Preprint version

Topology-Preserving Simplification of OpenStreetMap Network Data for Large-scale Simulation in SUMO

Zhuoxiao Meng[1]*, Xiaorui Du[1]*, Paolo Sottovia[1], Daniele Foroni[1], Cristian Axenie[1], Alexander Wieder[1] , Stefano Bortoli [1] , and Christoph Sommer [2]

[1] Intelligent Cloud Technologies Laboratory
Huawei German Research Center, Munich, Germany
[2] TU Dresden, Faculty of Computer Science, Germany

[1]{name.surname}@huawei.com
[2]https://www.cms-labs.org/people/sommer/

Abstract

Converting OpenStreetMap (OSM) data to a road network usable for microscopic traffic simulation keeps being a challenging task. Often, missing information, wrong topology, and excessive details present in the dataset will confuse automatic converters. A great amount of manual work is thus typically required after the conversion, which is time-consuming and outright precludes the generation of usable very-large-scale scenarios. In this paper we present a method along with a reference implementation, Traffic Simulation Map Maker (TSMM), which aims at substantially increasing the automation level of road network prototyping by simplifying the OSM data while preserving important topology information. The main objective of this work is enable the study of traffic simulation dynamics at scale, while minimizing the need for solving the long tail of problems related to the road network generation. What is proposed is what we believe is a good trade off between precision and automation, making bold yet acceptable decisions that solve at the source, i.e. the map, most of the errors. While there is definitely a lost in accuracy, many properties of the road network are preserved. We argue that TSMM greatly improves the availability of arbitrarily large and usable road networks on top of available OSM maps lifting reducing the complexity for tools like NETCONVERT and traffic simulation researchers. A proof-of-concept study using OSM data from Binjiang, China demonstrates that TSMM is able to generate a road network with well-preserved topological information which avoids the many disconnections and deadlocks that occur when building the network using the original input sources.

1 Introduction and Motivation

Microscopic traffic simulation is a mature domain which has been studied for decades. A well-established traffic scenario is often the basis for a meaningful simulation study. However, building such a scenario is

*Zhuoxiao Meng and Xiaorui Du contributed equally to this work

often a challenging task, especially when it comes to generating the road network. Considerable effort is usually required to gather, process, and transform the collected data from different government agencies and map providers. Achieving a highly automated efficient road network manufacturing approach is thus becoming more and more pressing in recent years both for academics and traffic engineering practitioners.

At present, OpenStreetMap (OSM) [1] data is shown as a common data source for the road network generation. As a publicly available platform, OSM offers geographic data for most of the areas of the world via standardized XML format. Traffic simulators such as SUMO [2] are able to convert it into road networks with simulator readable format. However, the road networks obtained directly after the conversion often contain crucial errors and inconsistencies. Cumbersome and time-consuming manual efforts for post-processing are thus still needed which outright precludes large scale scenarios.

There are three reasons why converting OSM data directly to a road network for microscopic traffic simulation keeps being problematic. (i) The difference in the representation of road data between OSM and traffic simulation disturbs a straightforward conversion. In particular, intersections are not modeled with an explicit data structure in OSM, leads to the extraction and identification of some relevant information that is always challenging, e.g., intersection geometry, waiting lines and lane-to-lane connection. (ii) Considering most of the OSM data is produced by volunteers, errors often happen. Incorrect connections, gaps, misclassifications, and broken `ways` are common issues found in the road network for a given region [3], which could result in disconnections and inconsistencies in the traffic simulation. (iii) The lack of standardization in data representation is another reason. It is not rare to see that roads are sometimes represented by bi-directional `ways` while sometimes are two uni-directional `ways`. Turn lanes are sometime classified separately and sometimes share the same road type as the jointing streets. Although there are conventions, they are not always followed due to the sheer number of editors and covered areas. This leads to a substantial increase in the map data complexity, which is often overwhelming for traffic simulators to address.

In this paper, we present a toolchain Traffic Simulation Map Maker (TSMM)[1] with a novel approach to convert OSM data to SUMO networks. So far, researchers attempting to generate a road network for traffic simulations based on OSM data must first simplify and fill in the data manually. TSMM substantially increases the automation level of this progress. By extracting data from OSM TSMM can generate a lane-level SUMO network for any regions. In a step-by-step fashion, the complexity of the OSM data is gradually reduced by eliminating auxiliary elements. Meanwhile, important data and information is kept and processed to ensure the consistency. TSMM is also able to overcome the gaps caused by incorrect and incomplete data in original database. In the end, a clean, fully connected, and positional accurate road network can be obtained through a one-click automated process.

The goal of TSMM is to create error-free road networks at scale. This translates into a higher priority for correctness compared to realism. The generated road network retains the original road layout, road classification, and intersection locations, while complex side roads, lane connections within intersections, ramps, etc. are simplified and represented in configurable templates. This trade-off can provide researchers who is less demanding in terms of accuracy an instantly usable road network. Typical use cases include, but not limited to, i) Generating a road network, which features the typical mix of low and high traffic density roads and can therefore be used in simulation studies of Inter-Vehicle Communication [4] applications. ii) Providing large-scale road networks for researchers who are simulating traffic with High Performance Computing frameworks to evaluate the reliability and performance of relevant designs and algorithms. iii) Offering general-purpose road networks which can be used to evaluate traffic policies that are not city-specific rather target to a certain type and scale, e.g, middle size European city and densely populated Asian metropolitan areas.

On the other hand, for usage scenarios that higher details are required, we hope TSMM can accelerate the process of road network prototyping. Through future work, TSMM will be able to incrementally

[1] TSMM source code and data repository URL to be populated in Camera Ready version.

increase the realism of the road network by being compatible with other data sources.

The details of TSMM are described in the remainder of this paper.

2 Background

2.1 OSM data

OSM is a crowdsourced publicly accessible platform, which provides rich Volunteered Geographic Information (VGI) for free to use and edit [5]. Since the focus of this paper is to generate road networks for microscopic traffic simulation, only the following elements regarding the road network in OSM are relevant: `node`s, `way`s, and `relation`s. They are briefly described in the following.

Nodes: A `node` is the basic element in the OSM data model. Each represents a single point and is defined by a coordinate. Usually each `node` should have a unique `node` id.

