Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

AUTOMATIC GENERATION OF ROUTE NETWORKS FOR MICROSCOPIC TRAFFIC SIMULATIONS

Niclas Feldkamp
Steffen Strassburger

Ilmenau University of Technology
Helmholtzplatz 3
98684 Ilmenau, GERMANY

ABSTRACT

Microscopic traffic simulation is a well-accepted simulation approach for simulation problems where the effects of individual driver behavior and/or vehicle interactions need to be taken into account at a fairly detailed level. Such problems include the optimization of traffic light controls patterns or the design of lane layouts at intersections. Such simulation models typically require very detailed and accurate models of the underlying road networks. The manual creation of such networks constitutes a high effort, limiting the simulated area in practical applications to the absolutely necessary. With the increased availability of satellite based geographical data we investigate, if and how such data can be automatically transformed into route networks with adequate level of detail for microscopic traffic simulation models. We further outline the design of data structures for an extensible simulation framework for microscopic traffic simulation which is capable of including different types of publically available data sources.

1 INTRODUCTION

There are several well-known approaches for traffic simulation. The majority of the approaches can be classified as either being macroscopic or microscopic. In macroscopic models, the modeling focus is on the traffic flow and its properties. Such models describe the dynamics of vehicles and drivers in terms of mathematical equations. The approach is comparable to models in physical sciences where the thermodynamic behavior of liquids and gases are described (Treiber and Kesting 2013). The focus of the investigations in macroscopic models is on collective effects that do not depend on the details of individual drivers or vehicles. Typical parameters of macroscopic models are variables such as traffic flow, traffic density and average velocity.

Microscopic models on the other hand model the behavior of individual vehicles and their drivers. They focus on the interactions with other vehicles as well as with the traffic infrastructure (e.g. traffic lights, intersections, etc.). For realistic models, several vehicle following models have been suggested (Olstam and Tapani 2004). One example are psycho-physical vehicle following models that try to emulate realistic driver behavior depending on psychological and physical factors (Wiedemann 1974).

Finally, there are mesoscopic approaches that combine elements of both classes, e.g., by maintaining individual vehicle representations in combination with aggregate representations of traffic dynamics (Burghout et al. 2006).

Modeling approaches can be further classified according to discrete vs. continuous time, state, and space representations. Our investigation focusses on discrete time models with continuous state and space representations, which is typical for car-following models and microscopic traffic simulations. These

models are typically simulated using time-stepped approaches, but it has been shown that the usage of discrete event based time advancement is possible and advantageous (Schulze and Fliess 1997).

The largest challenge for microscopic traffic simulations is their requirement for a very detailed and accurate representation of the underlying traffic infrastructure including road networks, traffic intersections, traffic lights etc.

In alignment with the promises of automatic (i.e. data driven) model generation (Bergmann and Strassburger 2010), a modern microscopic traffic simulation framework has to be able to dynamically adapt to a variety of potential data sources of input data, also beyond data concerning the traffic infrastructure.

Given these challenges, the objective of the research presented in this paper was to establish an extensible (preferably web based) framework for microscopic traffic simulation capable of integrating multiple open access data sources. The focus of our investigation was on the question if and how satellite based geographical data can be automatically transformed into route networks with adequate level of detail for microscopic traffic simulation models. We further investigated how additional data sources, preferably via automated web-services can be infused into the simulation model. The investigations in this paper extend previous work, in which the general suitability of OpenStreetMap as input data for logistics simulations was discussed (Meyer et al. 2013).

The remainder of this paper is structured as follows: Section 2 discusses categories of input data required for conducting microscopic traffic simulations and their potential sources and impact. Section 3 presents OpenStreetMap and our evaluation of its suitability as data source for generating route networks for microscopic traffic simulations. Section 4 presents the approach and our extensible data structure for route network generation and representation. Section 5 briefly introduces our prototypical implementation of a web-based microscopic traffic simulation framework. Section 6 summarizes our findings.

