`OSMNetFusion`: Topological Simplification for Multimodal Networks with Attribute-Preservation and Enrichment

Documentation

Victoria Dahmen*

June 27, 2025

1 Introduction & Motivation

The `OSMNetFusion` package is a tool designed to topologically simplify and enrich OpenStreetMap (OSM) networks to facilitate their use for further mobility or geospatial analyses. It provides researchers, transport analysts and urban planners with a simplified, information-rich, multimodal network that retains essential information for various transport applications. With only the location of interest as required input, this tool reduces network complexity by merging edges (roads) and nodes (intersections) and preserves essential tag information, while also adding additional open-source data. Additionally, the resulting network supports the effortless selection of transport-mode-specific subnetworks (e.g., walking, cycling, motorised), making them easier to analyse and visualise.

Several existing open-source tools address network simplification for specific use-cases in geospatial and mobility analysis. For instance, Ballo and Axhausen developed a road space reallocation tool [1], *SNMan*, which includes a topological network simplification with a reconstruction and visualisation of the lanes per link [2]. Fleischmann and Vybornova focus on topological network simplification in a mathematically rigorous manner [3] and recently published *neatnet* [4]. Tools such as OSMnx [5] also offer simplification capabilities to a limited extent. However, `OSMNetFusion` specifically aims to simplify multimodal networks (pedestrian, cyclist, motorised traffic) while retaining OSM tag information and integrating additional open-source data, an aspect often absent in current solutions but crucial for comprehensive mobility analysis.

2 Methodology

The framework is composed of three key steps: the data retrieval, the network enrichment, and the network simplification. These are described in the subsequent sections.

2.1 Data retrieval

OSM networks contain a wide range of metadata tags covering information from road surface to speed limits and lighting conditions. The OSM platform also contains many items other than the road network, such as information about land-use, public transit, restaurants and mobility-related infrastructure. Additionally, there also exists other relevant open-source data. Hence, data is gathered from all three categories. As each data is either network-specific or valid for the entire region of interest, the data is saved in two folders depending on this characteristic, as denoted by in Table 1.

The OSM network, the foundational element of this tool, is retrieved for the location of interest using the *osmnx* package and it is saved as a geopackage file. The following parameters

*Chair of Traffic Engineering and Control, Technical University of Munich. Email: v.dahmen@tum.de

are used for retrieving the graph with osmnx: *network_type*='all', *simplify*=True, *retain_all*=False, *truncate_by_edge*=False. Due to the *simplify* attribute, the OSM network will already have reduced complexity compared to the actual OSM network, which speeds up the processing and simplification carried out with `OSMNetFusion`. The network contains nodes and edges, where each item has a set of tag information (e.g., road type, number of lanes).

Various additional data is downloaded from OSM, also using osmnx, which are outline in Table 1. The land-use related data (green areas, retail areas) will be used to later on calculate the share of the respective land-use along each edge. The PT stops and routes are extracted for two reasons: firstly, to add information about accessibility, and secondly, to be able to approximate the bus density on along the edges. The latter may affect a link's suitability for cycling or make it more prone to congestion. Infrastructure data like bicycle racks and traffic lights is only rarely included in edge or node information, hence this data is retrieved from OSM and re-added to the network.

Lastly, other open-source data gathered. Geographic elevation data is relevant for transport analysis, as elevation changes can impact walkability, cycling difficulty, and vehicle speed. It is obtained using an open-source tool. By default the elevation retrieval is disabled, to reduce the number of requests. For some regions (open-source) cycle path widths may be available; if so, this information can be added as well as a csv-file.

Name	**Network/ Regional data**	**Source**	**Comment**
Road network	Network	OSM	-
Retail areas	Regional	OSM	Includes shops and buildings
Green areas	Regional	OSM	Includes parks and natural spaces
PT Stops	Regional	OSM	Locations of public transit stops
PT Routes	Regional	OSM	Routes of buses, trams, etc.
Bike amenities	Regional	OSM	Bike repair stations, parking, etc.
Traffic signals	Regional	OSM	-
Cycle path widths	Regional	Csv file (if available)	If the form of [osmid, width]
Elevation	Network	Open-Elevation API	Queried for the nodes

Table 1: With only the location of interest as input, various data is automatically downloaded and added to the OSM network, ranging from land-use to public transport (PT) data. Apart from the optional cycle path width input, all are open-access.

