flexible, standard and ready-to-use multi-modal transportation networks, as well as a bunch of customized and practical functions to facilitate various research and applications on traffic modeling.

PUBLICATION

Lu, J., & Zhou, X.S. (2023). Virtual track networks: A hierarchical modeling framework and open-source tools for simplified and efficient connected and automated mobility (CAM) system design based on general modeling network specification (GMNS). *Transportation Research Part C: Emerging Technologies*, 153, 104223. [paper link](#)

MAIN FEATURES

- Obtain any networks from OSM. osm2gmns parses map data from OSM and output networks to csv files in GMNS format.
- Standard network format. osm2gmns adopts GMNS as the network format for seamless data sharing and research collaboration.
- Ready-to-use network. osm2gmns cleans erroneous information from OSM map data and is able to fill up critical missing values, e.g., lanes, speed and capacity, to quickly provide ready-to-use networks.
- Directed network. two directed links are generated for each bi-directional osm ways identified by osm2gmns.
- Multi-modal support. five different network types are supported, including auto, bike, walk, railway, and aeroway
- Customized and practical functions to facilitate traffic modeling. functions include complex intersection consolidation, movement generation, traffic zone creation, short link combination, network visualization.
- Multi-resolution modeling. osm2gmns automatically constructs the corresponding mesoscopic and microscopic networks for any macroscopic networks in GMNS format.

CONTENTS

3.1 Installation

You can install the latest release of `osm2gmns` at [PyPI](#) via `pip`:

```
pip install osm2gmns
```

By running the command above, the `osm2gmns` package along with three required dependency packages ([Shapely](#), [osmium](#), and [numpy](#)) will be installed to your computer (if they have not been installed yet).

3.1.1 Potential Issues

- [Shapely](#)

If you install `osm2gmns` in a conda environment, you may get an error message: “`OSError: [WinError 126] The specified module could not be found`” when importing `osm2gmns`. To resolve this issue, you need to uninstall the [Shapely](#) package first, and reinstall it manually using the command below.

```
conda install shapely
```

- [osmium](#)

Windows users may get an error message related to [osmium](#) (one of the dependency packages of `osm2gmns`) when installing or using `osm2gmns` with Python version > 3.8. The reason is the highest Python version that `osmium` supports on PyPI is Py3.8 for Windows.

Affected users can download binary wheels of `osmium` from [our repository](#) or [osmium github homepage](#) and use `pip` to install the wheel file that matches your Python version.

3.2 GMNS

General Modeling Network Specification ([GMNS](#)), proposed by the [Zephyr Foundation](#), defines a common human and machine readable format for sharing routable road network files. It is designed to be used in multi-modal static and dynamic transportation planning and operations models. It will facilitate the sharing of tools and data sources by modelers. For additional information on GMNS goals, history and requirements, please see the [wiki](#).

GMNS (version 0.92) includes the following features for use in static models:

- Configuration information and use definitions.
- Node and link files, to establish a routable network.

For dynamic models, GMNS (version 0.92) includes the following optional additional features:

- A segment file, with information that overrides the characteristics of a portion of a link.
- A lane file that allocates portions of the right-of-way. Lanes include travel lanes used by motor vehicles. They may also optionally include bike lanes, parking lanes, and shoulders.
- A segment_lane file that specifies additional lanes, dropped lanes, or changes to lane properties on a segment of a link.
- A movement file that specifies how inbound and outbound lanes connect at an intersection.
- Link, segment, lane and movement time-of-day (TOD) files, that allocate usage of network elements by time-of-day and day-of-week.
- Signal phase and timing files, for basic implementation of traffic signals.

osm2gmns uses GMNS as the standard when processing and manipulating networks, and thus any network in GMNS format is fully compatible with osm2gmns.

3.3 Multi-Resolution Modeling

Multi-Resolution Modeling (MRM) is a modeling technology that creates a family of models that represent the same phenomenon or a set of questions at more than two different resolutions. The fine-grained spatial scales could cover corridors, roads, and lane representations, and the temporal resolution refers to the time interval (or time stamps) at which the dynamic state of the model is updated, typically ranging from days to seconds. Each type of model (macroscopic, mesoscopic, or microscopic) has its own advantages and disadvantages, and represents a trade-off between scales and resolution levels. The ultimate goal of MRM is to seamlessly integrate models with different temporal and spatial resolutions, while the focus of the cross-resolution approach is to bridge the gaps between macroscopic and microscopic levels, so as to provide strong theoretical support and deeper insights for both levels.

The osm2gmns package adopts the [GMNS](#) standard and further extends it to a inherently consistent multi-resolution network modeling standard. With a single line of code, osm2gmns can help users generate corresponding mesoscopic and microscopic networks for any given macroscopic networks in GMNS format, enabling practitioners and researchers to carry out various studies on transportation planning, designing, optimization, simulation, and computation under different spatial granularities. In this section, we mainly talk about the three levels of transportation network representation.

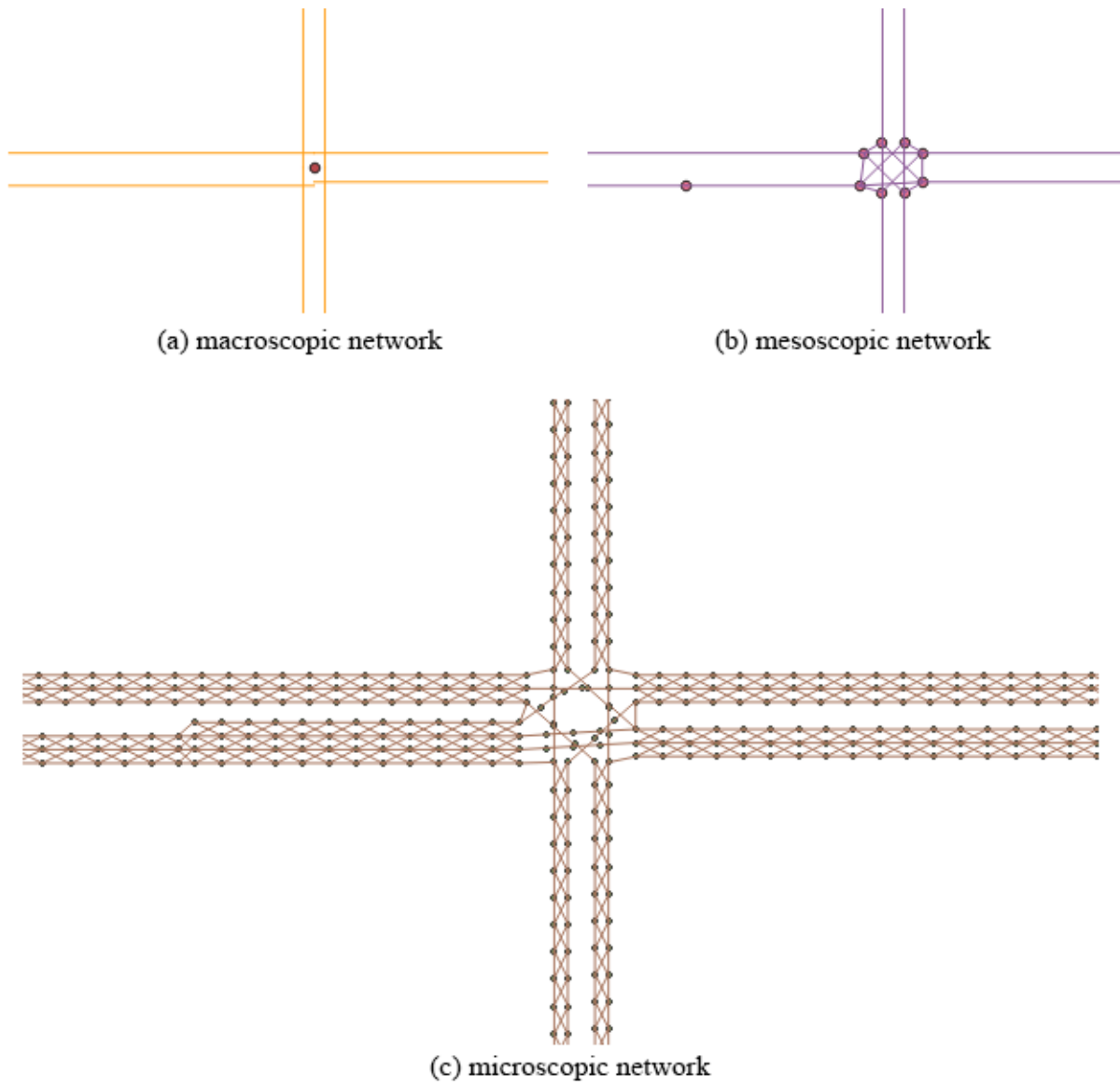


Fig. 1: Multi-resolution network representation

3.3.1 Macroscopic Network

- node.csv

Field	Type	Re- quired?	Comments
name	string		
node_id	int	yes	unique key
osm_node_id	string or int		corresponding node id in osm data
osm_highway	string		point type in osm data
zone_id	int		
ctrl_type	enum		signal; null
node_type	string		
activity_type	string		defined by adjacent links
is_boundary	enum		-1: boundary node only with incoming links; 0: no; 1: boundary node only with outgoing links; 2: boundary node with both incoming and outgoing links
x_coord	double	yes	WGS 84 is used in osm
y_coord	double	yes	WGS 84 is used in osm
intersection_id	int		nodes belonging to one complex intersection have the same id
poi_id	int		id of the corresponding poi
notes	string		

- link.csv

A link is an edge in a network, defined by the nodes it travels from and to. It may have associated geometry information^[2]. Similar to node.csv, We also added several new attributes to the link file. Detailed link data dictionary is listed below.