2 INPUT DATA FOR TRAFFIC SIMULATION

There are many classes of potentially relevant input data for microscopic traffic simulations. The most obvious class of input data is certainly related to geographical data of the traffic infrastructure. The traffic infrastructure primarily has to provide geographic location information about roads and their intersections. Further important attributes belonging to roads include the road type, speed limits, number of lanes, and several other. Secondary information about the traffic infrastructure may include the location of bus stops, rail road crossings, traffic flow limitations and the like.

Important information concerning intersections, next to their location and the roads that they connect, relates to questions concerning the intersection layout and potential traffic flow regulations. The former relates to the lane layout of the intersection (are there dedicated lanes for left/right turns, if so, how many of them and what is their length), the latter relates to questions of allowed or disallowed vehicle operations (is left turn allowed, are roads allowable both as input and output, etc.).

So far, we are dealing with information that is mostly available in modern car navigation systems. Therefore, we might assume that all required base information is contained in geographical information systems and thus readily available. However, as we will discuss further below in the paper, the requirements of microscopic traffic simulations go beyond what is required for car navigation systems. One example is the “lane assistant” contained in some car navigation systems. While this is a perfect tool for helping a user to choose the right lane when he/she approaches an intersection, it is typically based on a rather abstract level of detail of information about the lanes, and not on data such as their exact position, length, and shape. A microscopic traffic simulation would need exactly the latter data to perform its calculations, e.g., concerning the number of vehicles a dedicated left-turn lane can store before cars back up into other lanes. Such information is typically not contained in the base GIS data.

In section 3, we therefore investigate OpenStreetMap as an exemplary database concerning its suitability to provide information at a level of detail required for microscopic traffic simulation. While

this constitutes the focus of the work presented here, a framework for microscopic traffic simulation should be capable of integrating a wide range of other potentially relevant data sources.

Such data sources include the following data categories:

- Traffic load data: Next to the traffic infrastructure, the accuracy of microscopic traffic simulation highly depends on realistic traffic load data. In the microscopic approach, this data will be required for assigning entry and destination points for individual vehicles, as well as their type. There are several potential sources for obtaining such kind of data, including direct approaches like induction loops, traffic cameras with image recognition software, or manual observations as well as indirect approaches based on data mining methods applied to statistical data. Further methods, mostly applicable for macroscopic models, are the determination of car trajectories and floating car data (Treiber and Kesting 2013).
- Current traffic conditions: Information about current traffic conditions ("traffic overlays") are used by modern route planners or navigation systems (e.g. Google Maps, Tom Tom) to calculate the best route under current traffic conditions, e.g., by minimizing the encountered traffic delays. For microscopic traffic simulation, such data could be interesting, possibly on an aggregated level, to consider in dynamic (and in that way more realistic) routing decisions of individual vehicles.
- Weather data: Similar to current traffic conditions, the inclusion of weather conditions could be interesting for more accurate simulations, especially when it is used in operational decision support scenarios (Schulze et al. 1999), or when simulation experiments explicitly want to impose certain weather dependent restrictions on the infrastructure (e.g., icy roads, limited visibility, etc.).
- Nearby locations and points of interest: In order to extrapolate load data, it might be useful to analyze this additional information in order to investigate frequently used routes between entry and exit points within the network.
- Driver profiles: The correct representation of driver behavior is important for modeling vehicle interaction at a detailed level. It has implications on acceleration/deceleration, gaps towards the predecessor car, reaction time-lags etc. The representation of driver behavior is done in conjunction with the vehicle-following model applied. The required input data depends on the desired level of detail in the behavior representation. It differs from potentially fine-granular representation (e.g. found in agent-based models) to simulations with averaged driver behavior. In the latter, driver and vehicle behavior may not even be distinguished, leading to driver-vehicle-units.
- Traffic rules: For implementing the dynamic vehicle behavior, traffic rules must be taken into account. As they may vary depending of the location, they can be considered as input data. Examples include passing rules, traffic light rules (e.g., right turn on red), regulations for intersecting roads with equal rights, rules for roundabout traffic, speed limitations depending on road type, etc. Modern traffic guidance systems can dynamically impose further limitations (speed, passing) that may be interesting to dynamically include in certain scenarios.