2.2 Network enrichment

All the downloaded data is added to the OSM network to creates an enriched, detailed network that captures more than just the road layout. This multi-layered network provides the foundation for the last step, where the network is simplified (consolidated) to produce a model that supports various modes of transport and retains essential topographic and infrastructure characteristics.

The attributes that are added to the network based on the retrieved open-source data are summarised in Table 2. Generally, the data is either added to the nearest / nearby edges or nodes, as applicable. Some OSM native columns are merged as they contain redundant information. For the elevation and PT the data is further process to obtain the *gradient* and *severeness* for the former, and the list of PT routes halting on an edge for the latter. Furthermore, a new *cycleway_category* attributes is generated based on native OSM tags. It is also noted that for any edge that is bi-directional by foot or bike, the opposite edge (i.e., from V to U for an edge UV) is added whenever it is missing.

Type	Attribute	Type	Description
Edges	surface_smoothness	String	This is sometimes stored in the "_40" column, hence they are merged.
Edges	surface	String	This is sometimes stored in the "_30" column, hence they are merged.
Edges	width	Float	This is sometimes stored in the "_36" column, hence they are merged.
Edges	cycleway_category	String	Categorised based on other OSM tags: advisory lane, exclusive lane, shared lane, bicycle road, one-direction cycle path, two-direction cycle path, track/lane, foot & cycle path, pedestrian street.
Edges	gradient	Float	Calculated as: $(elev_{startNode} - elev_{endNode})/length$
Edges	severeness	Float	Calculated as: $(elev_{startNode} - elev_{endNode})^2/length$, representing the steepness severity.
Nodes	elevation	Float	-
Edges	cycle_path_width	Float	If available, [osmid, mean width] of cycle paths.
Nodes	traffic_signals	Boolean	Whether traffic signals are present at the node.
Edges	amenities_on	List(str)	List of bicycle amenities located on the edge.
Edges	amenities_nearby	List(str)	List of bicycle amenities within 200 m of the edge.
Edges	pt_stop_on	Boolean	Whether a PT stop is located on the edge.
Edges	pt_stop_count	Int	Number of PT stops located on the edge.
Edges	pt_stop_routes	List(str)	List of PT routes passing through the edge.

Table 2: Additional attributes are generated using open-source data

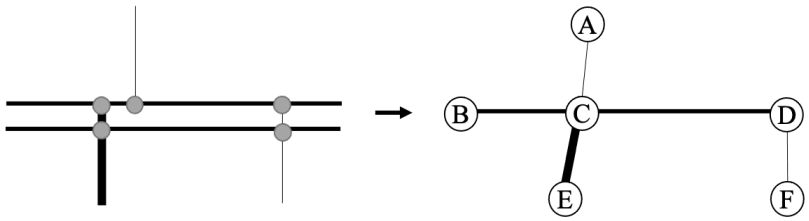
2.3 Network simplification

The topological simplification (and subsequent merging) consists of 8 key steps. The entire process is visualised in Fig. 1, where a simple example of the aim of the topological simplification is shown in Fig. 1a.

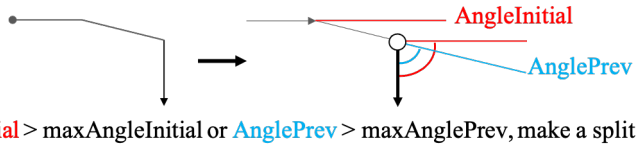
1. Curve Splitting, Fig. 1b: The simplification begins by breaking down curved edges into shorter, straight segments based on specific angle thresholds. Each segment in the network is evaluated based on its initial and previous angles; if these exceed preset thresholds (*maxAngleInitial* or *maxAnglePrev*), the curve is split into two segments at the point where these thresholds are breached. This step helps to ensure effective topological simplification by preserving only moderate or minimal curves in the network.

2. Hierarchy Quantification for Edges and Nodes, Fig. 1c and Fig. 1d: After splitting, the next stage involves quantifying the hierarchy of edges and nodes, assigning each a rank that reflects its importance within the network. Edges are given a hierarchy rank using the *Highway_Ranking* metric (customisable), which prioritises main roads like trunks and highways over smaller paths and footpaths. This metric can be customised if needed. The ranking allows the network to differentiate between various road types and preserve the fundamental structure of key transportation routes. Once edges are ranked, nodes inherit a hierarchy rank based on the average of the two highest-ranked edges connected to them. This node ranking further guides the simplification process by indicating the relative importance of specific intersections or routes, thus informing later steps like buffering and merging.