Ways: A `way` is the representation of (part of) a street. A `way` consists of several `node`s, which define the shape of the street. Each `way` is characterized by a set of tags, defining the typology and specifications of the street. These tags show, among the others, the road level (e.g., `motorway`, `primary`, `trunk`) and the number of lanes. Another important tag is `oneway`. When a `way` is tagged as `oneway = yes`, the `way` is then uni-directional and only allows vehicles to travel from its first `node` to the last one, conversely is then prohibited. Besides real one-way road in small districts, dual carriageways are also often marked with this tag and then represented by two approximate parallel uni-directional `way`s, each in an opposite direction. However, this will cause problems and confuses the conversion tools when generating a road network for traffic simulation. We will dive in the details in section 2.2.

Relations: Regarding road network, a common use of `relation`s in OSM is to define the access restrictions for connected `way`s. In OSM data, `way`s are connected by default when they share the same `node`s. It means vehicles are allowed to travel from one `way` to another via their common `node`s if there are no other rules applied. In order to have additional access restrictions, users can define specific rules for `node`s and `way`s in `relation`s, such as `no_right_turn` from one `way` to another, `no_u_turn` at a junction, and so on.

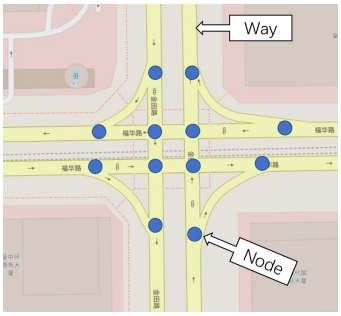
An example illustrating a junction represented by OSM data can be found in figure 1a;

2.2 Standard generation of SUMO network

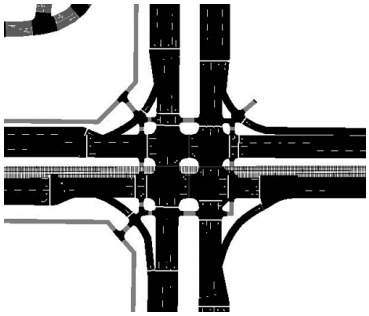
Similarly, SUMO network also consists of nodes and edges representing streets, bike lanes and walkways [2]. In order to generate SUMO networks based on OSM data, SUMO provides users several internal applications, such as NETCONVERT, OSMWebWizard and NETEDIT. Among them, NETCONVERT and OSMWebWizard are both able to convert data in OSM format to SUMO network, while NETEDIT is a graphical network editor that can be used to manually fill in missing information and correct errors in the converted network files.

The problem of converting OSM data to SUMO network is to identify the association of nodes to intersections. As shown in figure 1a, one intersection could consist of multiple `node`s and `way`s, particularly when the `way`s are modeled by two parallel uni-directional roads with opposing directions. Such a design is suitable for demand for navigation or routing services, however, it does not provide any benefit for converting to SUMO networks.

[2]Map data and OSM tiles © OpenStreetMap contributors; terms: www.openstreetmap.org/copyright



(a) original OSM data[2]



(b) SUMO map after conversion; multiple intersections are generated instead a complete one

Figure 1: Representation of a intersection in OSM and the result of converting it to SUMO without simplification.

To be able to form this intersection correctly, SUMO network converters like NETCONVERT must be able to identify which `nodes` and `ways` in OSM are just normal streets, and which are actually a part of an intersection. Incorrect identification will lead to a result as is shown in figure 1b, where a complete intersection is interpreted wrongly as multiple sub-intersections. Consequently, deadlocks can occur during the simulation, as is shown in figure 12a. Such a problem can also be confirmed in work by Codecá et al. [6] and Rapelli et al. [7] regarding to the generation of SUMO Luxembourg and Turin scenario, and to correct the intersections, researchers have to edit the network in a manual way.

3 Related Work

Since SUMO already provides users with NETCONVERT, which can generate a road network after inputting the OSM map, and since, as we analyzed above, the real problem that causes poor conversion is the complexity of the OSM data and redundant auxiliary elements (e.g., `nodes` conceptually form an intersection), NETCONVERT should be able to generate a clean road network directly if the OSM data is simplified in advance.

Similar studies have been done by many other researchers. Some studies [8], [9] suggest that nodes which represent the same intersection can be integrated into one through their geometric connection relations, such as the spacing among the nodes, and whether they can form a circuit or not. Such an approach can usually only handle intersections consisting of only a small number of nodes, and it is powerless for complex intersections, especially those with extra turn lanes. Other researchers present methods to use the semantic information in the OSM data to help locate intersections [4]. However, this is highly dependent on the quality of the OSM data being used. There are also many studies [10], [11] that are looking to capture parallel dual carriageways in OSM data through machine learning. However, the focus of these studies is to identify multi-lane roads for road hierarchy analysis rather than traffic simulation. The integration of multi-lane roads, the processing of single-lane roads, the identification and representation of intersections for traffic simulation are not in the scope of these studies.

In this paper, we present a novel approach that is applicable to a wide variety of cases to simplify the OSM data with a strong emphasis on generating an error-free and consistent road network. This is done by firstly merging multiple OSM `ways` (in particular those trying to approximate a median strip

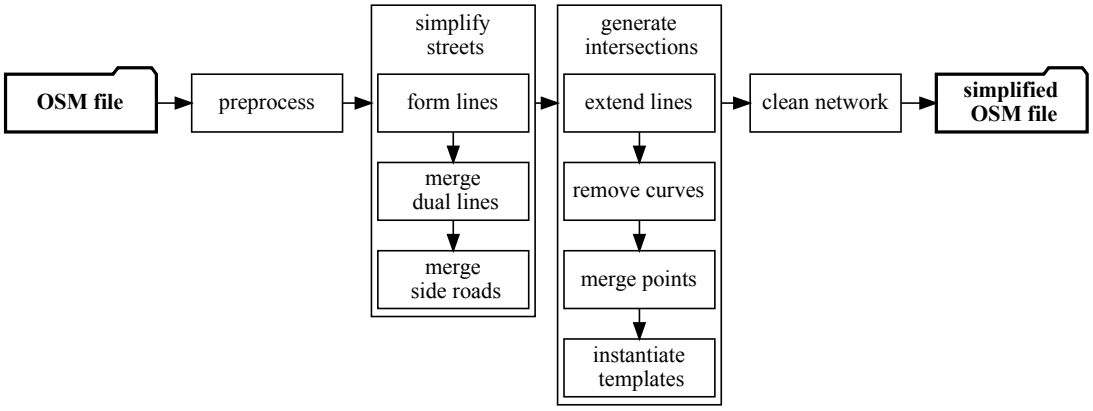


Figure 2: Steps performed by TSMM.

separating opposing lanes of the same road) into individual `ways` using a buffer method to represent the streets, and then using the crossing points among all the simplified `ways` to represent the intersections. In the end, the resulted simplified OSM network should include only streets and intersections, which can be easily converted to a SUMO network by NETCONVERT, as discussed before.

