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

Zhuoxiao Meng[1,2*], Xiaorui Du[1,2*], Paolo Sottovia[1], Daniele Foroni[1], Cristian Axenie[1], Alexander Wieder[1], David Eckhoff[2,3], Stefano Bortoli[1], Alois Knoll[2], and Christoph Sommer[5]

[1] Intelligent Cloud Technologies Laboratory, Huawei Munich Research Center, Munich, Germany
[2] Department of Informatics, Technical University of Munich, Munich, Germany
[3] TUMCREATE, Singapore, Singapore
[4] Nanyang Technological University, Singapore, Singapore
[5] TU Dresden, Faculty of Computer Science, Dresden, Germany

✉ Zhuoxiao Meng zhuoxiao.meng@huawei.com
✉ Xiaorui Du xiaorui.du@huawei.com
✉ Christoph Sommer https://www.cms-labs.org/people/sommer/

Abstract

Converting OpenStreetMap (OSM) data to a road network suitable for microscopic traffic simulation keeps being a challenging task: both missing information and excessive details, as well as wrong typologies present in the dataset frequently confuses automatic converters. 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 to enable the study of traffic simulation dynamics at scale using real-world road networks, while minimizing the need for solving the long tail of problems related to the road network generation. Our proposed approach yields what we believe is a good trade-off between precision and automation, making bold yet acceptable decisions that solve most of the errors at the source, i.e., the map. While there is definitely a loss in fidelity with respect to the real world, 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 by reducing the complexity for conversion tools and traffic simulation researchers alike. 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 errors and deadlocks that occur when building the network using the original input sources.

*Zhuoxiao Meng and Xiaorui Du contributed equally to this work

1 Introduction and Motivation

Microscopic traffic simulation is a mature domain which has been studied for decades [1]–[3]. A well-established traffic scenario is often the basis for a meaningful simulation study. However, building such a scenario is, in most of the cases, 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 production is thus becoming more and more pressing in recent years both for academics and traffic engineering practitioners.

At present, OpenStreetMap (OSM) [4] data is a common data source for the road network generation to support researchers and practitioners in generating proof of concept scenarios. As a publicly available platform, OSM offers geographic data for most of the areas of the world in a standardized XML format. Traffic simulators such as SUMO [5] are able to convert it into simulator-native road network formats. However, the road networks obtained from a direct conversion of OSM data often contain crucial errors and inconsistencies due to ambiguous and often erroneous long tail of details and corner cases. Hence, cumbersome and time-consuming manual efforts for post-processing are thus still necessary, outright precluding the generation of large scale scenarios.

There are three reasons why converting OSM data directly to a road network for microscopic traffic simulation is problematic: (i) Intersections are not modeled with an explicit data structure in OSM, requiring the conversion process to guess relevant information like intersection geometry, waiting lines, and lane-to-lane connections[1]. (ii) Considering that most of the OSM data is produced by volunteers, human errors are very frequent. Incorrect connections, gaps, misclassifications, and broken `ways` are common issues found in the road network for a given region [6], which could result in disconnections and inconsistencies in the traffic simulation. (iii) Variability in how real world geometry is represented in OSM data is another reason. Although there are conventions, they are not always followed due to the sheer number of editors and covered areas. For example, some bi-directional roads are represented by a single bi-directional `ways`, while others by two separate uni-directional `ways`. Turn lanes are sometimes classified separately and sometimes they share the same road type as the joining streets. Such variability leads to a substantial increase in map data complexity and further aggravates the challenges of automatically converting OSM data to a road network suitable for microscopic traffic simulation.

In this paper, we present the Traffic Simulation Map Maker (TSMM)[2] toolchain that follows 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 process. 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. At the same time, most important data and information is kept and processed to ensure the consistency. TSMM aims to support researchers in overcoming the challenges caused by incorrect and incomplete data in original OSM map data. The road network generation process proposed in this work allows to produce a fully connected and positionally accurate road network 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. Typical use cases of such simplified networks include, but are 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

[1] https://wiki.openstreetmap.org/wiki/Junctions

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

Inter-Vehicle Communication [7] 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 and rather target a certain type and scale, e.g., a middle size European city or a densely populated Asian metropolitan area.

For usage scenarios which require higher details, on the other hand, we hope that TSMM can accelerate the process of road network prototyping.

2 Data Representation

OSM is a crowdsourced publicly accessible platform which provides rich Volunteered Geographic Information (VGI) that is free to use and edit [8]. 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. We briefly describe these elements in the following.

Nodes: A `node` is the basic element in the OSM data model. Each `node` 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 roads 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, as we will discuss later.