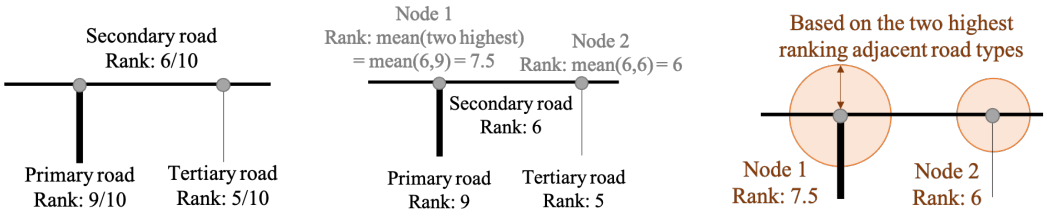
3. Node Buffering Based on Hierarchy, Fig. 1e: Following the ranking of edges and nodes, the tool buffers nodes according to their hierarchy rank, creating zones of influence around each nodes. Each node's buffer radius is determined by its rank; higher-ranked nodes, representing more critical intersections, receive larger buffer zones. This variation in buffer sizes ensures that major nodes have a broader area of influence, which will facilitate the clustering of nodes within these zones. The buffered regions created here help group densely packed nodes in later steps, thus establishing clusters that consolidate intersections while retaining essential connectivity.



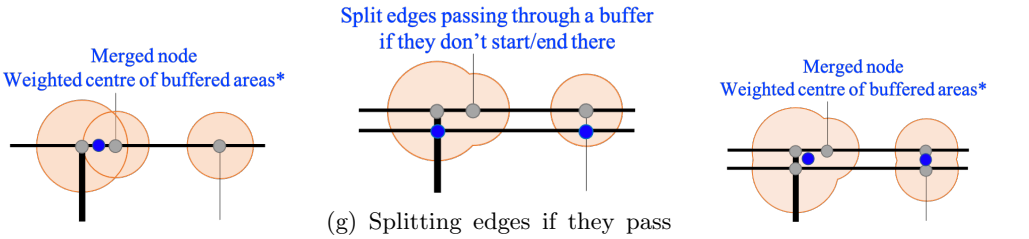
(a) Before vs. after the simplification.



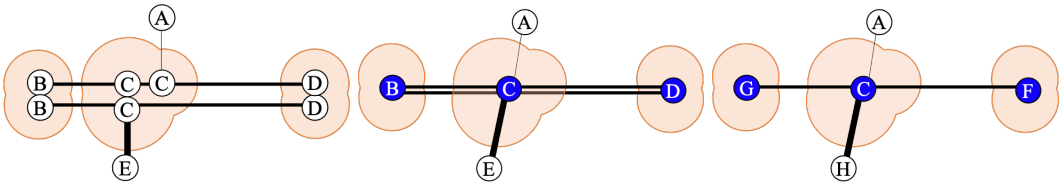
(b) Splitting of curves.



(c) Assignment of road hierarchy. (d) Determination of the node hierarchy. (e) Create a buffer around the nodes.



(f) Merging the nodes. (g) Splitting edges if they pass through buffered nodes. (h) Merge nodes again.



(i) Reassign the nodes. (j) Update all node information. (k) Update all edge information.

Figure 1: Simplification steps.

4. Clustering Nodes within Buffered Regions, Fig. 1f: The next step is to cluster any nodes within overlapping buffer regions into single points. This clustering is essential for reducing redundancy in areas where intersections and paths are densely packed. Nodes whose buffer zones overlap are grouped into clusters, effectively consolidating nearby nodes into representative single nodes. This approach simplifies complex intersections while preserving necessary connectivity. If the resulting cluster exceeds a specific node count threshold (*clusterThreshold*, default: 20), it is further divided into multiple smaller clusters using a clustering algorithm; this is further detailed in the next section. This clustering step effectively consolidates intersections and reduces network

complexity while preserving essential connectivity within the network.