Field	Type	Required?	Comments
name	string		
link_id	int	yes	unique key
osm_way_id	string or int		corresponding way id in osm data
from_node_id	int	yes	
to_node_id	int	yes	
dir_flag	enum		1: forward, -1: backward, 0:bidirectional
length	float		unit: meter
lanes	int		
free_speed	float		unit: kilometer/hour
capacity	float		unit: veh/hr/lane
link_type_name	string		
link_type	int		
geometry	Geometry		wkt
allowed_uses	enum		auto, bike, walk
from_biway	bool		1: link created from a bidirectional way, 0: not
is_link	bool		1: link connecting two roads, 0: not

There are two optional files including `movement.csv` and `segment.csv` that follow the exact same format as defined

in the GMMS standard. Readers can check the GMNS website for details.

In addition to the above files defined in the GMNS standard, osm2gmns can also produce `poi.csv` files where point of interest information is stored. Detailed poi data dictionary is listed below.

Field	Type	Required?	Comments
name	string		
poi_id	int	yes	unique key
osm_way_id	string or int		corresponding way id in osm data
osm_relation_id	string or int		corresponding relation id in osm data
building	string		building tag in osm data
amenity	string		amenity tag in osm data
way	string		way tag in osm data
geometry	Geometry	yes	wkt
centroid	Geometry		wkt
area	float		area of the poi. unit: square meter
area_ft2	float		area of the poi. unit: square feet

3.3.2 Mesoscopic Network

Compared to the original macroscopic network, the mesoscopic network has more detailed information of the intersections. In the mesoscopic network, the research team expanded each intersection represented by a node in the macroscopic network. The team built a connector link for each intersection movement to facilitate intersection modeling, especially for signalized intersections.

Macroscopic and mesoscopic networks have different link-level coding schemes. Macroscopic networks often represent a road segment between two adjacent intersections as a link; however, lane changes sometimes occur within a link, especially when close to intersections. Changes in the number of lanes result in capacity changes, but the link attributes cannot properly reflect these changes. This situation may bring inconvenience or even potential errors when performing network modeling. In the GMNS standard, the comma-separated values (CSV) file, `segment.csv`, stores lane changes. The research team split and converted each link with lane changes from a macroscopic network to multiple mesoscopic links so that each mesoscopic link has a homogeneous capacity.

- `node.csv`

Field	Type	Re-quired?	Comments
node_id	int	yes	unique key
zone_id	int		
x_coord	double	yes	WGS 84 is used in osm
y_coord	double	yes	WGS 84 is used in osm
macro_node_id	int		id of its parent macroscopic node
macro_link_id	int		id of its parent macroscopic link
activity_type	string		
is_boundary	enum		-1: boundary node only with incoming links; 0: no; 1: boundary node only with outgoing links

- `link.csv`

A link is an edge in a network, defined by the nodes it travels from and to. It may have associated geometry

information^[2]. Similar to node.csv, We also added several new attributes to the link file. Detailed link data dictionary is listed below.

Field	Type	Re-quired?	Comments
link_id	int	yes	unique key
from_node_id	int	yes	
to_node_id	int	yes	
dir_flag	enum		1: forward, -1: backward, 0:bidirectional
length	float		unit: meter
lanes	int		
free_speed	float		unit: kilometer/hour
capacity	float		unit: veh/hr/lane
link_type_name	string		
link_type	int		
geometry	Geom-etry		wkt
macro_node_id	int		id of its parent macroscopic node
macro_link_id	int		id of its parent macroscopic link
mvmt_txt_id	enum		NBL, NBT, NBR, NBU, SBL, SBT, SBR, SBU, EBL, EBT, EBR, EBU, WBL, WBT, WBR, WBU
allowed_uses	enum		auto, bike, walk

3.3.3 Microscopic Network

In the Maryland case study, microscopic networks used a lane-by-lane, cell-based representation. Instead of a conceptual line segment, lanes represented each link. The research team further discretized lanes into small cells to accurately describe vehicle motion status when moving on the road. The team also created changing cells to enable vehicles to switch trajectories between lanes. Users can customize the length of cells to accommodate different modeling needs.

- node.csv

Field	Type	Re-quired?	Comments
node_id	int	yes	unique key
zone_id	int		
x_coord	double	yes	WGS 84 is used in osm
y_coord	double	yes	WGS 84 is used in osm
meso_link_id	int		id of its parent mesoscopic link
lane_no	int		start from 1 from inner side to outer side
is_boundary	enum		-1: boundary node only with incoming links; 0: no; 1: boundary node only with outgoing links

- link.csv

A link is an edge in a network, defined by the nodes it travels from and to. It may have associated geometry information^[2]. Similar to node.csv, We also added several new attributes to the link file. Detailed link data dictionary is listed below.

Field	Type	Re- quired?	Comments
link_id	int	yes	unique key
from_node_id	int	yes	
to_node_id	int	yes	
dir_flag	enum		1: forward, -1: backward, 0:bidirectional
length	float		unit: meter
lanes	int		
free_speed	float		unit: kilometer/hour
capacity	float		unit: veh/hr/lane
link_type_name	string		
link_type	int		
geometry	Geom- etry		wkt
macro_node_id	int		id of its parent macroscopic node
macro_link_id	int		id of its parent macroscopic link
meso_link_id	int		id of its parent mesoscopic link
cell_type	enum		1: traveling cell, 2: lane changing cell
addi- tional_cost	float		
lane_no	int		start from 1 from inner side to outer side
mvmt_txt_id	enum		NBL, NBT, NBR, NBU, SBL, SBT, SBR, SBU, EBL, EBT, EBR, EBU, WBL, WBT, WBR, WBU
al- lowed_uses	enum		auto, bike, walk

^[1] <https://github.com/zephyr-data-specs/GMNS/blob/master/Specification/Node.md>

^[2] <https://github.com/zephyr-data-specs/GMNS/blob/master/Specification/Link.md>

3.4 Quick Start

In this section, some examples are provided to quickly show how to use osm2gmns to generate, manipulate and output networks.

3.4.1 Download OSM Data

To reduce uncertainties while directly parsing network data from the osm server via APIs, osm2gmns uses downloaded osm files to extract useful network information. As a result, the first step is preparing osm files.

Thanks to the open-source nature of OpenStreetMap, there are lots of APIs and mirror sites that we can use to download osm map data. We list several popular sites here for users to choose.

1) OpenStreetMap Homepage

On OpenStreetMap [homepage](#), click the **Export** button to enter Export mode. Before downloading, you may need to span and zoom in/out the map to make sure that your target area is properly shown on the screen. Or, you can use **Manually select a different area** to select your area more precisely. Click the **Export** button in blue to export the network you want.

Note that if the target area is too large, you may get an error message: “You requested too many nodes (limit is 50000). Either request a smaller area, or use planet.osm”. In this case, you can always click **Overpass API** to download the

network you need via a mirror site.

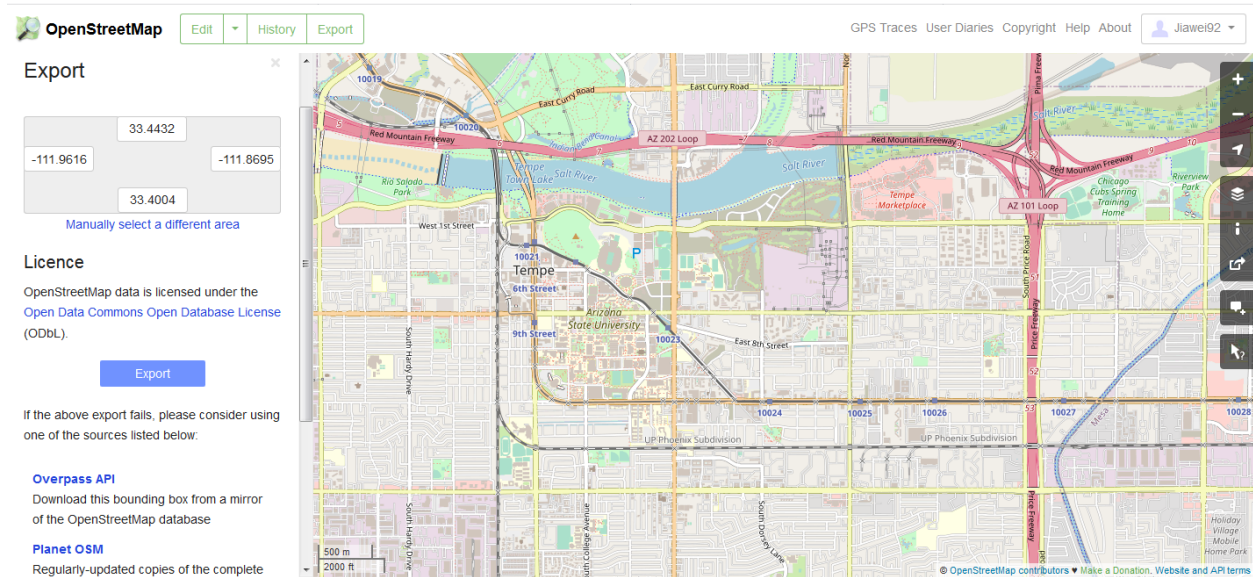


Fig. 2: Download osm data from OpenStreetMap homepage

2) Geofabrik

Different from the way of downloading map data from OpenStreetMap homepage, [Geofabrik](#) enables you to download network data for administrative areas. On OpenStreetMap homepage, we can only download areas defined by rectangles. In Geofabrik, you can click the corresponding quick link of your interested region to download the map data you need. You can always click the name of regions to check if sub region data are available.

Generally, there are three types of file format for users to choose when downloading map data. osm2gmns supports .pbf and .osm files. In osm2gmns, networks stored in .osm files are parsed more quickly than those stored in .pbf files. However, compared with .pbf files, .osm files take much more hard disk space to store networks and much more space in RAM while parsing.

3) BBBike

If your target area is neither an administrative region nor a rectangle, [BBBike](#) may be a good choice. [BBBike](#) enables you to select your region using a polygon. [BBBike](#) supports numerous file formats to output and store network data. Users can select a proper one according to their requirements.