The methodology is detailed in the following section.

4 Methodology

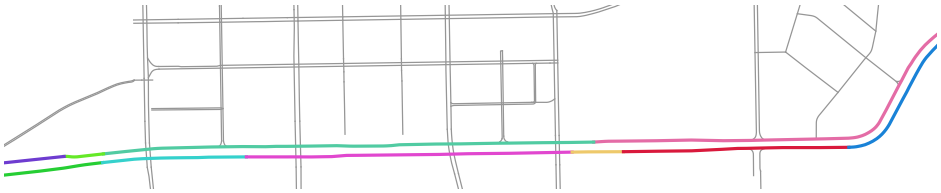
Starting from OSM data, TSMM comprises a sequence of processing phases for generating a road network for a given region. We detail the processing phases in the following. Figure 2 gives an overview.

4.1 Pre-Processing

At present, the emphasis of TSMM is solely generating road networks for motor vehicles. Roads used primarily by other traffic participants, such as bicycle lanes, crosswalks, and bus lanes, are beyond the scope of this work. Therefore, the raw OSM data is first pre-processed by filtering irrelevant elements out. In particular, roads of type `motorway`, `trunk`, `primary`, `secondary`, `tertiary`, and `residential` are extracted from the raw OSM data, while other types are disregarded.

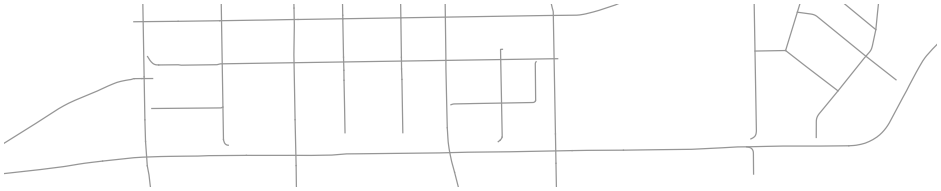
4.2 Simplifying streets

The first step of simplification is to form `lines`. Note that a `line` is not a data structure available in OSM. In this paper, a `line` is used to represent an ordered set of `ways`, these `ways` have the same road type and are connected to adjacent `ways` within the group trough their first or last `node`. Forming `lines` is straightforward, starting from an arbitrary `way`, searching forward and backward for its adjacent `ways`, if the angle between them is less than a certain value, then adding them in an ordered data structure to form the `line`. One notable situation is that it is possible for a `way` to have multiple other `ways` connected at its end. In this case, the `way` within all the connected `ways`, which has the smallest angle with the `line` will be selected and added henceforth to be the next element of the `line`. In this manner, the overall



(a) illustration of how a long road with a median strip separating the opposing lanes is commonly modeled in OSM; different colors illustrate how the road is composed of multiple different OSM `way` elements

(b) intermediate result after forming `lines`; different colors illustrate different `lines`



(c) result after merging dual `lines`

Figure 3: Simplification of `ways` in OSM data.

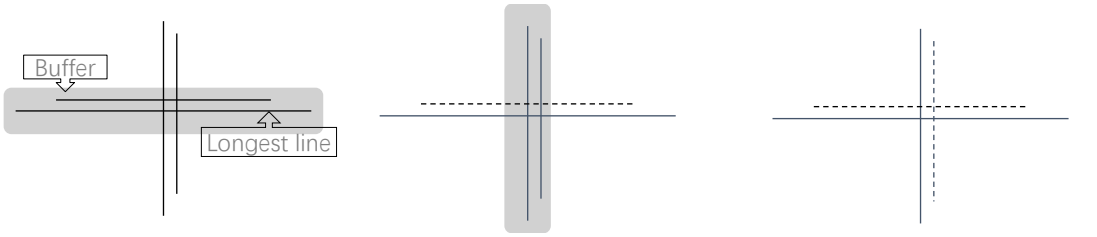
shape of the `line` will not change dramatically. A result after the forming `line` process is shown in figures 3a and 3b.

Two approximately parallel `lines`, as are colored in figure 3b, often represent a dual carriageway in the real world. If these dual `lines` are all identified and merged into one single `line`, the raw data can be hence substantially simplified. It is not difficult to find that the two `lines` in the dual `lines` are usually not too far apart, since they are representing the same street. For the same reason, their shape, the alignment should also be relatively similar. Therefore, a buffer approach is then proposed in this paper to merge the dual `lines`, as shown in figure 4.

Algorithm 1 Buffer Algorithm

Input: set of all `lines`, $L = \{l_0, l_1, ...\}$
Output: set of remaining `lines`, $R = \{\}$
  while L is not empty **do**
    Pick longest line l from L;
    Add l to R;
    Generate buffer polygon based on l;
    Remove any line from L of which at least 70% length is inside the buffer;
  end while

  return R

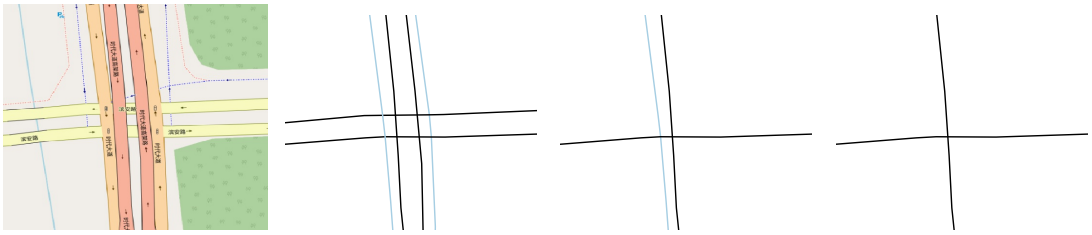


(a) after creating a buffer based on the longest `line`

(b) after deleting `lines` within the buffer and creating the next buffer; removed `ways` illustrated as dashed lines

(c) after merging all dual `lines`; removed `ways` illustrated as dashed lines

Figure 4: Buffer method for merging dual `lines`.