5. Edge Splitting at Buffered Regions, Fig. 1g: To ensure accurate connectivity, the tool splits edges that pass through buffer zones without starting or ending within them. By analysing each edge, `OSMNetFusion` identifies edges that intersect buffered regions and divides them these intersections (at the point along the edge that is closest to the centroid of the buffer).

6. Clustering Nodes within Buffered Regions (Again), Fig. 1h: Upon splitting edges that pass through node buffers, the nodes are clustered again. The same approach is used as before, yet the new complete set of nodes is used.

7. Node Reassignment and Removal of Degenerate Nodes, Fig. 1i: As previously mentioned, clusters of nodes will contain one key/critical node, and any number of nodes that are associated to it. In this reassignment stage, all nodes in a cluster are reassigned to a cluster's key node to accurately represent the updated network layout. This is done by denoting the new start and end node. In the next stage the new node structure will be implemented by merging all nodes and edges, where applicable.

8. Final Node and Edge Merging, Fig. 1j and Fig. 1k: All nodes with the same key node (i.e., within the same node buffer) and all edges with the same reassigned start and end node are merged, both w.r.t. geometry and node/edge information. Nodes are combined into weighted centres representing their group, and edges between clustered nodes are consolidated. This reduces the network size and complexity significantly. For example, one application saw a reduction from 12,200 to 1,600 nodes, maintaining critical intersections while reducing complexity by 87%. Degenerate nodes—those that were in the buffer of, i.e., were assigned to, a different more critical node— and edges, will inherently disappear. By removing these nodes and edges, the network becomes clearer, retaining only essential connectivity points. This consolidation phase ensures that the network is simplified and uncluttered, while maintaining all connectivity and preserving information.

The consolidation process of the node/edge of all attributes (including the geometries) is explained in depth in the next section. The result is a compact, well-organised network that is both efficient for computational tasks and accurate and information-rich for practical applications.

2.4 Information Preservation and Tag Restructuring

It is no trivial task to merge nodes or edges which have a wide range of information associated to them. Hence, the structure visualised in Figure 2 has been conceptualised. For each consolidated edge, there is a set of attributes that are associated with a specific mode (bike, walk, PT, car) and a set of general/high-level attributes (like the object ID). Additionally, four tags denote the accessibility of an edge to the key modes of transport, ensuring the easy selection of mode-specific subnetworks. For a given link, each edge is directional, so if the link is bidirectional (i.e., not one-way for all modes that may access the link), *Edge UV* and *Edge VU* are separate entities. The object-oriented approach reflects this design. Each link object has one or two edges, which in turn may have walking, cycling, and/or motorized path objects.

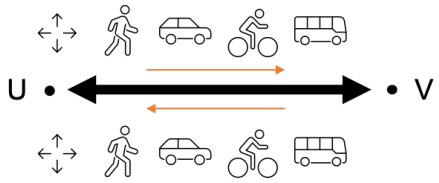


Figure 2: Structure of the information of the link (black arrow) and edge objects (orange arrows).

As the nodes have much fewer attributes than the edge objects, the merging process is more simplistic. Table 3 details all node attributes, an example and the data type, along with a description of the variable and how it was merged. For instance, the *g_crossing* attribute is determined by checking if there are *any* crossings documented in any of the considered nodes. The geometry of the nodes, i.e., the centre of each node cluster, is derived using one of two approaches. The weighted centre of the union of all nodes of the two highest node hierarchies in a cluster is used. In case the

number of nodes N is exceeds 50, the nodes are grouped into $\lceil N/50 \rceil$ clusters using the KMeans algorithm; this only applies to <2% of the resulting merged nodes.

Attribute	Example	Merging	Value Type	Description
geometry	Point(11.56 48.12)	Weighted centre (or clustering)	Shapely Point	Generally a weighted centre approach is used. *See text for detailed information.*
g_id	11516	Main node	Int	New edge id
g_x	11.58	-	Float	Longitude
g_y	45.15	-	Float	Latitude
g_infra	{traffic_signals}	Set	List(string)	Any highway or crossing infra.
g_crossing	False	Any	Boolean	Is there any crossing?
g_signals	True	Any	Boolean	Are there traffic signals?
l_hw_conn	{path, service}	Set	List(string)	Road type of the merged nodes
l_hw_rank	{3, 7}	Set	List(float)	Road rank of the merged nodes
l_osmid	{288676926}	Set	List(int)	OSM IDs of the merged nodes

Table 3: Node Attributes.