Relations: In the OSM data model, `way`s are connected by default when they share the same `node`s. That is, vehicles are allowed to travel from one `way` to another via their common `node`s unless other rules apply. Access restrictions for connected `way`s can be defined using `relation`s, for instance, to prohibit U-turns or right turns (with `no_u_turn` or `no_right_turn`, respectively).

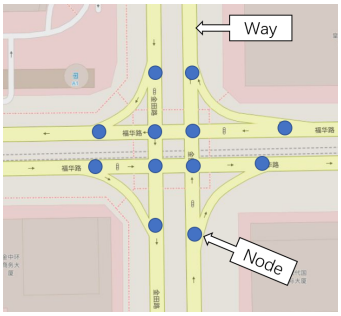
An example illustrating a junction represented by OSM data is depicted in figure 1a.

Similar to the OSM data model, a SUMO network also consists of nodes and edges, the latter describing, e.g., streets, bike lanes, or walkways [3]. SUMO provides several tools for generating SUMO road networks from OSM data: NETCONVERT and OSMWebWizard are both able to convert OSM data to SUMO networks, 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.

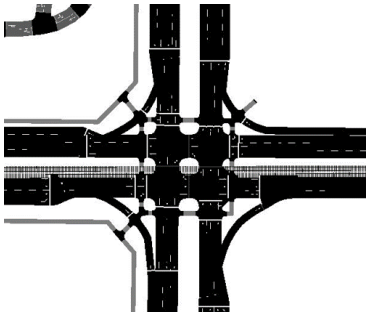
A major problem of converting OSM data to a SUMO network is to identify the association of nodes to intersections. As shown in figure 1a, one intersection could consist of multiple `node`s and `way`s, particularly when the `way`s are modeled by two parallel uni-directional roads with opposing directions. While this approach for modeling intersections is well suited for navigation or routing services, and allows for appealing visual map rendering, it poses challenges for the conversion to SUMO road networks.

Automatic conversion using, for instance, NETCONVERT, requires the correct identification of `node`s and `way`s that represent regular streets or are part of an intersection. Incorrect identification will lead to

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



(a) original OSM data[3]



(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.

inconsistencies, such as the one shown in figure 1b, where a single intersection is interpreted wrongly as multiple sub-intersections. This can cause vehicles to misbehave, triggering deadlocks, as we will show later, e.g., in figure 12a. Such problems are discussed at length in literature on medium-scale scenarios such as the work by Codecá et al. [9] and Rapelli et al. [10] regarding the generation of the SUMO Luxembourg and Turin scenarios. A commonly employed solution is therefore manual editing of large parts of the network.

3 Related Work

The challenges of road network generation from OSM data has been addressed in prior work. Some studies [11], [12] suggest that nodes which represent the same intersection can be integrated into one relying on reasoning over geometric connection relations, such as the spacing among the nodes, and whether they can form a circuit or not. However, this approach can typically handle intersections consisting of only a small number of nodes, and it struggles over complex intersections with extra turn lanes. Other researchers presented methods to use the semantic information in the OSM data to help locate intersections [7]. However, such methods highly depend on the homogeneous and consistent quality of the OSM data available.

Other studies [13], [14] aim 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, as well as 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 separating opposing lanes of the same road) into single `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 resulting simplified OSM network includes only streets and intersections which can be straightforwardly converted to a SUMO network by NETCONVERT, as discussed before.

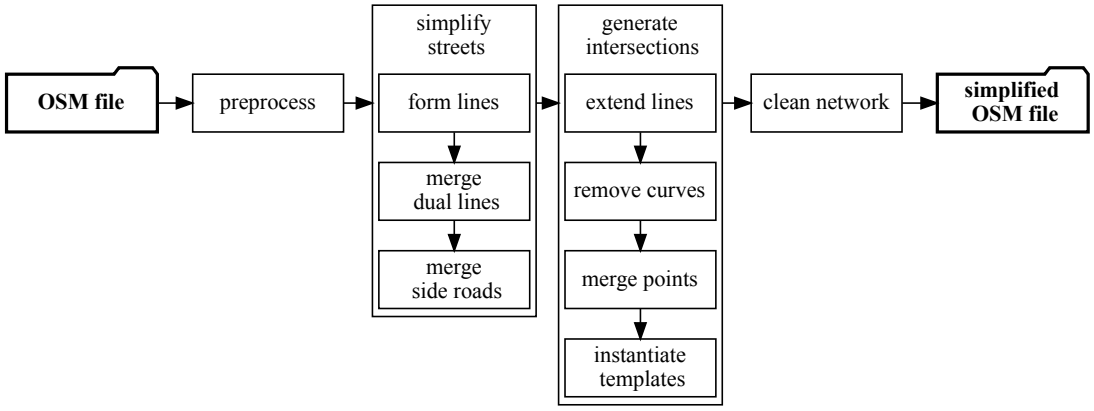


Figure 2: Steps performed by TSMM.