These are possible categories for data sources which may be integrated into the framework to support the accuracy of an automated route network generation. The prototype (see section 5) has exemplified the acquisition of weather data as well as nearby locations via a web service based approach.

3 OPEN STREET MAP

Digital maps have been steadily gaining popularity in the last few years, especially due to mass distribution of GPS- and web-enabled consumer devices like smartphones or car navigation systems. Not only do people want to query digital map data, but also mapping and sharing the world around them has

begun to get popular. Therefore crowdsourcing based data collections have turned out as a serious alternative to commercial, proprietary map data. In this work, we used OpenStreetMap, which is with its 1.5 million users the most important open-source map data project and is commonly accepted as the "Wikipedia of Geodata". Especially in crowded areas, including personal and local knowledge resulted in very detailed maps which include a variety of additional information such as points of interest, road signs, street surface or public transportation routes. Furthermore, changes in local conditions often causes an almost immediate adjustment of the map data. On the other hand, the data quality decreases in less populated and rural areas due to decreasing number of actively involved OSM community members living there.

The OpenStreetMap data is licensed under the Creative Commons 2.0 license, entailing no technical or legal restrictions to users in editing and utilizing the map data. For extracting data, OSM provides an API access where a specified map section can be exported via the OSM XML file format. This data structure is kept very simple and consists of three basic elements node, way, and relation. Nodes are geographic points stored with their ID and corresponding latitude/longitude. A way on the other hand is a collection of node elements to represent routes between these points in order to map roads, boundaries or building layouts. On top of that, the relation element is used to define more advanced logical or geographic relationships between these basis elements, most commonly mapping turn restrictions for vehicles. Moreover, the OSM data structure is gaining its variety and mightiness through the use of additional tag-attributes, which can store an information in a key/value pair of unicode strings of up to 255 full unicode characters (OSM 2014).

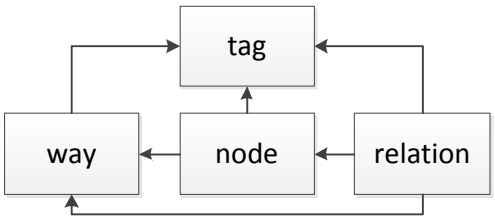

Figure 1: Simplified overview of OSM data structure.

An overview of the OSM data structure can be seen in Figure 1. Because of its simplicity, the OSM XML file format can be easily transferred into standardized GIS formats. Therefore, using OSM data in this work can be viewed as a generic example for utilizing digital map data for microscopic traffic simulation.

4 ROUTE NETWORK GENERATION

4.1 Basic Data Structures

In order to map any possible road network, a generic data structure is necessary which is capable of abstracting and representing the defining shape of the road network. Further, the data structure must facilitate the implementation of extensions or more detailed solutions. We therefore relied on the basic data structure of node and way elements that OSM provides, which can easily be transformed into a directed graph structure. A directed graph is needed as the basis for guiding moveable simulation entities through the network using graph-based path algorithms.

By transforming graph nodes with underlying GPS-Coordinates into X/Y-Coordinates and mapping them to the screen, a raw breakdown of the road network can be generated. We developed further data structures for more specialized classes like the *ConnectNode* class, which is used for representing supplementary inserted nodes in order to model crossroad areas. An extraction of the frameworks core

data structures is shown in Figure 2. By deriving from the basic classes *MapNode* and *MapWay*, more specific classes representing additional information can be added easily into the framework.

In addition, more advanced data structures have been implemented for creating a more realistic and rich simulation model, like intersections, traffic lights and induction loops.