Attribute	Example	Merging	Value Type	Description
geometry	MultiLineString (11.56 48.12, ...)	Geometry of the main edge	Shapely (Multi-) LineString	Only edges with the same start and end node are merged
access_bik	True	Or	Boolean	Accessible by bike
access_mot	True	Or	Boolean	Accessible by car
access_wal	True	Or	Boolean	Accessible by foot
g_crossing	{traffic_signals}	Set	List(string)	Is there are crossing?
g_geo_lin	LineString (11.56 48.12, ...)	-	Shapely LineString	Straight line from the start to the end node
g_gradient	0.02	Mean	Float	Steepness
g_greenR	0.1	Mean	Float	Land-use: share of green areas
g_height_d	2	Mean	Float	Elevation change
g_id	6018	Main edge ID	Int	Edge ID
g_incline	{up}	Set	List(string)	Direction of incline
g_lit	True		Boolean	Street lighting
g_parkingL	{parallel}	Set	List(string)	Type of car parking
g_parkingR	{parallel}	Set	List(string)	Type of car parking
g_retailR	0.2	Mean	Float	Land-use: share of retail areas
g_severity	0.02	Mean	Float	Severity of the steepness
g_u	14304	Shared U	Int	Start node
g_v	542	Shared V	Int	End node
l_highway	{path, service}	Set	List(string)	Road type of the merged edges
l_hw_rank	{3.5, 7.0}	Set	List(float)	Road rank of the merged edges
l_old_u	{11951676149}	Set	List(int)	Start nodes of the merged edges
l_old_v	{19414683}	Set	List(int)	End nodes of the merged edges
l_osmid	{1288889984, 1288889986}	Set	List(int)	OSM IDs of the merged edges

Table 4: General edge attributes.

Each edge has various general and mode-specific attributes, outlined below.

- **Access Tags:** Each edge includes access labels for transport modes (bike, walk, motorised),

ensuring mode-specific accessibility. These are shown at the top of Table 4.

- **General Tags, Table 4:** These include geometrical details ('geometry'), lighting ('lit'), road hierarchy ('highway_rank'), and previous OSM identifiers ('OSM ID'). This base information is essential for describing the edge's and the adjacent infrastructure and for identifying it within the network. General: There is only one geometry per edge, yet the length of a mode's initial edge (before consolidation) is retained.
- **Mode-Specific Tags (Enriched Metadata), Table 5:** The edges may also retain details such as oneway status, bike lane type, or crossing availability, based on their original OSM and enriched data. This comprehensive tag structure enables targeted analysis across transport modes, making the network highly adaptable for walking, cycling, motorised travel, and public transit applications. Additional mode-dependent details are preserved as needed: maximum speed, number of lanes, incline, surface type, and any segregated paths for cyclists or pedestrians.