4 Methodology

TSMM takes OSM data as input and produces simplified OSM data as output, which is suited for conversion with NETCONVERT. The simplification of the OSM data is performed in a sequence of steps building upon each other, as illustrated in figure 2. We detail these individual processing phases in the following.

4.1 Preprocessing

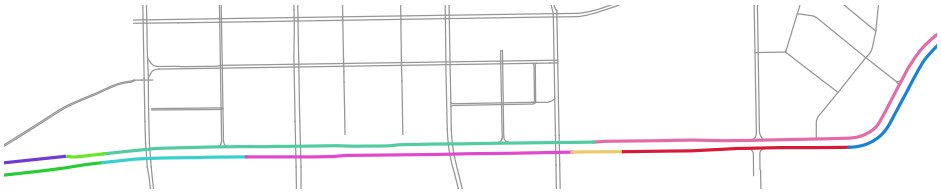
At present, the emphasis of TSMM is solely on generating road networks for motor vehicles. Roads used primarily by other traffic participants, such as bicycle lanes, crosswalks, and bus lanes, are beyond the scope of this work. Therefore, the raw OSM data is first pre-processed by filtering out irrelevant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

4.2 Simplifying Streets

The first step of simplification is to form `lines`. Note that a `line` is not the name of a data structure that is available in OSM nor in SUMO. Instead, in this paper, we use the concept of a `line` to refer to 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`.

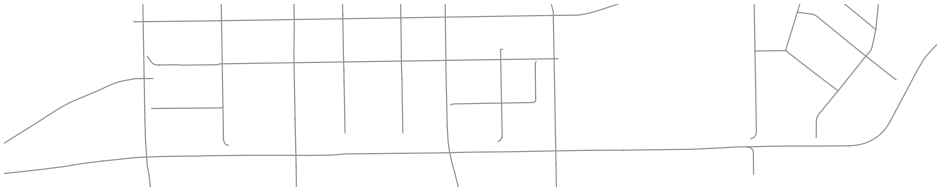
Forming `lines` is straightforward, starting from an arbitrary `way`, searching forward and backward for its adjacent `ways` and, if the angle between them is less than a certain value, adding them to the `line`. One notable situation is that it is possible for a `way` to have multiple other `ways` connected at its end. In this case, that `way` among all connected `ways` which has the smallest angle with the `line` will be selected to continue the `line`. Exemplary input and output data of this step is shown in figures 3a and 3b, respectively.

Two approximately parallel `lines` (such as those highlighted in figure 3b) often represent a dual carriageway in the real world. Thus, by identifying and merging these into one single `line`, the raw data can be substantially simplified. Typically, the two `lines` in the dual `lines` are not too far apart since they



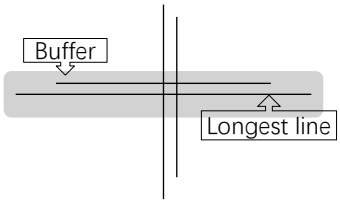
(a) illustration of how a long road with a median strip separating the opposing lanes is commonly modeled in OSM; different colors illustrate how the road is composed of multiple different OSM `way` elements

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

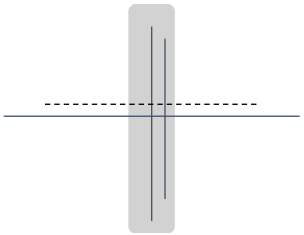


(c) result after merging dual `lines`

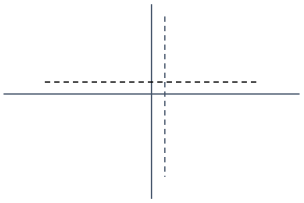
Figure 3: Simplification of `ways` in OSM data.



(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`.

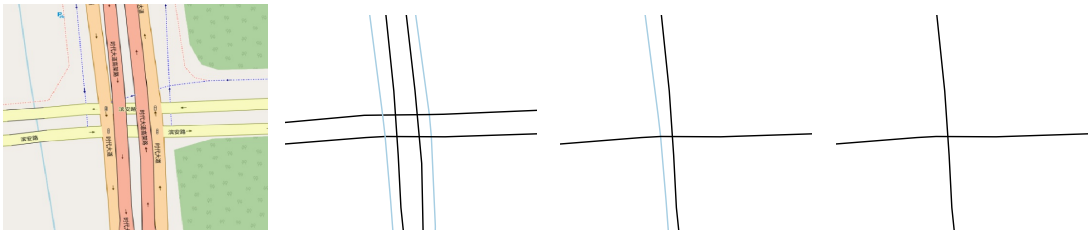
are representing the same street. For the same reason, their shape and alignment should also be relatively similar.