(a) rendering of side roads in OSM map[2]

(b) representation of side roads (blue) and main roads (black) with `lines`

(c) after merging dual `lines`

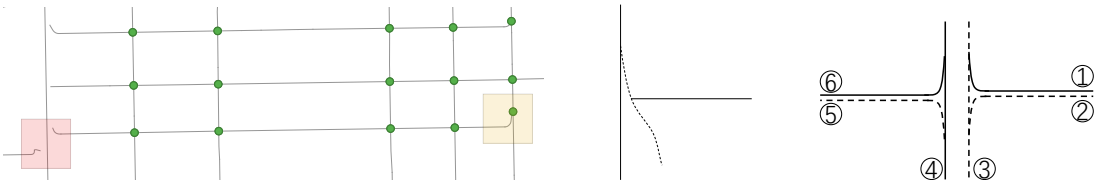
(d) after merging side roads

Figure 5: Process of merging side roads into a main road.

The term *buffer* refers to a polygon with a fixed width which is generated around a `line`. First, such a buffer will be generated based on the longest `line` l in the current map data. All other `lines` which have the same road type as l and are covered by the buffer over a certain percentage with respect to the length will be then deleted in the map data. The reason is that it is hypothesized that `line` l should be able to represent all the deleted `lines` within the buffer. Consequently, `line` l will be then tagged as a bi-directional street. The amount of its lanes is thus doubled to ensure a consistent road capacity as before. Next, the buffer is generated starting from the longest of the remaining `lines`, and the process is repeated until all `lines` are processed. As shown in figure 3c, all the dual `lines` in the original OSM data are identified and are represented with one single `line` now. As can be seen, the resulting network has a similar roads layout compared to the original map while its complexity is substantially reduced.

One remaining problem are side roads of a different road type than the main road (figures 5a and 5b). As figure 5c shows, side road and main road are still represented by two approximate parallel `lines` after the buffer approach if only roads of the same type are considered. To solve this, the buffer method is thus applied again – the only slight difference being that, this time, the road merge process also considers any road which has a "lower" type (e.g., `secondary` vs. `primary`) than a given road as being eligible for merging into the present road. In order to ensure the consistency of the data, the number of lanes of the main roads is increased with the corresponding lanes amount of side roads, same as the approach taken by Wang et al. [8].

At this point, the simplification for streets is completed. Next, we focus on the corresponding operations for generating the intersections.



(a) generation of intersections after simplifying streets; the left-hand rectangle marks an area where a disconnection occurs (figure 6b); the right-hand rectangle marks an area where the merging process created a sharp right-turn curve (figure 6c).

(b) disconnection caused by missing filtered out `ways`

(c) disconnection and sharp turn curve caused by simplifying streets

Figure 6: Illustration of issues for generating intersections.

4.3 Generating intersections

As shown in figure 6a, after the last step, streets are now all represented by only one `line`. In this case, the crossing points (green points) among the remaining `lines` are naturally the location of the intersections in real world. Simply calculate the coordinates of these crossing points and insert them as `nodes` into the corresponding `ways`, and the intersection detection and generation process should be then completed. While this approach is feasible in most cases, additional processing is required for certain issues, as explained below.

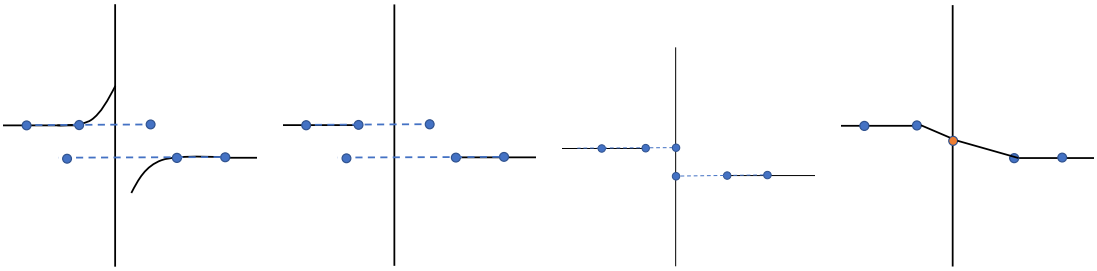
Several streets are disconnected after the previous simplification, as is highlighted with the bottom left rectangle in figure 6a.

One case of disconnections is shown in figure 6b. In this case, the `way` (dashed line) which connects the other two `ways` was filtered out during the data pre-processing because it is not tagged with an appropriate road type (e.g., `unclassified`) or tagged as auxiliary elements (e.g., `link`). This caused the break of those two originally connected `ways`. Such an issue occurs more frequently when data is less complete and accurate.

The other case of disconnection happens after the dual `lines` merging process, as is illustrated in figure 6c. Originally the `line` 1 is connected with `line` 3. However, `line` 3 is merged into `line` 4 as both `lines` jointly represent the dual street. This leads to the disconnection between `line` 1 and the dual street in the simplified road network.