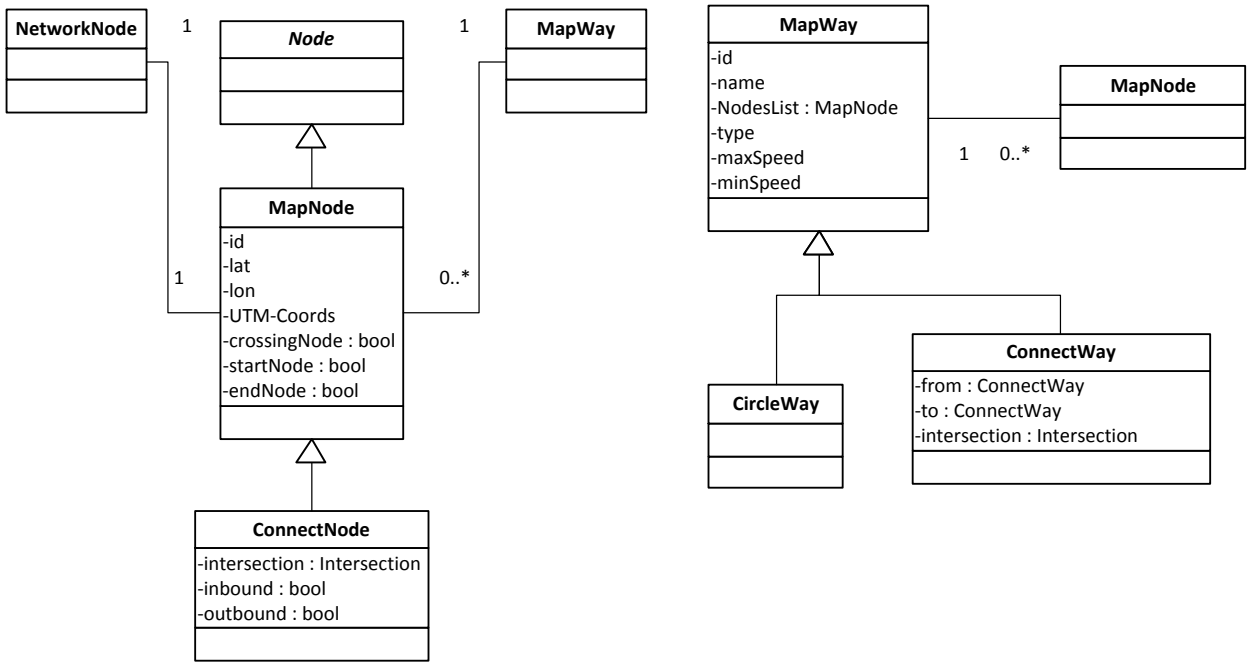

Figure 2: Core data structures with example attributes and derived classes from basic node and way structures**.**

In its entirety, the final route network framework consists of the basic graph elements plus underlying advanced elements. Those advanced elements are generated based on the rich additional information that OSM tags and map features provide. The data structure is further capable of adding even more information that could be gathered by connecting to GPS location-based web services. This includes data about geographical altitude, points of interest, road conditions, or nearby cities.

To make use of related information, we implemented a process of reverse geocoding, which describes the process of reverse converting a GPS point into human readable information. Therefore, while processing the users map extraction, we are simultaneously passing the maps bounding coordinates to geo information web services, which are registered via a XML configuration file beforehand. Furthermore, we created generic placeholder classes for managing and storing the obtained information and are independent of the services response format.

A special challenge was to define a data structure for intersections. In the context of OSM, an intersection is modeled as two or more ways intersecting in the same node, which is shown exemplarily in Figure 3. After the analysis of a variety of raw OSM data extractions, we found out that most intersecting points have a number of incoming ways between two and four. More complex traffic intersections like the one shown in Figure 4 are modeled by connecting multiple intersection points.