Therefore, we propose a *buffer* approach to merge the dual `lines`, as shown in figure 4. We use the term *buffer* to denote a polygon with a fixed width, which is generated around a `line`. First, such a buffer is generated based on the longest `line` l in the current map data. All other `lines` with 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. Consequently, `line` l will be then tagged as a bi-directional street. The amount of its

Algorithm 1 Buffer method for merging dual lines 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 that is of the same type and of which at least $p\%$ length is inside the buffer;
  end while

  return R



(a) rendering of side roads in OSM map[3]
(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.

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. The pseudo-code for the buffer algorithm can be found in algorithm 1.

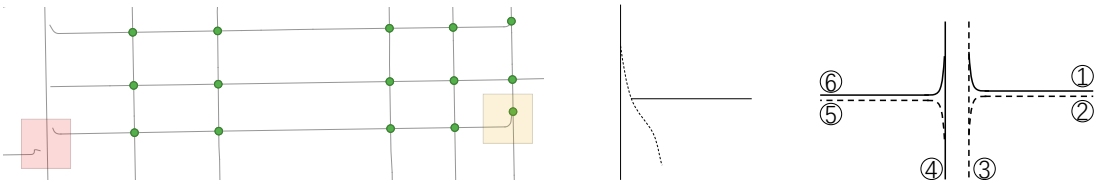
As shown in figure 3c, all the dual `lines` in the original OSM data are represented by a single `line` each now. As can be seen, the resulting network has a similar road layout compared to the original map while its complexity is substantially reduced.

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. [11].

At this point, the simplification for streets is completed. Next, we focus on the corresponding operations for generating the 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 among the remaining `lines` (illustrated as dots) are naturally the location of the



(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.

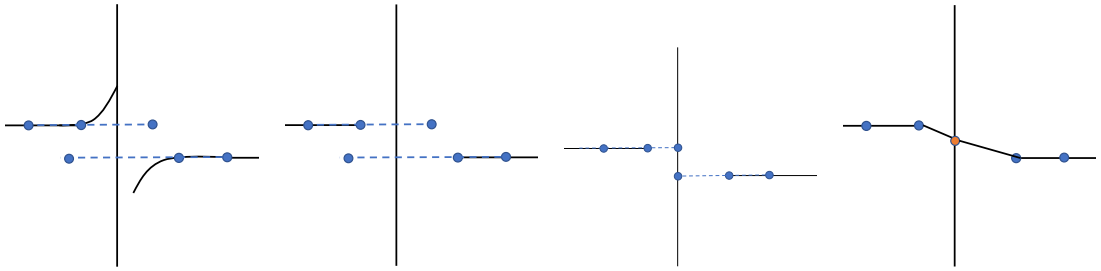
intersections in real world. Thus, the next step is to calculate the coordinates of these crossing points and insert them as `nodes` into the corresponding `ways`. While this approach works in most cases, additional processing is required for certain issues, as explained below.

Several streets are disconnected after the previous simplification, as is highlighted with the bottom left rectangle in figure 6a. One case of such 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. Another case of disconnection happens after the dual `lines` merging process, as is illustrated in figure 6c. Originally, `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 a disconnection between `line` 1 and the dual street in the simplified road network.

Apart from the problem of disconnections, unreasonable intersection shapes (bottom right rectangle in figure 6a) caused by the simplification process also need to be considered. The reason for this issue is shown in figure 6c as well. After `line` 5 was 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 edge in the SUMO network, an intersection with an unreasonable shape will be generated: Right-turning vehicles driving from `line` 6 to `line` 4 are confronted with a sharp turn angle. This can lead to severe congestion in the simulation that is purely an artifact of the unreasonable intersection shape.

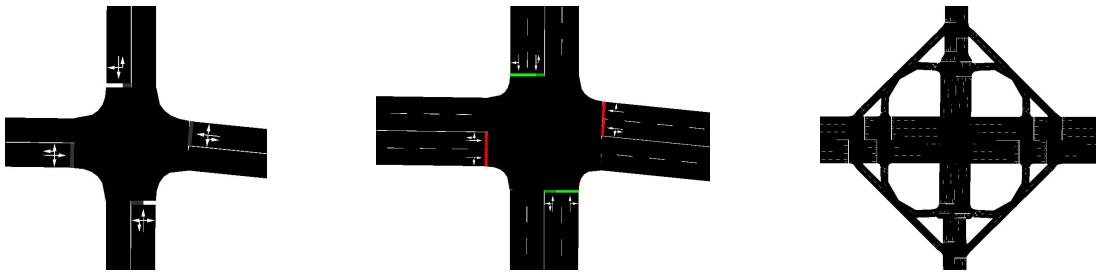
Therefore, there is an automatic pipeline in TSMM, illustrated in figure 7, which aims to solve the issues mentioned above: First, all `lines` are extended by a certain length at the head and tail ends. If a curve is detected there, the extension will be based on the overall angle of `line` and the curve will be subsequently removed. By doing this, the `lines` remaining in the map will reconnect to the others and crossing points can be formed to represent the intersections. Next, among the crossing points formed by the extended `lines`, very closely spaced points can be further merged together and excess trimmed. 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. We found that the length the `lines` need to be extended and the proximity parameter for merging intersections play an important role. TSMM suggests different values based on the road level, and users can tune them according to the roads characteristics (e.g., road density) of their target city.