Note:

- The file formats of map data supported in osm2gmns include .osm, .xml, and .pbf.

4) Overpass API

osm2gmns also enables users to download OSM data within the region of interest using a built-in function. A region can be a state, city, or even university. On OpenStreetMap [homepage](#), search the region name to get its unique relation id. The following example shows how to download Tempe city OSM data using function `downloadOSMData`.

```
>>> import osm2gmns as og
>>> og.downloadOSMData(110833, 'tempe.osm')
```

OpenStreetMap Data Extracts

The OpenStreetMap data files provided on this server do **not** contain the user names, user IDs and changeset IDs of the OSM objects because these fields are assumed to contain personal information about the OpenStreetMap contributors and are therefore subject to data protection regulations in the European Union.

[Extracts with full metadata](#) are available to OpenStreetMap contributors only.

Welcome to Geofabrik's free download server. This server has data extracts from the [OpenStreetMap project](#) which are normally updated every day. Select your continent and then your country of interest from the list below. (If you have been directed to this page from elsewhere and are not familiar with OpenStreetMap, we highly recommend that you read up on OSM before you use the data.) This open data download service is offered free of charge by Geofabrik GmbH.


Willkommen auf dem Geofabrik-Downloadserver. Hier gibt es Daten-Auszüge aus dem [OpenStreetMap-Projekt](#), die normalerweise täglich aktualisiert werden. Wählen Sie aus dem Verzeichnis unten den Kontinent und ggf. das Land, für die Sie Daten benötigen. (Wenn Sie von anderswo auf dieser Seite gelandet sind und von OpenStreetMap nichts wissen, dann ist es empfehlenswert, sich mit dem Projekt vertraut zu machen, bevor Sie mit den Daten arbeiten.) Diese Downloads werden von der Geofabrik GmbH kostenlos angeboten.

Click on the region name to see the overview page for that region, or select one of the file extension links for quick access.

Sub Region	Quick Links		
	.osm.pbf	.shp.zip	.osm.bz2
Africa	[.osm.pbf] (4.2 GB)	[.shp.zip]	[.osm.bz2]
Antarctica	[.osm.pbf] (29.1 MB)	[.shp.zip]	[.osm.bz2]
Asia	[.osm.pbf] (8.9 GB)	[.shp.zip]	[.osm.bz2]
Australia and Oceania	[.osm.pbf] (828 MB)	[.shp.zip]	[.osm.bz2]
Central America	[.osm.pbf] (419 MB)	[.shp.zip]	[.osm.bz2]
Europe	[.osm.pbf] (22.4 GB)	[.shp.zip]	[.osm.bz2]
North America	[.osm.pbf] (9.9 GB)	[.shp.zip]	[.osm.bz2]

Not what you were looking for? Geofabrik is a consulting and software development firm based in Karlsruhe, Germany specializing in OpenStreetMap services. We're happy to help you with data preparation, processing, server setup and the like. [Check out our web site](#) and contact us if we can be of service.

Nicht das Richtige dabei? Die Geofabrik ist ein auf OpenStreetMap spezialisiertes Beratungs- und Softwareentwicklungsbüro mit Sitz in Karlsruhe, Deutschland. Wir helfen Ihnen gerne bei der Datenvorbereitung, -verarbeitung, -serveraufbau und ähnlichem. [Besuchen Sie unsere Website](#) und kontaktieren Sie uns, falls wir Ihnen weiterhelfen können.

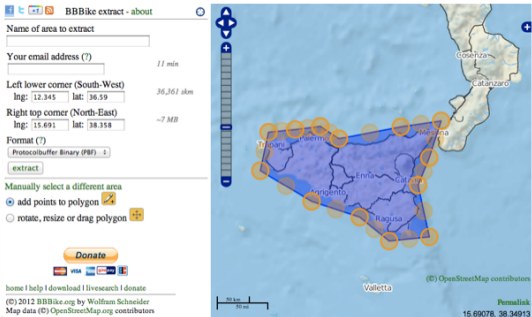


Geofabrik downloads

Fig. 3: Download osm data from Geofabrik

Welcome to BBBike's free download server! This server has data extracts from the OpenStreetMap project for more than 200 areas world wide in different formats and customized extracts.

- Download a ready extract of **more than 200 areas world wide**, 2-50MB in size for each city. Supported formats are: OSM, PBF, GeoJSON, SQLite, Garmin (style OSM, Cycle, Leisure, Onroad, Openfietstyle, OpenSeaMap, OpenTopoMap, BBBike), Osmand, mapsforge, Navit, maps.me, SVG, and Esri Shapefile.
- Didn't find the area you want? **Select your own region** - a rectangle or polygon up to 6000 x 4000km large, or 512MB file size.
- For experts only: download the **full planet**, in PBF (40GB) format



BBBike extract - about

Name of area to extract

Your email address (?) 11 min

Left lower corner (South-West) lat: 12.345 lat: 36.59 36,561 km

Right top corner (North-East) lat: 15.691 lat: 38.358 ~7 MB

Format (?)

Download the binary (PBF) extract

Manually select a different area

add points to polygon

rotate, resize or drag polygon

Donate

home | help | download | live search | donate

(C) 2012 BBBike.org by Wolfen Schneider

Map data (C) OpenStreetMap contributors

Penalick 15.69078 38.34913

Fig. 4: Download osm data from BBBike

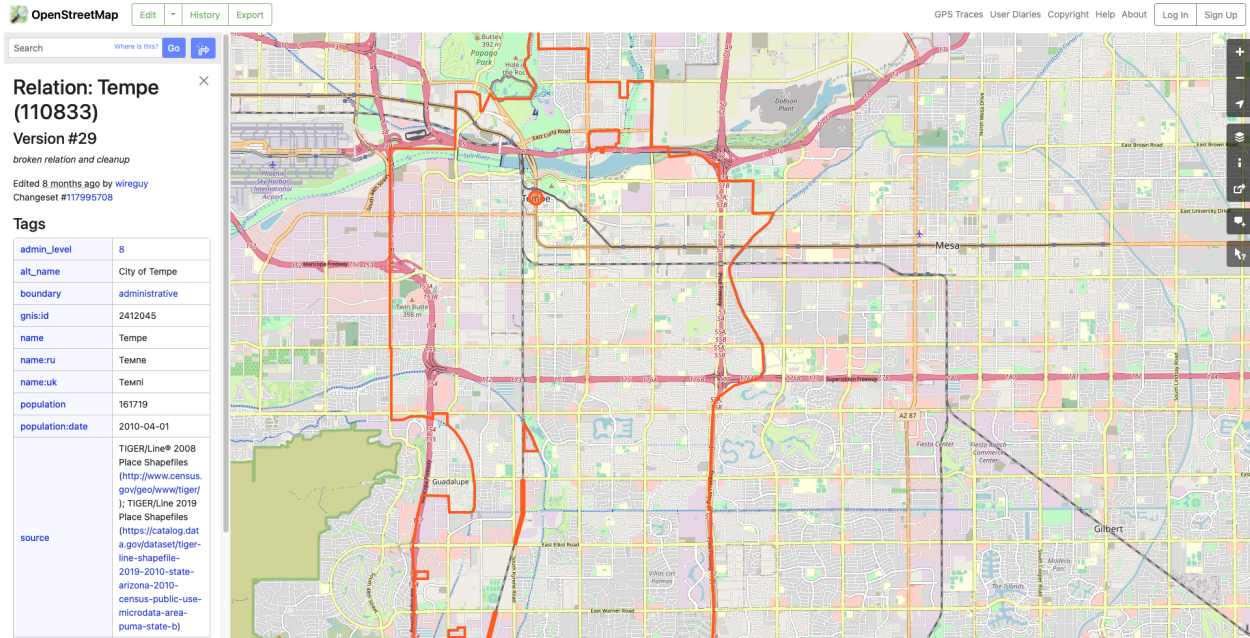


Fig. 5: Get region id from OpenStreetMap homepage

3.4.2 Parse OSM Data

We use the region around Arizona State University, Tempe Campus in this guide to introduce some major functions in osm2gmns.

Obtain a transportation network from an osm file.

```
>>> import osm2gmns as og

>>> net = og.getNetFromFile('asu.osm')
```

A link will be included in the network file from osm database if part of the link lies in the region that users selected. If argument `strict_mode` (default: `True`) is set as `True`, link segments that outside the region will be cut off when parsing osm data. If argument `strict_mode` is set as `False`, all links in the network file will be imported.

One loaded network may contain several sub networks, with some sub networks not accessible from others. In most cases, these sub networks include a large sub network and some isolated nodes or links. When the number of nodes of a sub network is less than argument `min_nodes` (default: `1`), this sub network will be discarded.

Users can use argument `combine` (default: `False`) to control short link combinations. If `combine` is enabled, two-degree nodes (nodes with one incoming link and one outgoing link) will be removed, and two adjacent links will be combined to generate a new link. Note that link combination will be performed only when two candidate links have the exact same link attributes, e.g., name, speed, lanes.