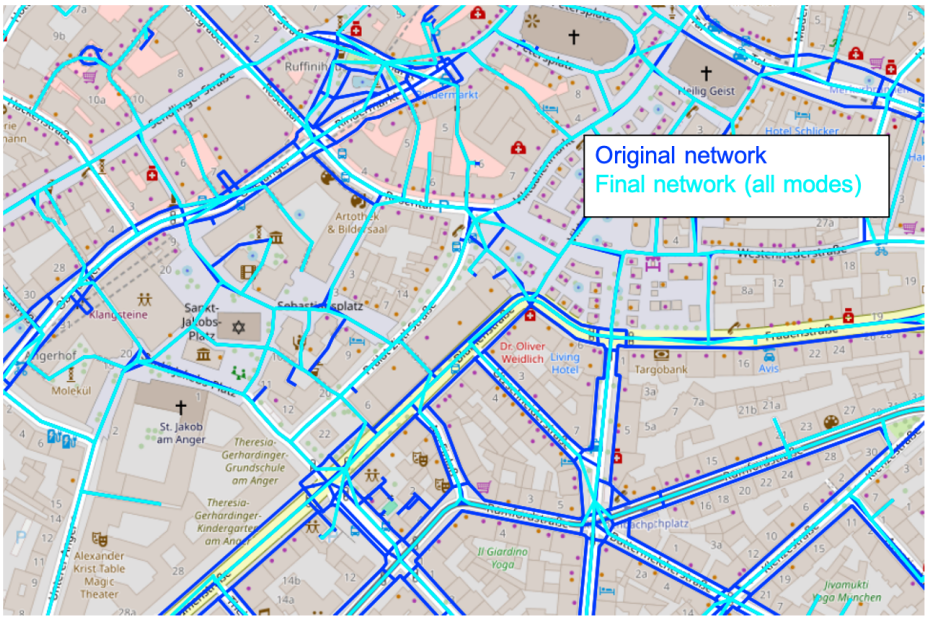
Attribute	**Example**	**Merging**	**Value Type**	**Description**
b_amntyNea	{bicycle_parking}	Set	List(string)	Bicycle amenities nearby (within 200m)
b_amntyOn	{bicycle_parking}	Set	List(string)	Bicycle amenities on the edge
b_attribut	{sidepath}	Set	List(string)	Other bike path attributes
b_bikeRoad	False	Any	Boolean	Is the edge a bicycle road?
b_bikerack	True	Any	Boolean	Is there bicycle parking?
b_category	{bicycle_road}	Set	Int	Type of cycle path
b_length	124.8	Max	Float	Length of the bicycle path
b_oneway	False	Any	Boolean	Is it one-way for bikes?
b_segregat	{yes}	Set	Int	Is the cycling infra. segregated?
b_smoothne	{excellent}	Set	List(string)	Smoothness of the bike path
b_surface	{asphalt}	Set	List(string)	Surface of the bike path
b_width	1.4	WAvg	Float	Manual input, or from OSM
m_lanes	2	WAvg	Int	Number of car lanes
m_length	124.8894	Max	Float	Length of car route
m_maxspeed	30	WAvg	Int	Speed limit
m_oneway	True	Any	Boolean	Is it one-way for cars?
m_ptRoutes	{Bus58City...}	Set	List(string)	PT routes that have a stop here
m_ptStop	3	Max	Int	Number of PT stops
m_width	5.6	WAvg	Float	Width of the road
w_length	124.8894	Max	Float	Length of footpath
w_segregat	{yes}	Set	List(string)	Is the pedestrian route segregated?
w_smoothne	{}	Set	List(string)	Smoothness of footpath
w_surface	{asphalt, sand}	Set	List(string)	Surface of footpath
w_width	2.4	WAvg	Float	Width of footpath

Table 5: Mode-specific edge attributes. WAvg: weighted average (by length).

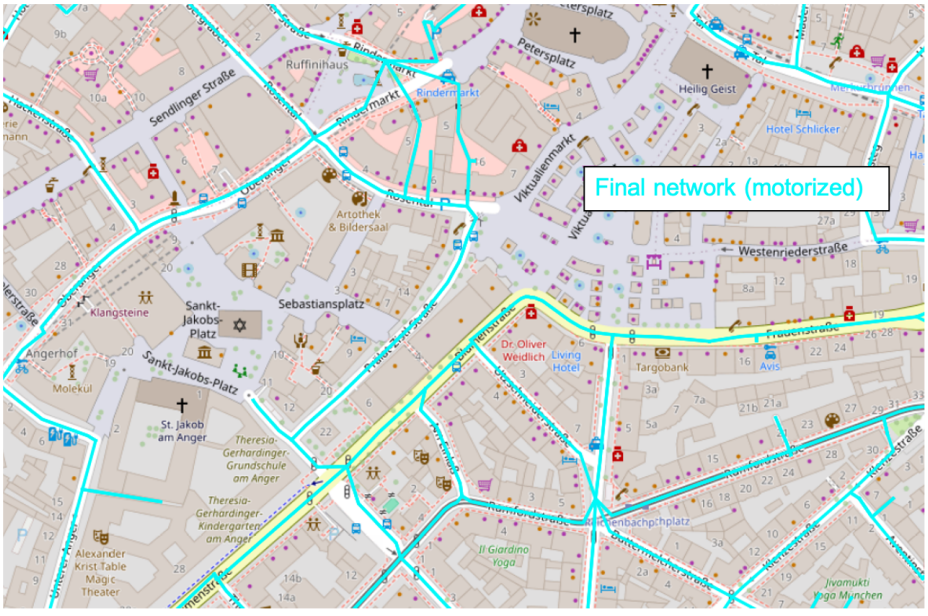
By restructuring edges and tags in this way, `OSMNetFusion` produces a simplified yet fully detailed network where each edge preserves critical information for multimodal transport analysis and connectivity modelling.