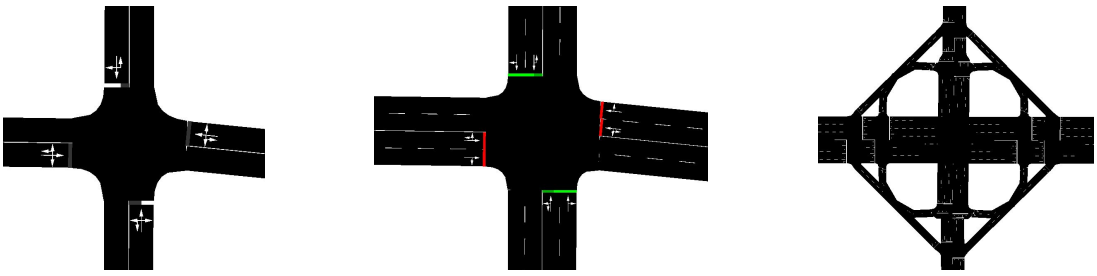
Apart from the disconnection, the unreasonable intersection shapes (bottom right rectangle in figure 6a) caused by the simplification process also needs to be considered. The reason for this issue is shown in figure 6c as well. After `line` 5 being merged, `line` 6 became the complete dual street. Unlike `line` 1, the connection between `line` 6 and the dual street represented by `line` 4 and `line` 3 is still ensured. However, in the original data, `line` 6 has a left-turn curve. If the turn curve is directly converted to a bi-directional in the SUMO network, an intersection with unreasonable shape will be thus generated. Right-turning vehicles driving from `line` 6 to `line` 4 are confronted with a sharp turn angle. It can lead to severe congestion in the simulation that does not apply in the real world.

Therefore, there is an automatic pipeline in TSMM, which aims to solve the issues mentioned above. Firstly, all the `lines` are extended by a certain length at the head and tail ends. If a curve is detected there, the extension will be based on the overall angle of `line` and the curve will be subsequently removed. By doing this, the `lines` remaining in the map will reconnect to the others, crossing points are thus able to be formed to represent the intersections. Next, among the crossing points formed by the extended `lines`, those very closely spaced points can be further merged together, and the extended excess will then be trimmed off as well (figure 7). In this manner, intersections that were not able to be captured previously can be created, and sharp turn curves can be straightened out to form normal streets.



(a) extending roads along main direction (b) removing curves near the intersection (c) creating nodes at the intersection (d) merging nearby nodes

Figure 7: Steps in the `line` extension process. TODO: pseudo-code as well



(a) uncontrolled intersection (b) traffic light controlled intersection (c) grade separated intersection

Figure 8: Classification of intersections.

Regarding the representation of intersections, TSMM provides three different types with templates depending on the level of its connected roads.
TODO: reference algorithms from text via LaTeX label

Algorithm 2 Lines Extension Algorithm

Input: Lines' set L
Output: Extended lines' set R
  while L is not empty **do**
    Pick a random line l from L;
    Extend a length of d_1 along the main direction of l at the head and tail ends;
    if Curves are detected at the head or tail ends of l **then**
      Remove curves of l;
    end if
    Add l to R;
    Remove l from L;
  end while

  return R

Uncontrolled intersection (figure 8a) In the real world, these intersections are regulated only by right-of-way rules. They are found most in the crossing points among low-level streets, like residential `ways`. For this type of intersections, no additional operations have to be applied in TSMM. When a `node` is shared with more than two `ways`, by default this is converted to an uncontrolled intersection with NETCONVERT.

Traffic light controlled intersection (figure 8b) Besides, there are intersections which requires traffic lights to resolve traffic conflicts. The standard identification of traffic light controlled intersection in NETCONVERT is through the information provided by the tags of the `nodes`. For instance, if a `node` is tagged with `traffic_signals`, the intersection formed by this `node` will be converted to a traffic light controlled intersection. Standard traffic light programs will also be provided by NETCONVERT by default, if there are no additional customized data filled in by the users. With TSMM, users can first define which level of intersections should be controlled by traffic lights. TSMM is able to identify and tag the `nodes` correctly to ensure a proper conversion later.

Grade separated intersection (interchange) (figure 8c) For streets like `motorway` and `highway`, it is necessary to separate the traffic in the vertical grade, in order to maintain the high driving speed of vehicles and maximize traffic throughput. As mentioned earlier, users can first define which type of street should be served with interchange, then TSMM will treat the intersections locating in these streets as interchanges. As a template, so far TSMM uses clover leaf interchanges to represent all the grade separated intersections.

4.4 Network cleaning

Full connectivity plays an important role in SUMO network.

After simplifying the road network from the previous steps, TSMM uses the Breadth-first search (BFS) algorithm to group all the intersections into different clusters according to their connectivity.

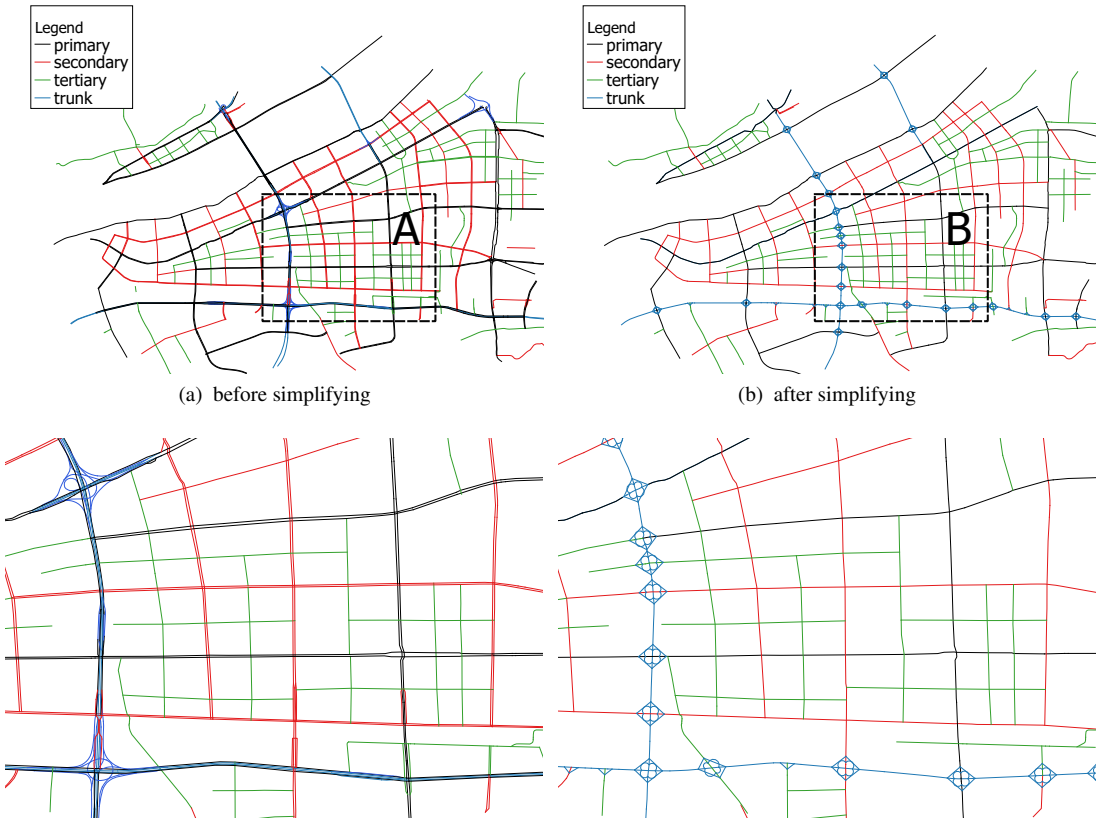
Intersections within the same cluster must be connected, while different clusters are separated from each other.