Notice that most links do not have “lanes” information in the map data provided by OpenStreetMap. Thus, we use a default lanes dictionary for each link type in osm2gmns. By setting `default_lanes` (default: `False`) as `True`, the default value will be assigned to a link if it does not come with “lanes” information. The default dictionary in osm2gmns:

```
default_lanes_dict = {'motorway': 4, 'trunk': 3, 'primary': 3, 'secondary': 2, 'tertiary': 2,
                      'residential': 1, 'service': 1, 'cycleway': 1, 'footway': 1, 'track': 1}
```

(continues on next page)

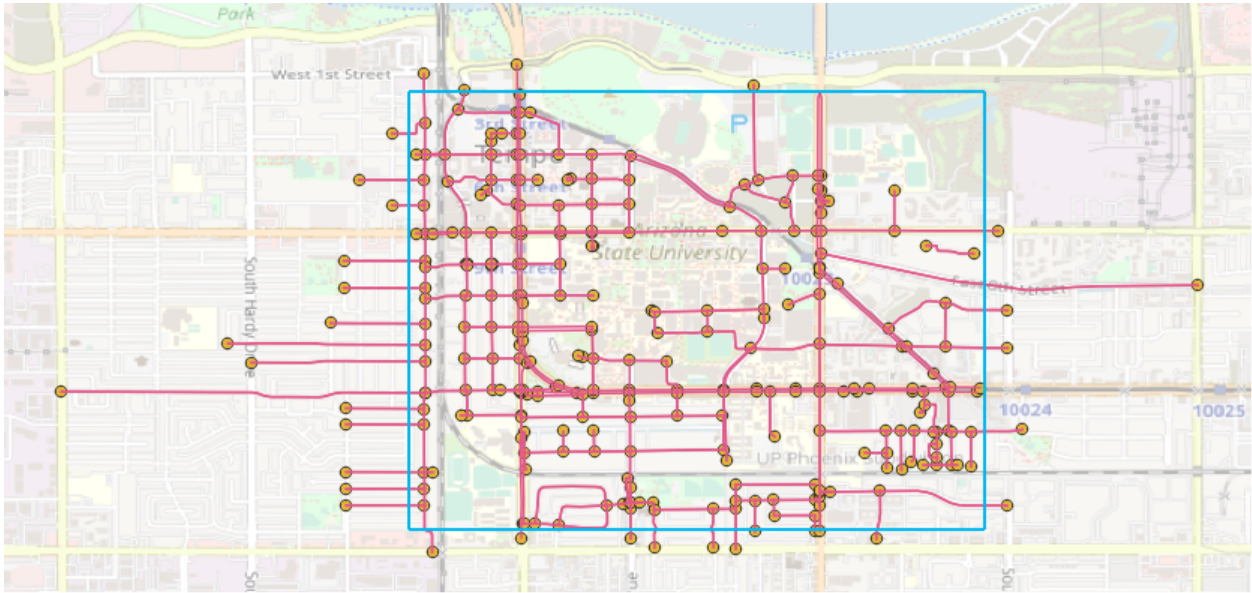


Fig. 6: Parsed network with `strict_mode=False`

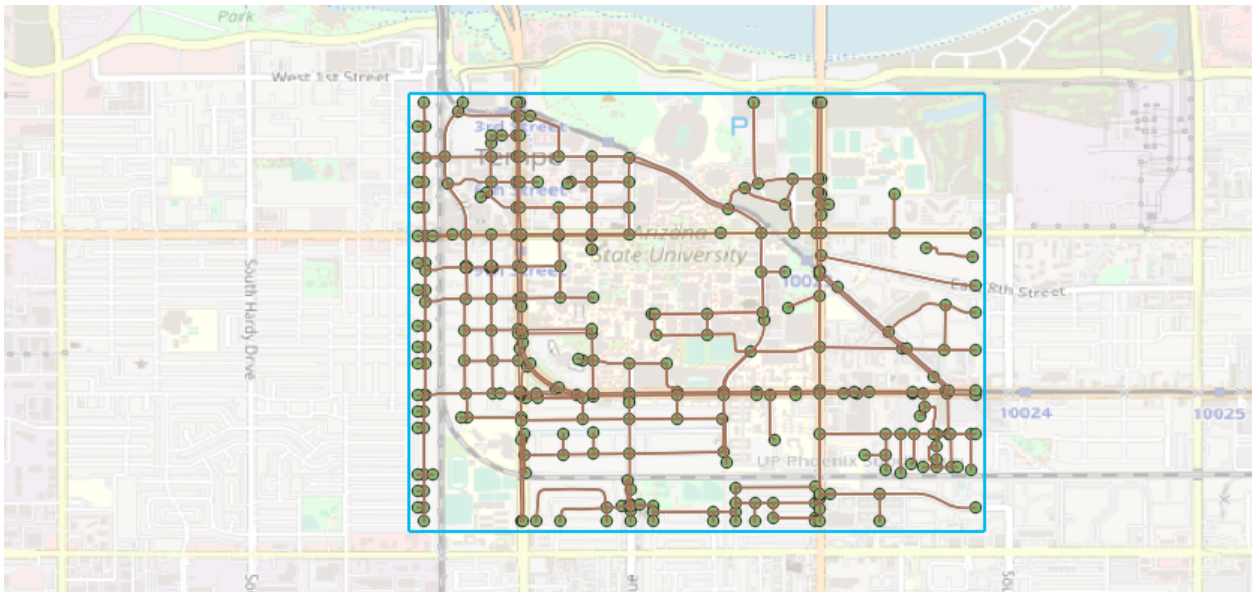


Fig. 7: Parsed network with `strict_mode=True`

(continued from previous page)

```

↪ ':1,
                        'unclassified': 1, 'connector': 2}
default_speed_dict = {'motorway': 120, 'trunk': 100, 'primary': 80, 'secondary': 60,
↪ 'tertiary': 40,
                        'residential': 30, 'service': 30, 'cycleway':5, 'footway':5, 'track
↪ ':30,
                        'unclassified': 30, 'connector':120}
default_capacity_dict = {'motorway': 2300, 'trunk': 2200, 'primary': 1800, 'secondary':
↪ 1600, 'tertiary': 1200,
                        'residential': 1000, 'service': 800, 'cycleway':800, 'footway':800,
↪ 'track':800,
                        'unclassified': 800, 'connector':9999}

```

`default_lanes` also accepts a dictionary. In that case, `osm2gmns` will use the dictionary provided by users to update the default dictionary.

A similar fashion applies for argument `default_speed` and `default_capacity`.

3.4.3 Output Networks to CSV

Based on the `net` instance obtained from the last step, `outputNetToCSV` can be used to output the parsed network to CSV files.

```
>>> og.outputNetToCSV(net)
```

Users can use argument `output_folder` to specify the folder to store output files. Node information will be written to `node.csv`, while link information will be written to `link.csv`.

3.4.4 Consolidate Intersections

In OpenStreetMap, one large intersection is often represented by multiple nodes. This structure brings some difficulties when performing traffic-oriented modelings. `osm2gmns` enables users to consolidate intersections that are originally represented by multiple nodes into a single node. Note that `osm2gmns` only identifies and consolidates signalized intersections.

```

>>> net = og.getNetFromFile('asu.osm')
>>> og consolidateComplexIntersections(net, auto_identify=True)
>>> og.outputNetToCSV(net)

```

Users can visualize the consolidated network in [QGIS](#) or [NeXTA](#). For complex intersections that were not successfully identified and consolidated by `osm2gmns`, users can manually specify them by revising the column “`intersection_id`” in `node.csv` and utilize the commands below to do the re-consolidation. Nodes assigned with the same “`intersection_id`” will be consolidated into a new node.

```

>>> net = og.loadNetFromCSV(node_file='node.csv', link_file='link.csv')
>>> og consolidateComplexIntersections(net, auto_identify=False)
>>> og.outputNetToCSV(net, output_folder='consolidated')

```

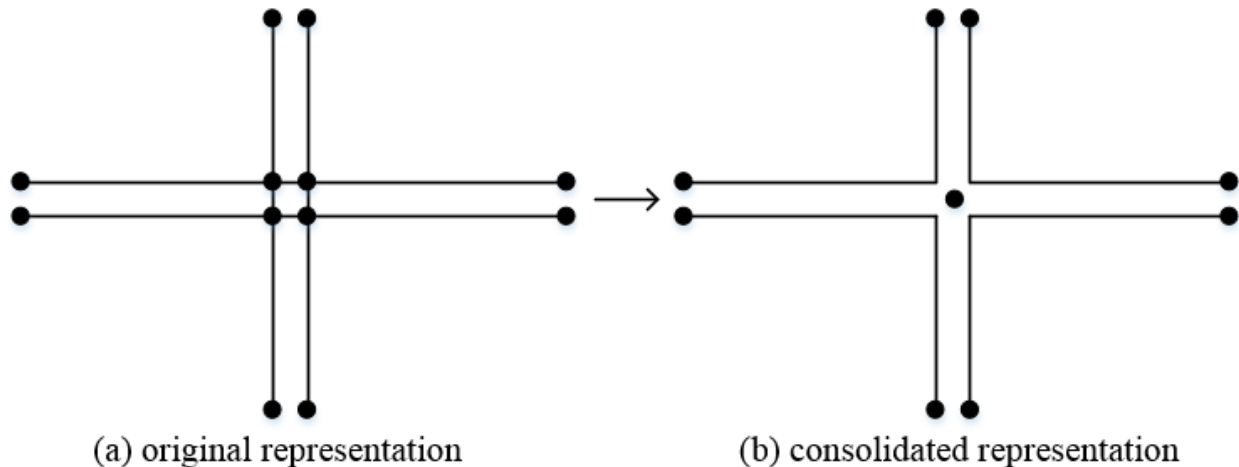


Fig. 8: Complex intersection consolidation

3.4.5 Network Types and POI

osm2gmns supports five different network types, including auto, bike, walk, railway, aeroway. Users can get different types of networks by specifying the argument `network_types` (default: `(auto,)`).

```
>>> # obtain the network for bike
>>> net = og.getNetFromFile('asu.osm', network_types='bike')
>>> # obtain the network for walk and bike
>>> net = og.getNetFromFile('asu.osm', network_types=('walk', 'bike'))
>>> # obtain the network for auto, railway and aeroway
>>> net = og.getNetFromFile('asu.osm', network_types=('auto', 'railway', 'aeroway'))
```

Obtain POIs (Point of Interest) from osm map data.

```
>>> net = og.getNetFromFile('asu.osm', POI=True)
```

If POI (default: `False`) is set as `True`, a file named `poi.csv` will be generated when outputting a network using function `outputNetToCSV`.