Regarding the representation of intersections, TSMM provides three different types of templates which it instantiates depending on the level of roads that connect to the intersection.



(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.



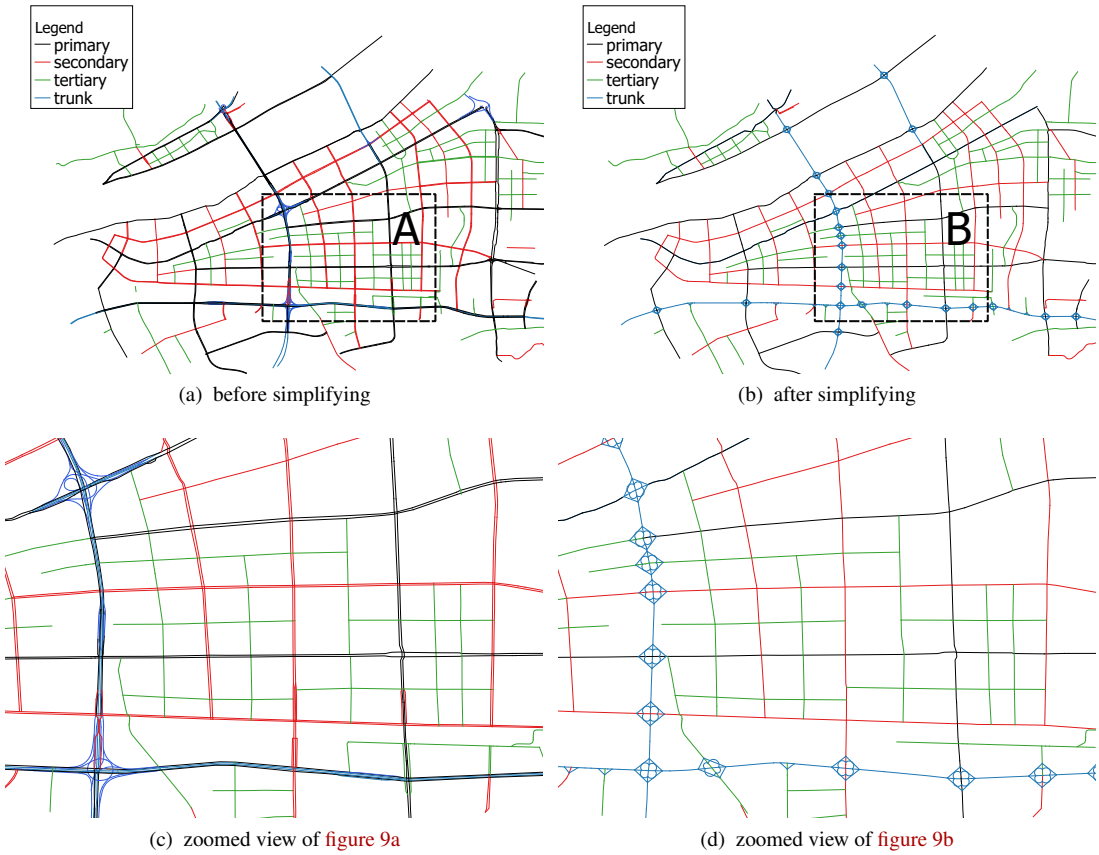
(a) uncontrolled intersection (b) traffic light controlled intersection (c) grade separated intersection

Figure 8: Classification of intersections.

Uncontrolled intersection (see figure 8a): In the real world, these intersections are regulated only by right-of-way rules. They are found mostly 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 (see figure 8b): Some intersections require traffic lights. The standard identification of traffic light controlled intersections 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 if there is no additional data filled in by the users. In addition, with TSMM, users can define which level of intersections should be controlled by traffic lights. TSMM is then able to identify and tag these additional `nodes` to ensure a proper conversion later.