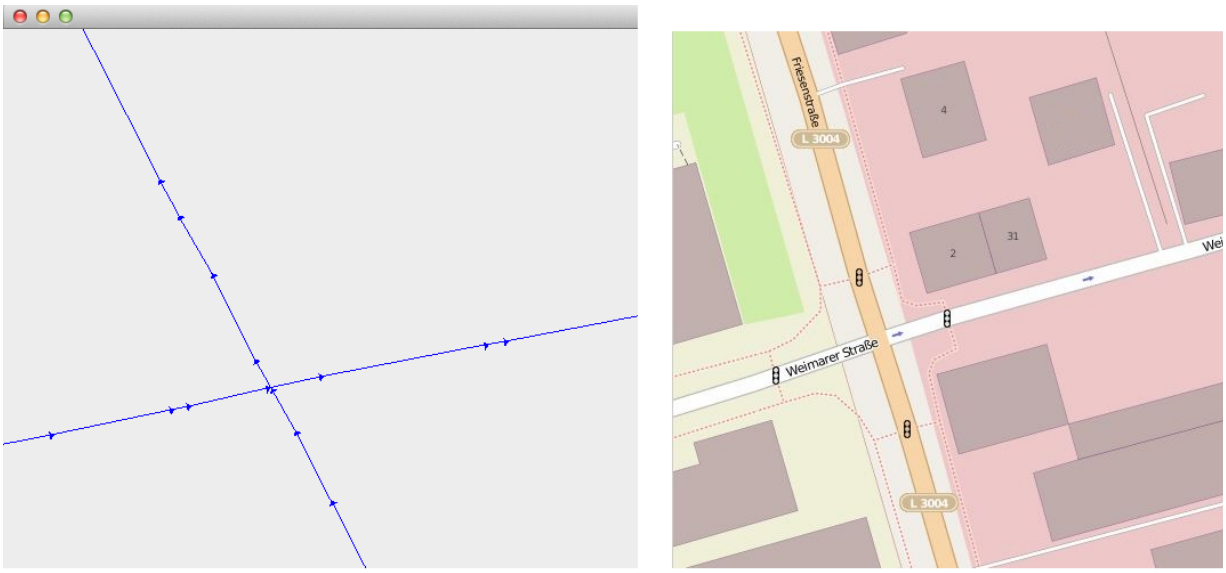

Figure 3: On the right side: OSM map section with a simple intersection of two roads. Left side: Corresponding graph structure which has been made visible by connecting the OSM nodes with lines and drawing them on a window panel.