Then, the cluster with the highest number of intersections will be defined as the main cluster, and intersections and streets which are not in it will be removed from the road map. The proportion of the

Algorithm 3 Intersection Merging Algorithm

Input: Lines' set R
Output: Lines' set M after merging intersection points
  $M = R$;
  while R is not empty **do**
    Pick a random line r_2 from R;
    Calculate all intersection points p of r_2 and other lines;
    Add p to Intersection points set P;
    Remove r_2 from R;
  end while
  Calculate intersection clusters $\{c_1, c_2, c_3 \dots\}$ of set P according to a distance threshold d_2;
  Calculate average position of each intersection cluster and get point set $\{avgP_1, avgP_2, avgP_3 \dots\}$;
  Replace intersection points in M with the corresponding average position points.

  return M



(a) before simplifying

(b) after simplifying

(c) zoomed view of figure 9a

(d) zoomed view of figure 9b

Figure 9: Comparison before and after simplifying.

number of intersections in the main cluster to the total number of is considered as an indicator of road network connectivity, so far.

If this ratio is too low, TSMM will throw warning to the users and suggest reconsidering the parameters being used.

5 Evaluation

The purpose of this experimental case study is to evaluate the performance of our tool using real OSM map data. The road network in Binjiang, a central district of Chinese city Hangzhou which covers 72.2 km^2 in total was tested. As shown in figure 9, the original OSM network mainly consists of dual uni-directional roads. Based on the proposed methodology, TSMM is able to simplify it and accurately retain the topological information with respect to location of intersections, overall roads layout, and road classifications.

It is worth noting that in the original network, `primary` (in black) roads appear in several places as the side roads of `trunk` roads (in blue). That means, the vehicles driving on the `trunk`s should not have conflicts with traffic from other roads, the connection among them should be achieved through

Figure 10: SUMO network of Binjiang.

the `primary` side roads. However, the design of these side roads is often so complicated and non-standardized that converting them into a SUMO network directly through NETCONVERT rarely succeeds. A poor identification often even leads to broken roads and deadlocks. Therefore, as described in section 4, TSMM merged side roads into main roads and inserted clover leaf interchanges for all the intersections on the merged roads during the simplification process. In this manner, although the layout visually differs from reality, it still ensures as much as possible the topology of the road network regarding traffic characteristics. With respect to road capacity, driving speed and overall connectivity, the simplified road network will keep consistent with original network, and most importantly, it can be easily converted to a SUMO version for traffic simulation.

As a result, the so simplified road network contains 161 km roads and 128 intersections, among which 28 are grade separated intersections.

The simplification was executed on a mid-range Desktop machine operating at 3.90 GHz. The current execution time for Binjiang road network is 21 seconds, and it is believed that this performance can continue to improve after code optimization.

5.1 Experimental settings

After the simplification, the simplified OSM road network is fed into NETCONVERT henceforth to be converted to a SUMO readable network file (figure 10). For comparison, the non-simplified OSM road network is converted as well.

NETCONVERT itself contains many parameters and switches; for a comprehensive comparison, we converted the non-simplified road network with two different versions. The first version, same as converting the simplified OSM network, is converted with default options, as follows:

```
netconvert --osm-files <osm-file> -o <sumo-network-file>
```

For the second version, it is converted with additional options which are frequently recommended[3] for NETCONVERT:

[3] https://sumo.dlr.de/docs/Networks/Import/OpenStreetMap.html

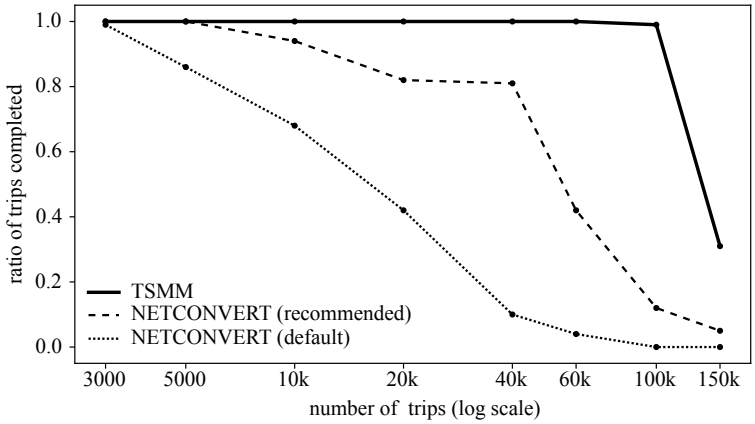


Figure 11: Rate of completed trips vs. increasing travel demand. Different road networks can sustain differently high demand.

```
netconvert --osm-files <osm-file> -o <sumo-network-file> --geometry.remove
    --ramps.guess --junctions.join --tls.guess-signals --tls.discard-simple
    --tls.join --tls.default-type actuated
```

To summarize, conversion with these additional parameters are able to simplify the raw OSM data automatically, join the nodes heuristically, guess location of ramps and insert traffic lights for uncontrolled intersections. In general, the network generated with the recommended options offers a higher quality compare to the default one. But for many complex networks, it still cannot handle either.