Grade separated intersection (interchange) (see figure 8c): For streets like `motorway` and `highway`, it is necessary to separate the traffic in the vertical grade in order to maintain high driving speeds of vehicles and maximize traffic throughput. Users of TSMM can therefore define which type of street should be served with interchange. So far, TSMM always uses clover leaf interchanges to represent all grade separated intersections.



(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.

4.4 Network cleaning

Full connectivity of the road network plays an important role in many SUMO simulations, yet mapping errors in OSM data can sometimes leave spurious roads in the network. After simplifying the road network from the previous steps, TSMM therefore uses the Breadth-first search (BFS) algorithm to group all intersections into different clusters according to their connectivity to each other. In the end, the cluster with the highest number of intersections will be defined as the true road network – and intersections and streets which are not in it will be removed from the road map. In general, the removed parts are supposed to occupy only a small proportion, otherwise, the user needs to reconsider 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 the Chinese city Hangzhou, which covers 72.2 km^2 in total, was tested. The road network contains 161 km roads and 128 intersections, among which 28 are grade separated intersections.

Figure 10: SUMO network of Binjiang.

As shown in figure 9a and figure 9c, the original OSM network mainly consists of dual uni-directional roads. As can further be seen, `primary` (in black) roads appear in several places as the side roads of `trunk` roads (in blue). This means that the vehicles driving on the `trunk`s should not have conflicts with traffic from other roads; the connection among them should be achieved through 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. Such poor identification often even leads to broken roads and deadlocks.

Therefore, as described in section 4, TSMM merges side roads into main roads and inserted clover leaf interchanges for all the intersections on the merged roads during the simplification process. The simplification was executed on a mid-range Desktop machine operating at 3.90 GHz taking 21 seconds with non-optimized code.

Based on the proposed methodology, TSMM is able to simplify the original road network while retaining the topological information with respect to location of intersections, overall road layout, and road classifications. In this manner, although the layout visually differs from reality, it still ensures to a large degree the topology of the road network regarding traffic characteristics (figure 9b and figure 9d).

5.1 Preparation

After the simplification, the OSM road network is fed into NETCONVERT 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[4]

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

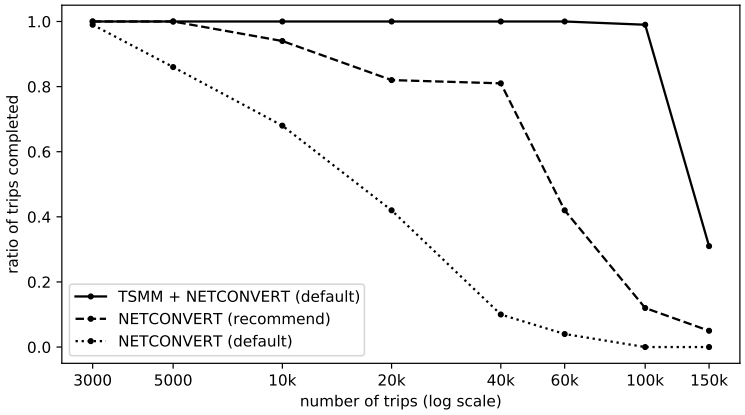


Figure 11: Rate of completed trips vs. increasing travel demand. Different road networks can sustain differently high demand.

for NETCONVERT. Conversion with these additional parameters is able to simplify the raw OSM data automatically, joining the nodes heuristically, guessing location of ramps, and inserting traffic lights for uncontrolled intersections. The parameters are:

```
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
```

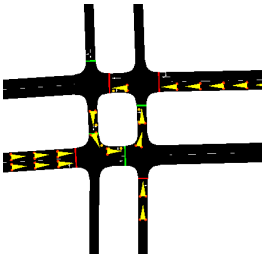
We also prepare traffic demand for the experiments as follows. To ensure the consistency across all scenarios, the trips files for each network are taken from the same dataset. In the dataset, a certain amount of trip data is stored, where each trip consists of a coordinate pair within the study area, and a randomly assigned departure time within the time interval 0–180 min. In the next step, the trip data is converted to SUMO readable trips files for each network. The departure time is kept identical to the one in the dataset, while the origin and destination places are set to the closest edges to the coordinates.

The simulation time is set to 6 hours, which is sufficient for all the vehicles to finish their trip if there is no congestion. The teleporting of the vehicles (the fail-safe procedure employed by SUMO when unexpected events such as deadlocks happen) is disabled unless a collision has occurred or if no valid route has been found for a vehicle. At the end of each simulation, by analyzing the car counts statistics and visually inspecting the network to identify bottlenecks, the performance of each road network can be assessed.