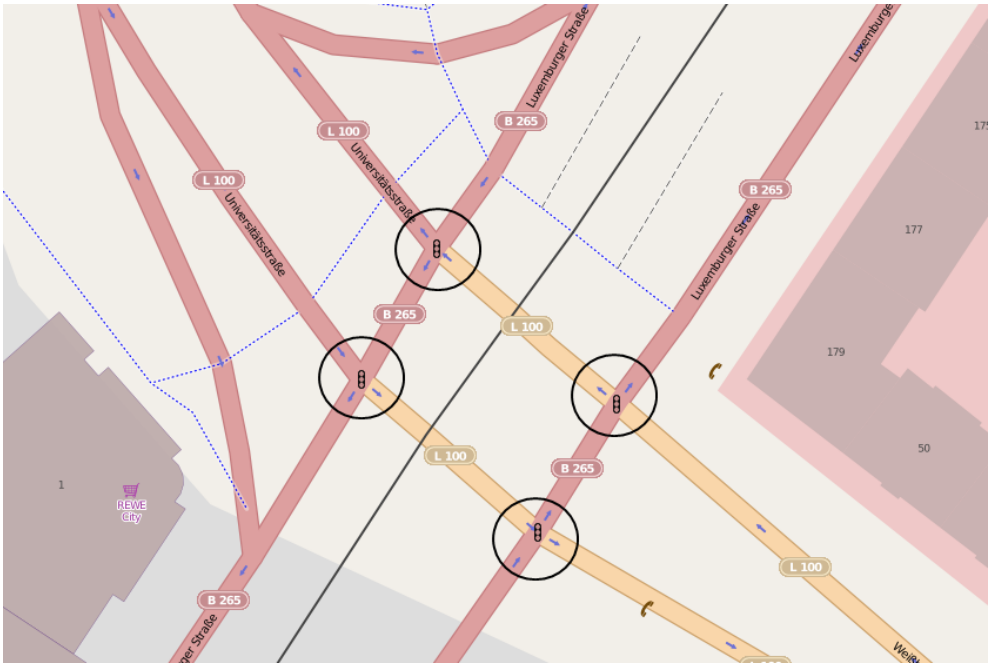

Figure 4: A complex intersection consisting of four basic intersections.

The limited number of intersections with more than four incoming ways frequently have been erroneously modeled as roundabouts. Therefore we created a few intersection phenotypes that are able to cover almost every thinkable intersection, which can be seen in Figure 5.

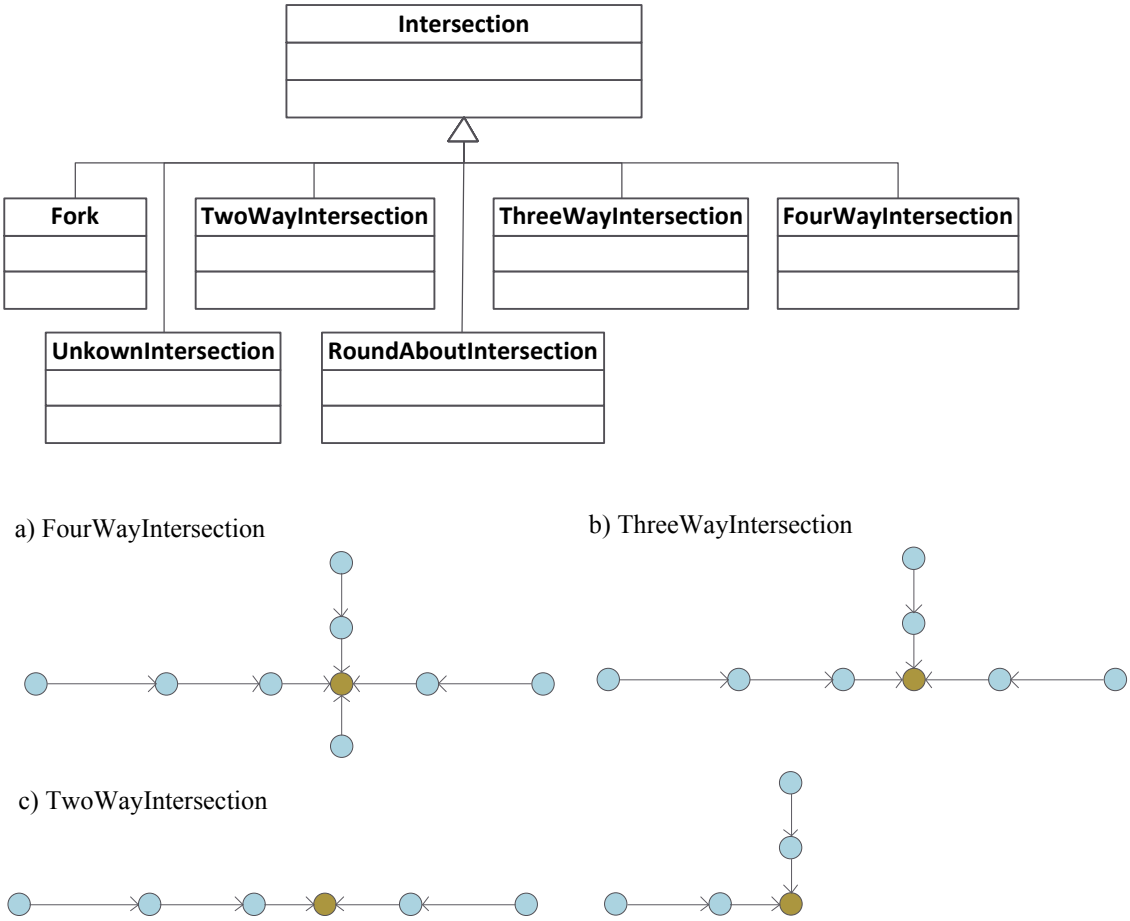

Figure 5: Intersection classes for intersection phenotypes.