Connect POIs with transportation network.

```
>>> net = og.getNetFromFile('asu.osm', POI=True)
>>> og.connectPOIWithNet(net)
```

By using function `connectPOIWithNet`, a node located at the centroid of each POI will be generated to represent the POI. Then connector links will be built to connect the POI node with the nearest node in the transportation network.

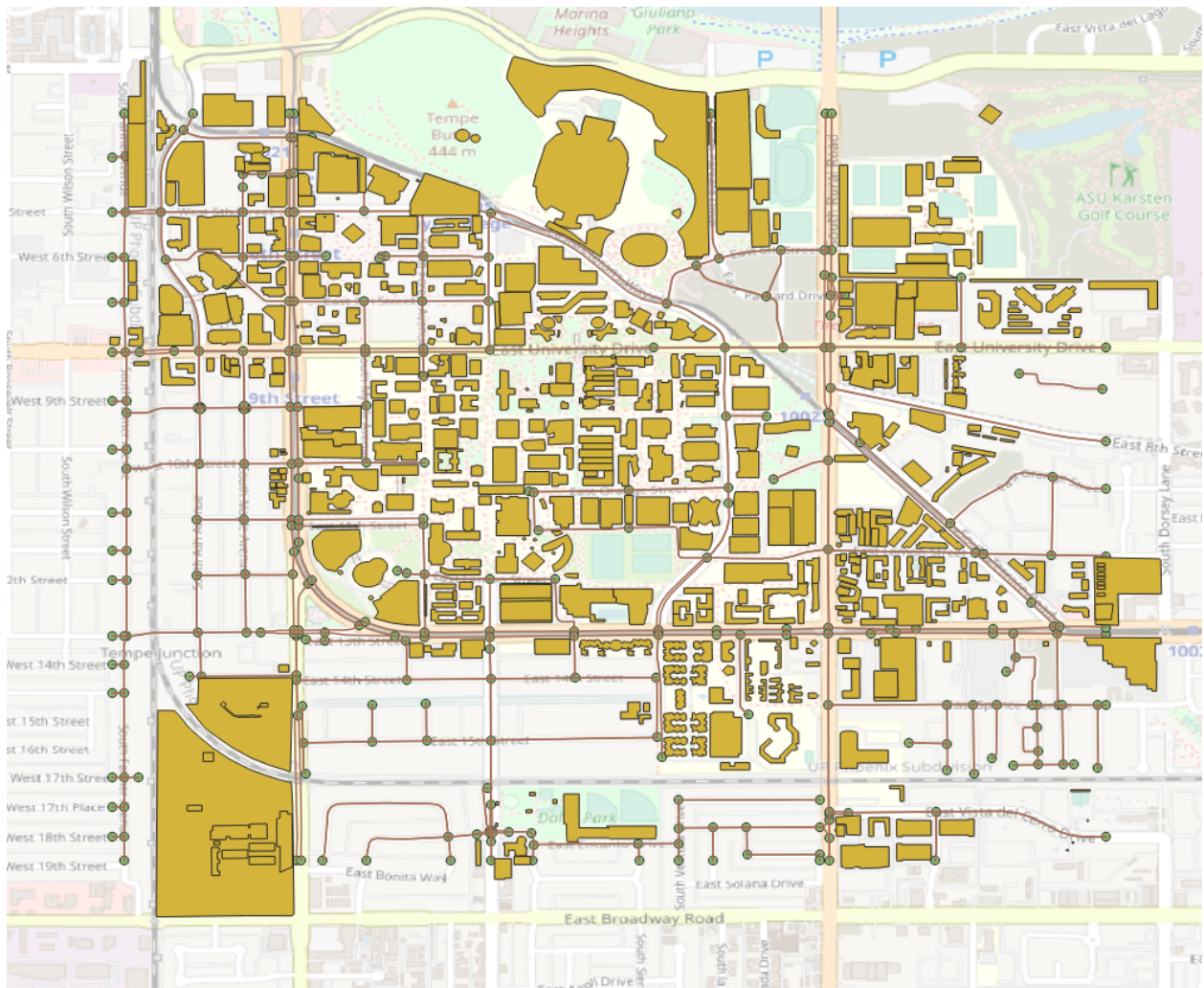


Fig. 9: Network with POIs

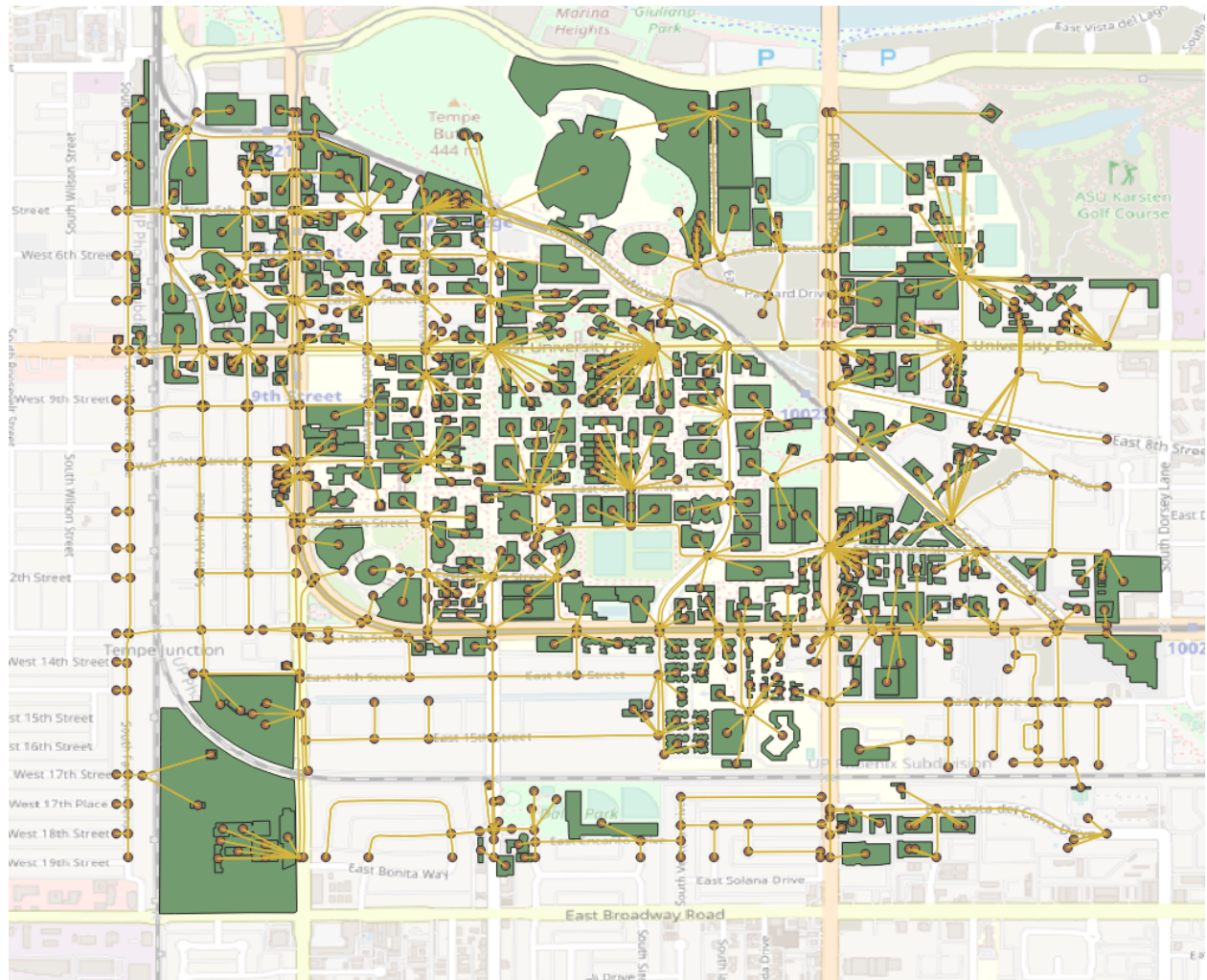


Fig. 10: Connect POIs with network

3.4.6 Generate Multi-Resolution Networks

osm2gmns can generate the corresponding mesoscopic and microscopic network for any macroscopic networks in GMNS format.

Generate multi-resolution networks from an osm file.

```
>>> net = og.getNetFromFile('asu.osm', default_lanes=True)
>>> og consolidateComplexIntersections(net, auto_identify=True)
>>> og.buildMultiResolutionNets(net)
>>> og.outputNetToCSV(net)
```

Generate multi-resolution networks from a macroscopic network provided by user (movement information is automatically generated by osm2gmns).

```
>>> net = og.loadNetFromCSV(node_file='node.csv', link_file='link.csv')
>>> og.buildMultiResolutionNets(net)
>>> og.outputNetToCSV(net)
```

Generate multi-resolution networks from a macroscopic network provided by user (movement information is from the user).

```
>>> net = og.loadNetFromCSV(node_file='node.csv', link_file='link.csv', movement_file=
↪ 'movement.csv')
>>> og.buildMultiResolutionNets(net)
>>> og.outputNetToCSV(net)
```

3.5 Functions

3.5.1 osm2gmns.osmnet

```
osm2gmns.osmnet.build_net.getNetFromFile(filename='map.osm', network_types=('auto',), link_types='all',
                                          POI=False, POI_sampling_ratio=1.0, strict_mode=True,
                                          offset='no', min_nodes=1, combine=False, bbox=None,
                                          default_lanes=False, default_speed=False,
                                          default_capacity=False, start_node_id=0, start_link_id=0)
```