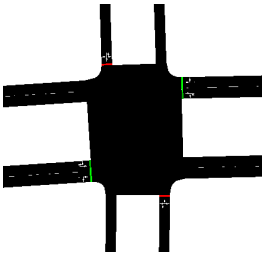
5.2 Assessment

As illustrated in figure 11, continuously increasing the initial amount of traffic reveals that the tested road networks start to show uncompleted trips when the traffic volume reaches 5000, 10 000, and 100 000 trips, respectively. Much more so than the quantitative performance of the networks, however, we are interested in why bottlenecks materialize.

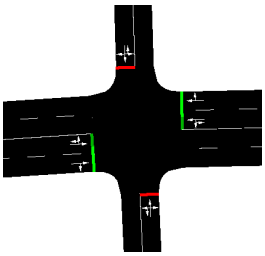
At the end of the simulation we therefore inspect visualizations of jammed intersections. This analysis of the causes of bottlenecks allows to determine whether the congestion is caused by excessive traffic volume or by errors in the road network itself.



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

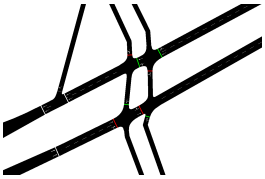


(b) same intersection as figure 12a converted via recommended settings

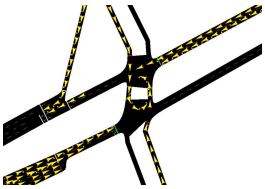


(c) same intersection as figure 12a converted via TSMM

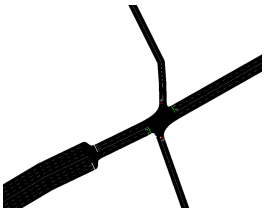
Figure 12: Analysis of uncompleted trips in the road network converted directly via default settings with 5000 initial trips.



(a) same intersection as figure 13b converted via default settings (shown without traffic)



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



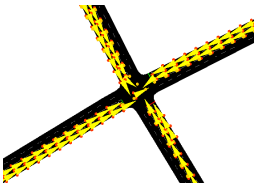
(c) same intersection as figure 12b converted via TSMM

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

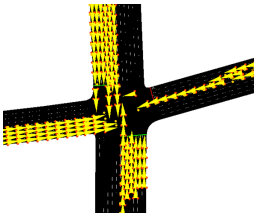
The results after the simulation reveal that the non-simplified road network generated using the default settings has the lowest traffic capacity compared to others. When 5000 cars were imported into the network as input demand, nearly 400 cars were already unable to complete their trip. Congestion occurred at many intersections at the end of the simulation. Through observing one of their representatives (figure 12a), it is revealed that the congestion was not caused by excessive traffic but rather by errors in the road itself. This is because, by 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, when too many vehicles enter this complex multi-intersection area, the vehicles block each other due to traffic lights and narrow spaces. A deadlock was thus created and henceforth obstructed all vehicles that were supposed to pass by. On the contrary, the same intersection generated via recommended parameters (figure 12b) and via TSMM (figure 12c) are free of such deadlocks.

The road network, converted with the recommended parameters via NETCONVERT shows a remarkable improvement in the deadlock discussed above. However, the algorithm it uses can only cope with simple cases when multiple `node`s form one intersection in OSM data. It still lacks the ability to address more complex situations. Thus, at the end of the simulation, unresolved traffic congestion at many intersections still appeared across several intersections. One of these intersections can be seen in figure 13b. Part of the

(a) bottlenecks of network converted via TSMM



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



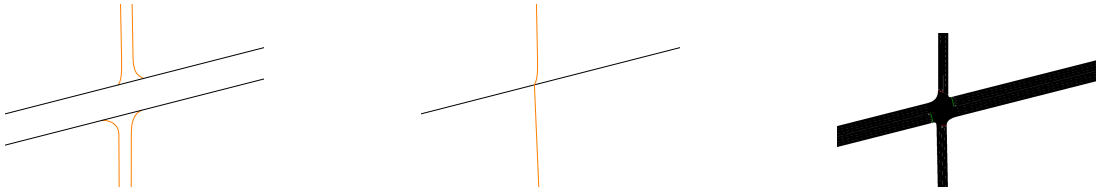
(c) congested intersection in right-hand of figure 14a; converted via TSMM

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

nodes in this widely spaced intersection are clustered together and two independent traffic light controlled intersections are generated in the end. Although better than the result converted via default settings (figure 13a, shown without traffic), where four intersections are created, a deadlock is still unavoidable when a large amount of traffic is flowing into this area. On the contrary, TSMM is able to identify all the nodes associated with this intersection properly and generate a deadlock-free intersection (figure 13c).