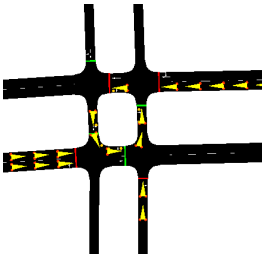
The traffic demand for the experiments need to be prepared at the beginning. To ensure the consistency across all scenarios, the trips files for each network are taken from the same dataset. In the dataset, a certain amount of trip data is stored, where each trip consists of a coordinate pair within the study area, and a randomly assigned departure time within the time interval 0–180 min. In the next step, the trip data are converted to SUMO readable trips files for each network. Origin and destination places are giving by finding the closet edges to the coordinates and the departure time is identical as in the dataset.

The simulation time is set to 6 hours, which is sufficient for all the vehicles to finish their trip if there is no congestion. Teleport trip is forbidden unless a collision has occurred or no valid route has been found. At the end of each simulation, by analyzing the car counts statistics and using visualization tool to view the bottlenecks of the road network, the performance of each road network can be assessed.

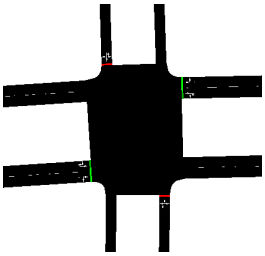
5.2 Network assessment

As is illustrated in figure 11, by continuously increasing the initial amount of traffic, the tested road networks start to show uncompleted trips when the traffic volume reaches to 5000, 10 000, and 100 000 trips, respectively. At the end of the simulation, the bottlenecks of the road networks could be observed by viewing the location of the stranded vehicles. The analysis of the causes of bottlenecks allows to determine whether the congestion is caused by excessive traffic volume or by errors in the road network itself. A comparison of the three different tested road networks with respect to the correctness will be detailed in this section.

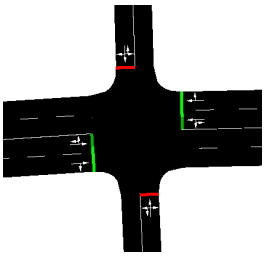
The results after the simulation reveal that the non-simplified sumo road network generated using the default settings has the lowest traffic capacity compared to others. When 5000 cars were imported into



(a) a representation of congested intersections at the end of simulation; converted via default settings

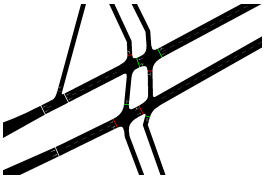


(b) same intersection as figure 12a converted via recommended settings

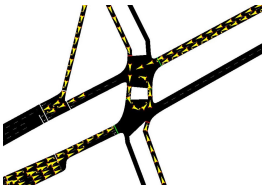


(c) same intersection as figure 12a converted via TSMM

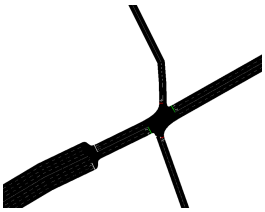
Figure 12: Analysis of uncompleted trips in the road network converted directly via default settings with 5000 initial trips.



(a) same intersection as figure 13b converted via default settings



(b) a representation of congested intersections at the end of simulation; converted via recommended settings



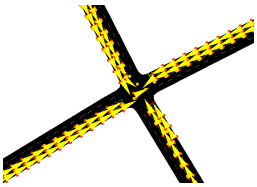
(c) same intersection as figure 12b converted via TSMM

Figure 13: Analysis of uncompleted trips in the road network converted directly via recommended settings with 10 000 initial trips.

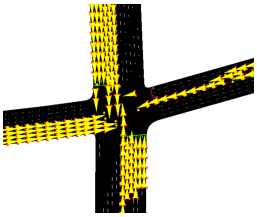
the network as input demand, nearly 400 cars were already unable to complete their trip. Congestion occurs at many intersections at the end of the simulation. Through observing one of their representatives (figure 12a), it is revealed that the congestion was caused not by excessive traffic, rather by errors in the road itself. In default, NETCONVERT treats all `node`s crossing different `way`s in OSM data as intersections, four intersections each with an independent traffic light controller are hence created here. Consequently, as shown in figure 12a, when an excessive number of vehicles flood into this complex multi-intersections area, due to traffic lights and narrowed spaces vehicles block each other. A deadlock is thus created that obstructs all vehicles that are supposed to pass by. On the contrary, same intersection generated via recommended parameters (figure 12b) and via TSMM (figure 12c) are similar to the real world shape. Traffic capacity is hence substantially increased in both.

The road network, converted with the recommended parameters via NETCONVERT shows a remarkable improvement in the deadlock discussed above. The algorithm it used can cope with the simple cases when multiple `node`s forming one intersection in OSM data, but it still lacks the ability to address sophisticated situations. At the end of the simulation with road network generated via recommended parameters and 10 000 initial demands, the unresolved traffic congestion is present across several intersections. Among them, intersection as shown in figure 13b explains the reason for the congestion. Part of the `node`s in this

(a) bottlenecks of network converted via TSMM

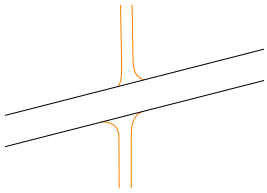


(b) congested intersection in left-hand of figure 14a; converted via TSMM

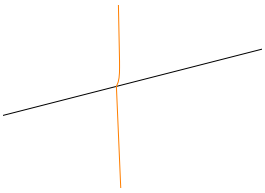


(c) congested intersection in right-hand of figure 14a; converted via TSMM

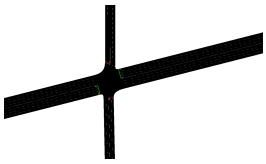
Figure 14: Analysis of uncompleted trips in the road network converted directly via TSMM with 100 000 initial trips.



(a) original OSM data: Straight ahead is not allowed in highlighted streets.



(b) simplified OSM data: No turn restriction is applied in highlighted (yellow) streets.



(c) converted SUMO network: straight ahead is allowed in each entry.

Figure 15: At the moment, TSMM is incapable of identifying turn restrictions.

widely spaced intersection are clustered together and two independent traffic light controlled intersections are hence generated. Although better than figure 13a, which has four, deadlock still happens when a large amount of traffic flowing into here. On the contrary, network produced via TSMM consists of only one intersection as expected, as is shown in figure 13c.