Get an osm2gmns Network object from an osm file

Parameters

- **filename** (*str*) – path of an osm file; can be absolute or relative path; supported osm file formats: .osm, .xml, and .pbf
- **network_types** (*str, tuple of strings, list of strings, or set of strings*) – osm2gmns supports five different network types, including auto, bike, walk, railway, and aeroway. network_types can be any one or any combinations of the five supported network types
- **link_types** (*str, tuple of strings, list of strings, or set of strings*) – supported link types: motorway, trunk, primary, secondary, tertiary, residential, service, cycleway, footway, track, unclassified, connector, railway, and aeroway.
- **POI** (*bool*) – if extract point of interest information

- **POI_sampling_ratio** (*float*) – percentage of POIs to be extracted if POI is set as True. this value should be a float number between 0.0 and 1.0.
- **strict_mode** (*bool*) – if True, network elements (node, link, poi) outside the boundary will be discarded
- **offset** (*str*) – offset overlapping links. the value of this argument can be 'left', 'right', or 'no'
- **min_nodes** (*int*) – a network return by the function may contain several sub-networks that are disconnected from each other. sub-networks with the number of nodes less than min_nodes will be discarded
- **combine** (*bool*) – if True, adjacent short links with the same attributes will be combined into a long link. the operation will only be performed on short links connected with a two-degree nodes (one incoming link and one outgoing link)
- **bbox** (*tuple of four float/int values, list of four float/int values, None*) – specify the boundary of the network to be extracted, consisting of minimum latitude, minimum longitude, maximum latitude, and maximum longitude. if None, osm2gmns will try to find network boundary from the input osm file
- **default_lanes** (*bool, dict*) – if True, assign a default value for links without lanes information based on built-in settings. if a dict, assign a default value for links without lanes information based on the dict passed by users.
- **default_speed** (*bool, dict*) – if True, assign a default value for links without speed information based on built-in settings. if a dict, assign a default value for links without speed information based on the dict passed by users.
- **default_capacity** (*bool, dict*) – if True, assign a default value for links without capacity information based on built-in settings. if a dict, assign a default value for links without capacity information based on the dict passed by users.
- **start_node_id** (*int*) – osm2gmns assigns node_ids to generated nodes starting from start_node_id.
- **start_link_id** (*int*) – osm2gmns assigns link_ids to generated links starting from start_link_id

Returns

network – osm2gmns Network object

Return type

Network

```
osm2gmns.osmnet.complex_intersection.consolidateComplexIntersections(network,
                                                                    auto_identify=False,
                                                                    intersection_file=None,
                                                                    int_buffer=20.0)
```

Consolidate each complex intersection that are originally represented by multiple nodes in osm into one node. Nodes with the same intersection_id will be consolidated into one node. intersection_id of nodes can be obtained in three ways.

- (1) set the argument auto_identify as True, then osm2gmns will automatically identify complex intersections and assign intersection_id for corresponding nodes.
- (2) provide an intersection file that specifies the central position (required) and buffer (optional) of each complex intersection.
- (3) user can assign intersection_id to nodes manually in network csv files (node.csv), and load the network using function loadNetFromCSV provided by osm2gmns.

The priority of the three approaches is (3) > (2) > (1). Rules used in the approach (1) to identify if two nodes belong to a complex intersection: (a) `ctrl_type` of the two nodes must be signal; (b) there is a link connecting these two nodes, and the length of the link is shorter than or equal to the argument `int_buffer`.

Parameters

- **network** (*Network*) – osm2gmns Network object
- **auto_identify** (*bool*) – if automatically identify complex intersections using built-in methods in osm2gmns. nodes that belong to a complex intersection will be assigned with the same `intersection_id`
- **intersection_file** (*str*) – path of an intersection csv file that specifies complex intersections. required fields: central position of intersections (in the form of `x_coord` and `y_coord`); optional field: `int_buffer` (if not specified, the global `int_buffer` will be used, i.e., the forth argument). For each record in the `intersection_file`, osm2gmns consolidates all nodes with a distance to the central position shorter than `buffer`.
- **int_buffer** (*float*) – the threshold used to check if two nodes belong to one complex intersection. the unit is meter

Return type

None

`osm2gmns.osmnet.combine_links.combineShortLinks(network)`

Combine links connected by two-degree nodes into a longer link

Parameters

network (*Network*) – osm2gmns Network object

Return type

None

`osm2gmns.osmnet.enrich_net_info.generateNodeActivityInfo(network)`

Generate activity information, including `activity_type`, `is_boundary`, `zone_id` for nodes. `activity_type` includes motorway, primary, secondary, tertiary, residential, etc, and is determined by adjacent links

Parameters

network (*Network*) – osm2gmns Network object

Return type

None

`osm2gmns.osmnet.enrich_net_info.generateLinkVDFInfo(network)`

Generate VDF information, including `VDF_fftt1` and `VDF_cap1` for links. The unit of `VDF_fftt1` and `VDF_cap1` are min and veh/hour/link, respectively

Parameters

network (*Network*) – osm2gmns Network object

Return type

None

`osm2gmns.osmnet.pois.connectPOIWithNet(network)`

Connect POIs with the traffic network. Specifically, for each POI, osm2gmns will build a bi-directional connector to connect the POI with its nearest node in the traffic network

Parameters

network (*Network*) – an osm2gmns Network object

Return type

None

`osm2gmns.osmnet.visualization.show(network, save=False, figsize=None)`

Show the network in a pop-up window

Parameters

- **network** (*Network*) – an osm2gmns Network object
- **save** (*bool*) – if True, the plot will also be saved to a local file named network.jpg
- **figsize** (*tuple of int/float, list of int/float*) – size of the figure

Return type

None

`osm2gmns.osmnet.visualization.saveFig(network, picpath='network.jpg', figsize=None)`

Save the network plot to a local file

Parameters

- **network** (*Network*) – an osm2gmns Network object
- **picpath** (*str*) – path to store to network plot. can be an absolute or a relative path
- **figsize** (*tuple of int/float, list of int/float*) – size of the figure

Return type

None

3.5.2 osm2gmns.io

`osm2gmns.io.load_from_csv.loadNetFromCSV(folder='', node_file=None, link_file=None, movement_file=None, segment_file=None, geometry_file=None, POI_file=None, coordinate_type='lonlat', encoding=None)`

Load a network from csv files in GMNS format

Parameters

- **folder** (*str*) – the folder where network files are stored
- **node_file** (*str*) – filename of the node file. required
- **link_file** (*str*) – filename of the link file. required
- **movement_file** (*str, None*) – filename of the movement file. optional
- **segment_file** (*str, None*) – filename of the segment file. optional
- **geometry_file** (*str, None*) – filename of the geometry file. optional
- **POI_file** (*str, None*) – filename of the POI file. optional
- **coordinate_type** (*str*) – the coordinate system used by the network to be loaded. can be lonlat, meter, feet
- **encoding** (*str, None*) – the encoding used by the network files. if None, osm2gmns will use the default encoding of the local operating system

Returns

network – an osm2gmns Network object

Return type

Network

```
osm2gmns.io.downloader.downloadOSMData(area_id, output_filename='map.osm',  
                                         url='www.overpass-api.de/api/interpreter')
```

Download OpenStreetMap data via overpass API

Parameters

- **area_id** (*int*) – relation_id of the area of interest
- **output_filename** (*int*) – full path where the downloaded network will be stored
- **url** (*int*) – OpenStreetMap API url

Return type

None

```
osm2gmns.io.writefile.outputNetToCSV(network, output_folder="", prefix="", projection=False,  
                                       encoding=None)
```

Output an osm2gmns network object to csv files in GMNS format

Parameters

- **network** (*Network*) – an osm2gmns network object
- **output_folder** (*str*) – path of the folder to store network files. can be an absolute or a relative path
- **prefix** (*str*) – prefix of output csv files
- **projection** (*bool*) – if True, osm2gmns will project the network to a local coordinate system when ouptting a network
- **encoding** (*str*) – the file encoding used to output a network

Return type

None

3.5.3 osm2gmns.movement

```
osm2gmns.movement.generate_movements.generateMovements(network)
```

Use osm2gmns built-in methods to generate movements for each node (intersection) in a network

Parameters

network (*Network*) – an osm2gmns Network object

Return type

None

3.5.4 osm2gmns.multipresolutionnet

```
osm2gmns.multipresolutionnet.build_mrnet.buildMultiResolutionNets(macronet,  
                                                                    generate_micro_net=True,  
                                                                    auto_movement_generation=True,  
                                                                    exclu-  
                                                                    sive_bike_walk_lanes=True,  
                                                                    connector_type=None,  
                                                                    width_of_lane=3.5,  
                                                                    length_of_cell=7.0)
```

Build the corresponding mesoscopic and microscopic networks for a source (macroscopic) network

Parameters

- **macronet** (*Network*) – a source osm2gmns Network object
- **generate_micro_net** (*bool*) – True: generate meso and micro networks; False: only generate meso network
- **auto_movement_generation** (*bool*) – automatically generate movements for intersections without movement information by calling function generateMovements in osm2gmns. if auto_movement_generation is set as False, movements at intersections without movement information will not be generated
- **exclusive_bike_walk_lanes** (*bool*) – build exclusive lanes for bike and walk
- **connector_type** (*int*) – link_type of connectors
- **width_of_lane** (*float*) – width of lanes in meter
- **length_of_cell** (*float*) – length of cells in meter