Still, even the simplified network generated by TSMM is not free of congestions. As shown in figure 11, however, the simplified road network generated via TSMM can accommodate nearly 100 000 vehicles to finish their trips. This is a multi-fold increase in capacity compared to the other two networks and might hint at less deadlocks. Figure 14a depicts the location where congested vehicles gathered after the simulation. By zooming in on these two intersections, as shown in figure 14b and figure 14c, respectively, it can be confirmed that the congestion is solely caused by excessive traffic volumes, as undesired network errors are not revealed in either area.

Summing up, in our experiments, errors were found in both networks generated without TSMM, most owing to the data quality of OSM data. These errors caused deadlocks and further limited the capacity that the networks should have reached. As discussed, the errors are mainly caused by the failure of identifying the correct OSM nodes to form the intersections, especially in cases involving dual carriageways and side roads. This matches exactly with the goals of TSMM. Our proposed approach proved its ability to simplify OSM data while preserving the topology of the network (at the cost of a less detailed road network representation). After the simplification with TSMM, the generated road network not only reaches the largest capacity, but also avoids any deadlocks caused by road network errors.



(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.

6 Limitations

While already a worthwhile tool for the simplification of OSM data, follow-up research will be carried out to address some limitations still present in TSMM.

One limitation concerns turn restrictions. Currently, TSMM is discarding turn restrictions modeled in OSM – both explicitly and implicitly. The simplified road network is always fully connected at every intersection, i.e., straight ahead, right turn, left turn, and U-turn are all allowed at every intersection. This is also due to the simplification process. For instance, figure 15a shows an intersection as modeled in the original OSM data, where traffic on the highlighted road is restricted to right and left turns only; driving straight through the intersection from north to south or vice versa is not possible. In the simplified road network generated via TSMM, however, this complex intersection is replaced with one crossing point (figure 15b). Consequently, the implicit turn restrictions are lost in the final generated SUMO network, as shown in figure 15c.

Another limitation concerns roundabouts. The method proposed in this paper is not applicable to roundabouts. In OSM data, a roundabout is always a closed circular loop consisting of one or several ways. Thus, when forming `ways` into `lines`, the algorithm of TSMM would be trapped in an infinite loop. The current interim approach to avoid an infinite loop is to identify and filter out any roundabout during the pre-processing phase, and filling it back in 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 therefore needs to be proposed.

Yet another limitation concerns extensibility. TSMM is hoped not be limited only to OSM data. To meet the different needs of map detail, more Application Programming Interfaces (APIs) must be developed to be able to consume other data sources. Speed limit, lane count, layout of intersections, and traffic lights can all be configured in an incremental way within the pipeline only if additional data sources and corresponding APIs for processing them are available.

Lastly, by leveraging validated scenarios such as SUMO Luxembourg [9], quantitative evaluations of similarity of the generated road network must 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, indicating if TSMM might not just work for large-scale simulations interested in more macroscopic effects, but also for high-detail simulation.

7 Conclusion

In this paper, we presented a method along with a reference implementation, Traffic Simulation Map Maker (TSMM), to substantially increase the automation level of road network prototyping by simplifying the OpenStreetMap (OSM) data while preserving important topology information. The main objective was to enable the study of traffic simulation dynamics at scale using real-world road networks, while minimizing the need for solving the long tail of problems related to the road network generation.

Primarily this is achieved by merging dual carriageways and side roads into single `lines`, then instantiating intersection templates. We believe this is a good trade-off between precision and automation that solves many conversion problems at the source, i.e., the map, most of the errors. While there is definitely a loss in fidelity with respect to the real world, many properties of the road network are preserved. We thus argue that TSMM greatly improves the availability of arbitrarily large and usable road networks on top of available OSM maps by reducing the complexity for conversion tools and traffic simulation researchers alike.

A proof-of-concept study using OSM data from Binjiang, China demonstrated that TSMM is able to generate a road network with well-preserved topological information within seconds. Compared to the non-simplified OSM data, the SUMO network generated from the simplified version effectively avoided all artificial deadlock situations.

The reference implementation of TSMM is publicly available to interested researchers.[2] In future work, it will be continually upgraded to address more sophisticated situations like roundabouts and to enhance the realism of the resulting network by incorporating more details.

References

[1] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator VISSIM," in *Fundamentals of traffic simulation*. Springer New York, 2010, pp. 63–93. DOI: 10.1007/978-1-4419-6142-6_2.

[2] D. Zehe, S. Nair, A. Knoll, and D. Eckhoff, "Towards CityMoS: A Coupled City-Scale Mobility Simulation Framework," in *5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication (FG-IVC 2017)*, Erlangen, Germany: FAU Erlangen-Nuremberg, 2017.

[3] 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.

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

[5] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "SUMO (Simulation of Urban MObility)-an open-source traffic simulation," in *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, 2002, pp. 183–187.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.