4.2 Algorithm for Route Network Generation

Based on this data structure, we implemented an algorithm for an automatic transformation of raw OSM data into the frameworks data structures. In the end, this algorithm generates a route network adequate for microscopic traffic simulations. This process consists of five steps:

4.2.1 Importing the OSM XML file

Firstly, the relevant OSM data elements are transformed into the basic framework elements. Further, the provided OSM tag information has to be stored for using it later in the process.

OSMs richness in additional tag information is also a weak point when automatic processing of this information is needed. Currently, OSM provides over forty thousand distinct tags and a mapped node has on average 3.36 tags attached. In order to control the huge amount of available information, we created a pre-defined configuration file containing inclusion or elimination criteria. Thus, the amount of stored data can be drastically reduced by using only the data needed for the current application. To avoid problems regarding ambiguous or redundant tags, the configuration file maps OSM tags to their corresponding framework classes. In addition, possibly missing data, e.g., speed limits or road access restrictions, may be added via the configuration file.

4.2.2 Converting coordinates

As most GPS devices do, OSM handles point coordinates in the WSG84 coordinate system, which means that coordinates are presented in a geographical longitude and latitude unit. To project this spheroidal surface accurately onto a screen, it is required to transform those coordinates into a two-dimensional reference system. For this purpose we used the commonly used Universal Transverse Mercator grid.

4.2.3 Finding intersection points and splitting ways

In order to gather information about intersections and related incoming and outgoing ways, it is necessary to identify the center node where two or more ways are crossing. By splitting those into connected but independent ways, we are able to count the number of incoming ways on an intersecting node for determining the corresponding intersection phenotype.

4.2.4 Creating opposite lanes

By definition, OSM ways are undirected graphs edges. For a more accurate road network model, it is required to separate them into distinct road lanes which can be seen in Figure 6. In this process, potentially tagged one-way road information has to be taken into account.

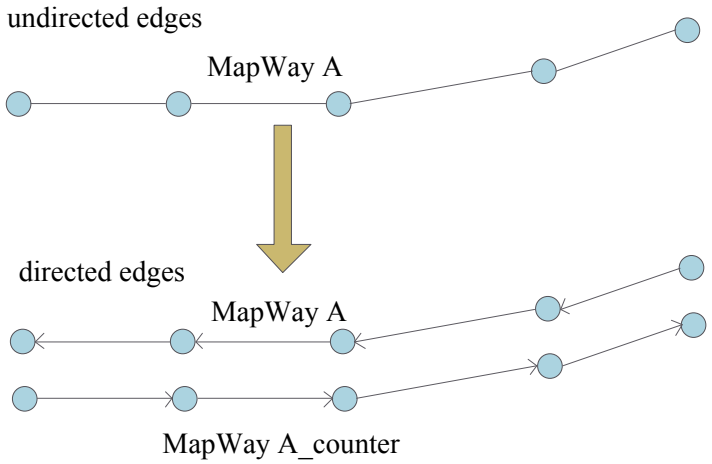

Figure 6: Converting ways into a directed graph structure.

4.2.5 Generating intersection areas

In general, road intersections in map data are presented by two or more roads that cross in a specific coordinate. In the context of microscopic traffic simulation, this is an improper simplification. In order to simulate accurate traffic behavior and to gain insights in potential traffic flow regulation around an intersection, we need to model distinct turning lanes instead of a single abstracted crossing node. Therefore, we provide algorithms to approximate a more realistic crossroad layout, which is demonstrated in Figure 7.