6 Limitations

This section discusses the main limitation of the present approach. Follow-up research will be carried out to further address these issues and to continuously improve our proposed tool.

6.1 Turn restrictions

Currently, TSMM is still lacking in the determination of turning restrictions. As described, the simplified road network, is by default fully connected at every intersection, i.e., straight ahead, right turn, left turn and U-turn are allowed in the entry of each directions. However, this is not always the same case in the real world. For instance, figure 15a shows an intersection in original OSM data, where traffic on the highlighted road is restricted to right and left turns only, straight ahead is in fact not allowed. In the simplified road network generated via TSMM, however, this complex intersection is replaced with one crossing point (figure 15b). Consequently, the truing restrictions are lost in the final generated SUMO network, as shown in figure 15c. By allowing vehicles to drive straight ahead along the highlighted street, the connectivity of the simplified road network differs to some extent from the original data we have.

6.2 Roundabouts

The method proposed in this paper is not applicable to roundabouts yet. In OSM data, a roundabout is always a closed circular loop consisting of one or several ways. Thus, when forming the lines, TSMM will keep traversing the same ways of a roundabout, leading to a dead loop in the simplification process. The current interim approach to avoid the dead loop is to identify and filter out the roundabout first during the pre-processing phase, and fill in them back after the step of simplifying dual streets. However, the roundabouts are often cut by extended lines when generating the intersection points. In the future, a specific algorithm for the roundabouts should be proposed. Not only the roundabouts themselves, but also the ways that are connected to the roundabouts should be identified and treated exceptionally.

6.3 APIs for extensibility

TSMM is hoped not be limited only to OSM data. To meet the different needs of map detail, more APIs will be developed to couple with other data sources. Speed limit, lane count, layout of intersections, traffic lights can thus all be configured in an incremental way within the pipeline, if additional data sources and corresponding APIs for processing them are available. In this way, the output obtained from the TSMM is not just the synthetic prototype it has been so far, but can approach a realistic road network. By leveraging validated scenarios such as SUMO Luxembourg [6], further evaluation of the generated road network can be performed. The comparison of metrics such as travel-times and travel-distances will reveal, how close the one-click automatically generated road network via TSMM is to a real one prepared over several months.

7 Conclusion

In this paper, we developed a tool Traffic Simulation Map Maker (TSMM) to simplify OpenStreetMap (OSM) data for converting to generate SUMO traffic road networks. By using a buffer method to merge the dual `lines` which formed with connected `ways` and subsequently integrating the side roads and main roads, all the streets in the road network are solely represented by single bi-directional `line`, respectively. In further, the identification of intersections are processed by locating the crossing points of remaining lines. Through the use of standardized templates of intersections, the lane-level connectivity in each intersection is always promised. In addition, TSMM provides three different types of intersections with regard to various road classifications, which can guarantee the corresponding traffic characteristics to reproduce the realistic traffic patterns in the simulation. After checking the connectivity of the road network and eliminating the disconnected parts, the generated road network must be a fully connected road system and no invalid routes should occur in the simulation.

By evaluating the conversion results with a real case study, it can confirm that TSMM is able to simplify an OSM data and preserve its important topological information. Compared to the non-simplified OSM data, the SUMO network generated from the simplified version effectively avoids deadlock situations and thus greatly increases the overall allowable traffic capacity.

Our goal is to reduce the pain of generating a road network while retaining all the important information as much as possible. In general, OSM maps do not contain the right level of details for the generation of a good road network in SUMO or any other simulator. A great amount of guessing is required, and at scale the number of errors to be manually verified and corrected can be overwhelming and expensive. For users who may lack of other data resources, TSMM will help them generate a traffic road network from scratch. For those who require higher accuracy, hopefully TSMM can be their first step in generating a prototype as an easy start, thus reducing their investment in the entire road network generation process.

References

[1] J. Bennett, *OpenStreetMap*. Packt Publishing Ltd, 2010.

[2] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Nov. 2018. DOI: 10.1109/ITSC.2018.8569938.

[3] F. Tabet et al., "OSMRunner: A System for Exploring and Fixing OSM Connectivity," in *22nd IEEE International Conference on Mobile Data Management (MDM 2021)*, IEEE, 2021, pp. 193–200. DOI: 10.1109/MDM52706.2021.00039.

[4] C. Bewermeyer, R. Berndt, S. Schellenberg, R. German, and D. Eckhoff, "Poster: cOSMetic – towards reliable OSM to sumo network conversion," in *2015 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2015, pp. 151–152. DOI: 10.1109/VNC.2015.7385562.

[5] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008. DOI: 10.1109/MPRV.2008.80.

[6] L. Codecá, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 52–63, 2017. DOI: 10.1109/MITS.2017.2666585.

[7] M. Rapelli, C. Casetti, and G. Gagliardi, "TuST: from Raw Data to Vehicular Traffic Simulation in Turin," in *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, IEEE, Oct. 2019. DOI: 10.1109/DS-RT47707.2019.8958652.

[8] H. Wang, J. Wang, P. Yu, X. Chen, and Z. Wang, "Extraction and construction algorithm of traffic road network model based on OSM file," in *2021 4th International Conference on Advanced Algorithms and Control Engineering (ICAACE 2021)*, vol. 1848, Sanya, China: IOP, Jan. 2021, p. 012 084. DOI: 10.1088/1742-6596/1848/1/012084.

[9] T. Ziemke and S. Braun, "Automated generation of traffic signals and lanes for MATSim based on OpenStreetMap," *Elsevier Procedia Computer Science*, vol. 184, pp. 745–752, 2021. DOI: 10.1016/j.procs.2021.03.093.

[10] Y. Xu, Z. Xie, L. Wu, and Z. Chen, "Multilane roads extracted from the OpenStreetMap urban road network using random forests," *Wiley Transactions in GIS*, vol. 23, no. 2, pp. 224–240, Dec. 2019. DOI: 10.1111/tgis.12514.

[11] Q. Li, H. Fan, X. Luan, B. Yang, and L. Liu, "Polygon-based approach for extracting multilane roads from OpenStreetMap urban road networks," *Taylor & Francis International Journal of Geographical Information Science*, vol. 28, no. 11, pp. 2200–2219, May 2014. DOI: 10.1080/13658816.2014.915401.