Return type

None

3.6 Sample Networks

3.6.1 Phoenix Sky Harbor International Airport

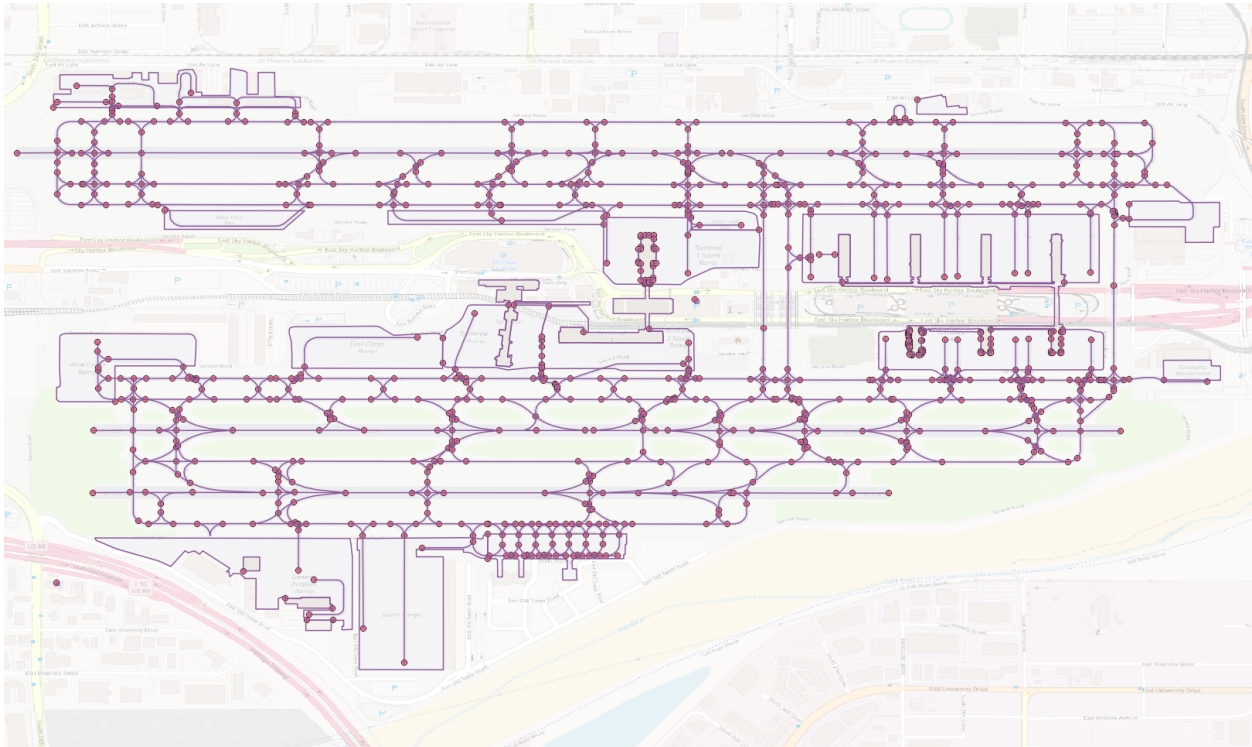


Fig. 11: Phoenix Sky Harbor International Airport

3.6.2 Arizona State University, Tempe Campus

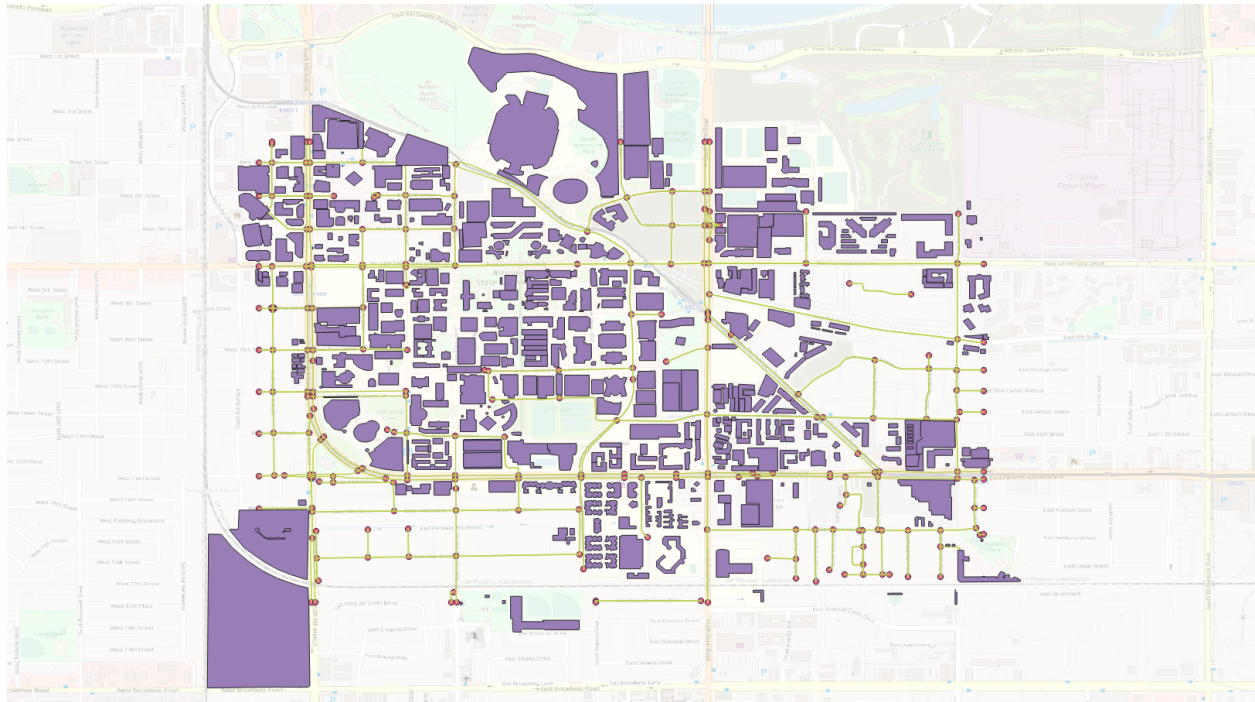


Fig. 12: Arizona State University, Tempe Campus

3.6.3 Arizona, US

3.6.4 US railway network (midwest)

3.6.5 Greater London, UK

3.6.6 Melbourne, Australia

3.7 Acknowledgement

This project is partially supported by National Science Foundation - United States under Grant No. CMMI 1663657 “Collaborative Research: Real-time Management of Large Fleets of Self-Driving Vehicles Using Virtual CyberTracks”

The second author also thanks for the early support from FHWA project titled “The Effective Integration of Analysis, Modeling, and Simulation Tools - AMS Data hub Concept of Operations”. <https://www.fhwa.dot.gov/publications/research/operations/13036/004.cfm>

Many thanks for GMNS specification efforts led by Scott Smith and Ian Berg from Volpe Center, USDOT. Their TRB poster can be found at https://github.com/zephyr-data-specs/GMNS/blob/TRB/TRBPoster_22-02127.pdf

This document is prepared with the help from Entai Wang and Chongnan Li.

For program source code and sample network files, readers can visit the project [homepage](#) at ASU Trans+AI Lab Github. Interested readers can also check the [link](#) for our online transportation modelling visualization platform, in which network data are provided by osm2gmns.