3 Examples and Usage

The reduced complexity of the topologically simplified network is depicted in Figure 3a. The simplified network is on top of the initial network, hence this visualisation highlights the edges that were removed, because they were consolidated with other edges. Figure 3b illustrated what the subnetwork for the motorised traffic looks like; it can be filtered using the *access_motorised* attribute.



(a) Before vs. after the simplification.



(b) Mode-specific simplified network: motorized traffic.

Figure 3: Simplification steps.

It is emphasised that only the city or location of interest is needed as input, as all other input data is open-source and queried automatically, if not disabled in the *configFile*.

In various test runs and previous use-cases, this framework has scaled well. Particularly the simplification has been adjusted to allow for larger networks. For instance, the entire process takes

about 20min for a dense 16km^2 network, of which the simplification takes 3min (if parallel computing is supported), and 15min for a 9km^2 area if all steps are executed sequentially.

To begin using `OSMNetFusion`, clone the repository and set your desired location in configFile. The tool can be run either as a package (demonstrated in Example_code.ipynb) or by executing runSimplification.py directly in an IDE. Additional parameters can be configured, such as whether a type of data should be included in the downloading/processing, or whether only the simplification should be carried out. The process is modular, with each step building on the previous, but steps can be skipped if they have been run previously. The package is organised into key modules, each representing a stage in the data preparation and simplification process:

- **p1_getOSMNetwork** downloads the primary OSM network and tags.
- **p1_getFurtherOSMData** collects specific OSM data such as amenities or retail areas within the specified location.
- **p1_getOtherData** allows for additional open-source data downloads, such as elevation.
- **p2_enrichData** integrates the gathered information into the OSM network, assigning attributes like bicycle amenities and land-use data within defined radii, merging columns, and categorizing cycle paths.
- **p3_simplification** consolidates the network topology by merging connected edges and retaining only essential path data across transport modes.
- **p3_functions** contains utility functions supporting the p3_simplification process.

4 Conclusion

`OSMNetFusion` offers a powerful approach to creating a streamlined yet information-rich network, which is multimodal, versatile, and well-suited for spatial network analyses, transportation planning, and mobility services. By merging different types of OSM data with external geographic data and then simplifying the network topology, this framework enables efficient, transport-mode-specific analyses.

For instance, `OSMNetFusion`'s network simplification and high information density would be particularly useful for assessing route choice of pedestrians or cyclists, which is affected by many variables. Similarly, perceived stress levels could be assessed. Furthermore, it can aid mobility service providers to strategically place stations by mapping accessibility based on factors like bike racks and elevation.

Planned enhancements include improved data consistency, better handling of bike paths, and refinements for better usability and performance. Its high information density and ability to preserve essential connectivity make it a valuable resource for planners and analysts, supporting informed decision-making in complex urban environments. Through these capabilities, `OSMNetFusion` enhances the usability of spatial data, paving the way for smarter, data-driven infrastructure and mobility solutions.

References

[1] Lukas Ballo and Kay Axhausen. "Designing an E-Bike City: An automated process for network-wide multimodal road space reallocation". In: *Journal of Cycling and Micromobility Research* (2024).

[2] Lukas Ballo and Kay Axhausen. "Modeling sustainable mobility futures using an automated process of road space reallocation in urban street networks A case study in Zurich". In: *Transportation Research Board 103rd Annual Meeting*. Poster. The National Academies of Sciences, Engineering, and Medicine. Washington, D.C., 2024.

[3] Martin Fleischmann and Anastassia Vybornova. *A shape-based heuristic for the detection of urban block artifacts in street networks*. 2024. arXiv: 2309.00438 [cs.CY].

[4] Martin Fleischmann et al. *Adaptive continuity-preserving simplification of street networks*. 2025. arXiv: 2504.16198 [cs.CY]. URL: https://arxiv.org/abs/2504.16198.

[5] Geoff Boeing. *Modeling and Analyzing Urban Networks and Amenities with OSMnx*. Working paper. 2024. URL: https://geoffboeing.com/publications/osmnx-paper/.