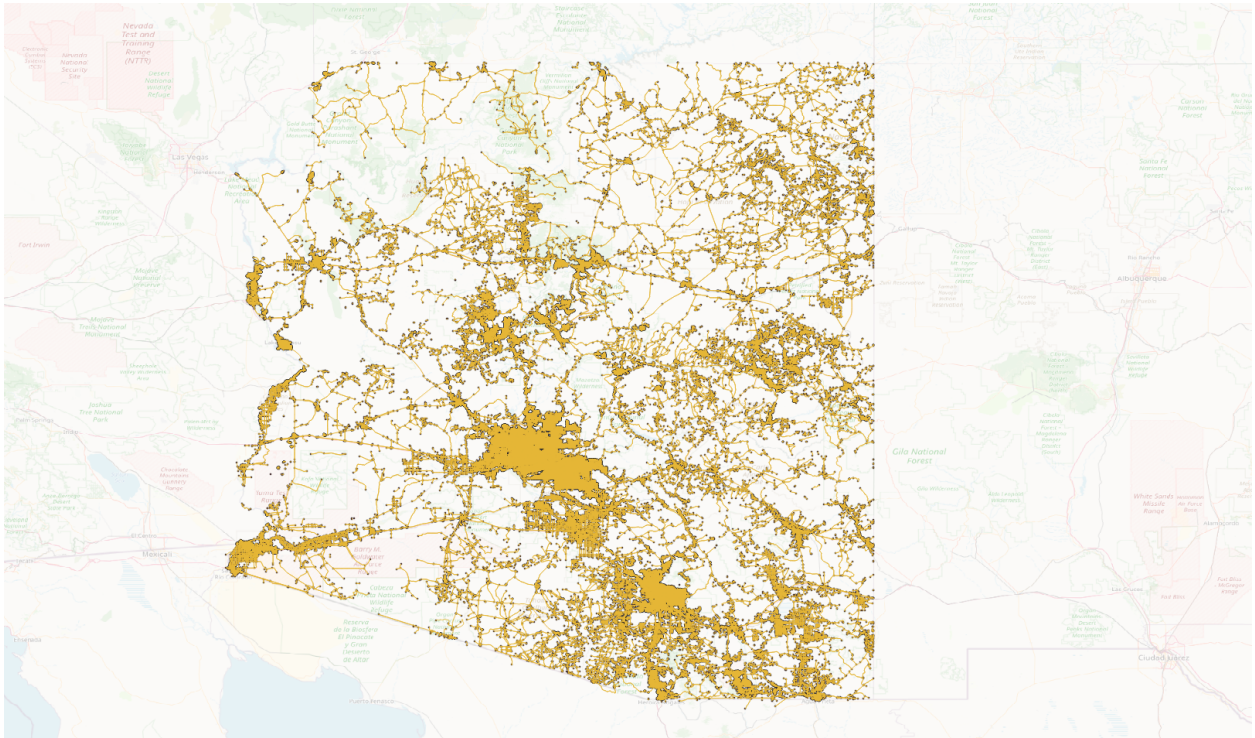


Fig. 13: Arizona, US

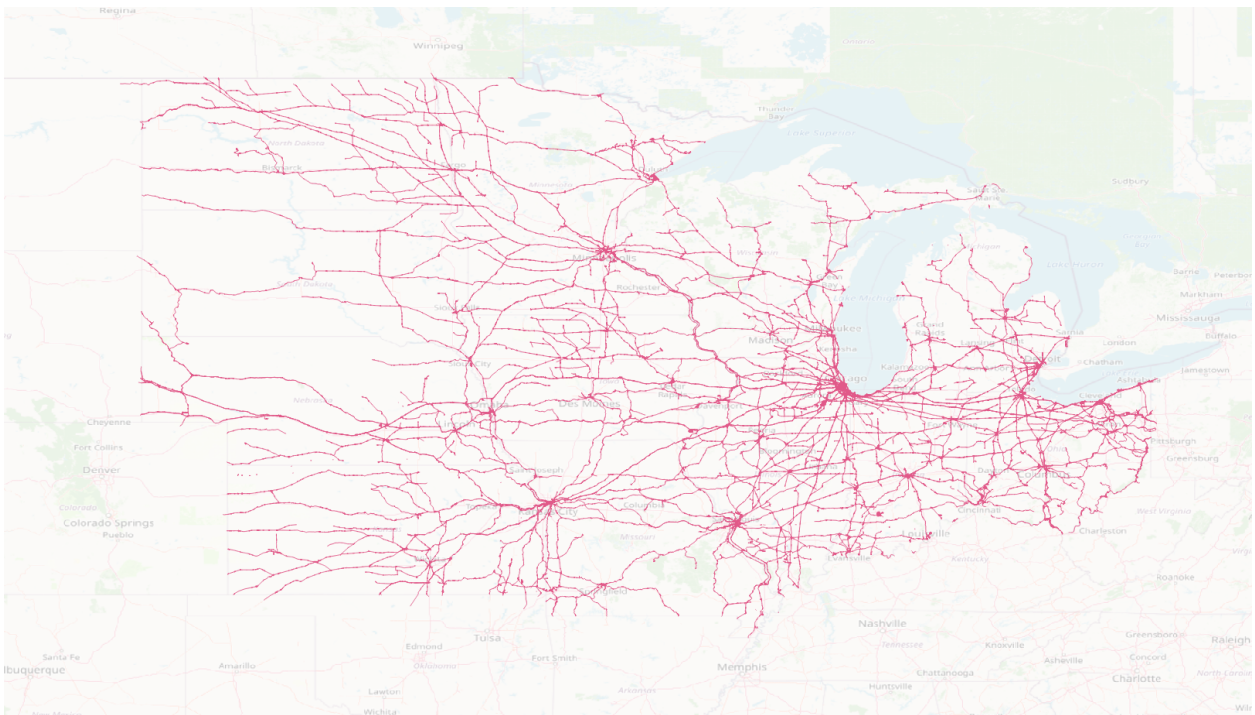


Fig. 14: US railway network (midwest)

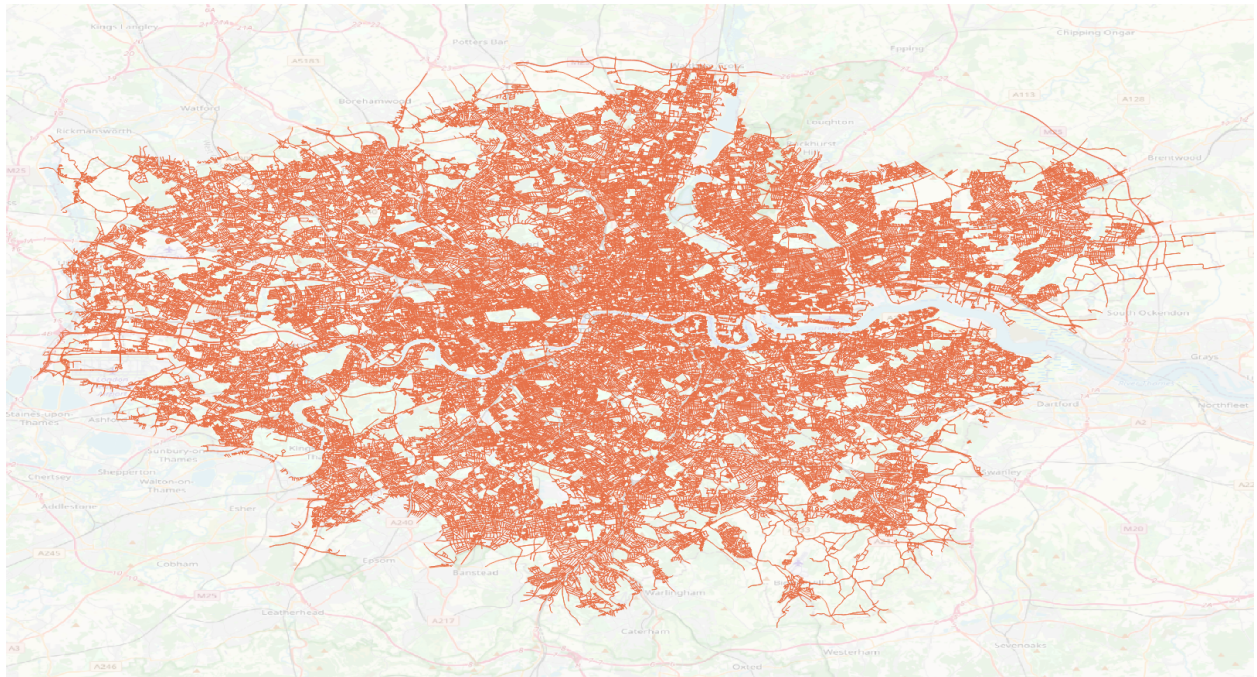


Fig. 15: Greater London, UK

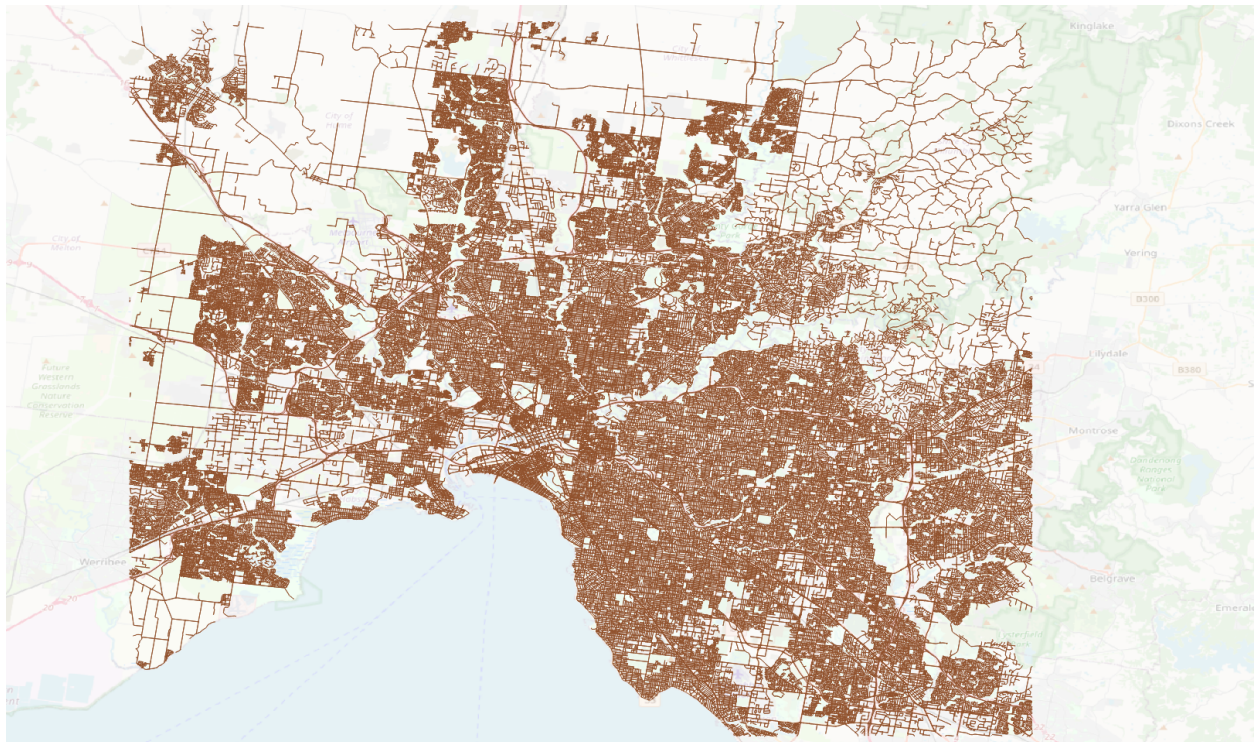


Fig. 16: Melbourne, Australia

INDEX

B

`buildMultiResolutionNets()` (in module *osm2gmns.multiresolutionnet.build_mrnet*), 26

C

`combineShortLinks()` (in module *osm2gmns.osmnet.combine_links*), 24

`connectPOIWithNet()` (in module *osm2gmns.osmnet.pois*), 24

`consolidateComplexIntersections()` (in module *osm2gmns.osmnet.complex_intersection*), 23

D

`downloadOSMData()` (in module *osm2gmns.io.downloader*), 25

G

`generateLinkVDFInfo()` (in module *osm2gmns.osmnet.enrich_net_info*), 24

`generateMovements()` (in module *osm2gmns.movement.generate_movements*), 26

`generateNodeActivityInfo()` (in module *osm2gmns.osmnet.enrich_net_info*), 24

`getNetFromFile()` (in module *osm2gmns.osmnet.build_net*), 22

L

`loadNetFromCSV()` (in module *osm2gmns.io.load_from_csv*), 25

O

`outputNetToCSV()` (in module *osm2gmns.io.writefile*), 26

S

`saveFig()` (in module *osm2gmns.osmnet.visualization*), 25

`show()` (in module *osm2gmns.osmnet.visualization*), 24