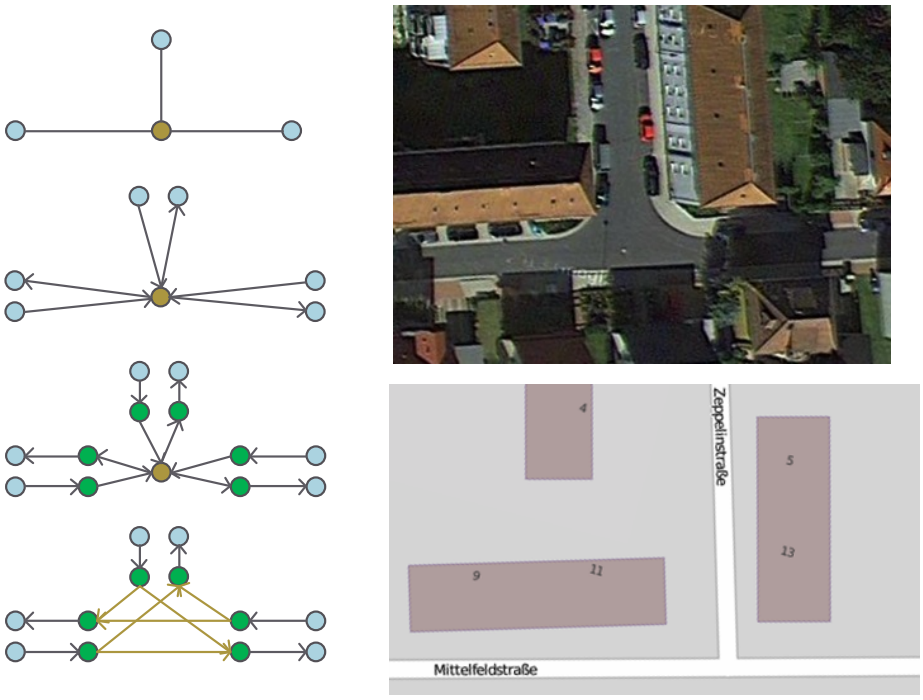

Figure 7: Left side from top to bottom shows the process of creating a more realistic intersection. Lower right side shows a screenshot of the original OSM data and upper left side shows a Google Maps satellite image of the actual intersection.

The creation of the intersection also takes optional OSM information like turn restrictions or tagged traffic lights into account.

On the basis of the identified intersection phenotype, we further provide basic traffic light signal patterns. By blocking/unblocking distinct turning lanes in the signal circuit, it is also possible to implement any thinkable signal pattern manually. Right of way priorities are defined for each intersection by defining priorities for different road types via the configuration file depending on the road type that is attributed through OSM-tag elements. For example, a "primary"-road has a higher priority than a "residential"-road. By analyzing the road type of the intersections incoming ways, the right of way priorities can be determined and queried by objects moving through the intersection during the simulation process.

After finishing the steps above the corresponding graph can be drawn simply by connecting all nodes in the order they are connected with their corresponding *MapWay*-Object.

While this process enables us to automatically generate a more realistic intersection layout for microscopic traffic simulations, there are some potentially significant limitations of this approach. We obviously cannot conjure up geographical location data that is simply not present in OSM. Therefore, our algorithm has no way of knowing the correct length of the generated lanes, stopping lines depending on lines of sight, or the like. For these things, a simple heuristic approach is used. It is, however, a reasonably good first guess on the basic turning lane structures that probably will be in place at a certain intersection depending on its type and the turn restrictions modeled in OSM. For several applications, a manual post-processing of the generated layout will be required, though (see section 6).

5 PROTOTYPICAL IMPLEMENTATION

For the proof of concept and the framework's flexible application, we implemented a prototype into a server-sided web application using a Tomcat application server and the java based discrete event

simulator Desmo-J (Lechler and Page 1999, Page and Kreutzer 2005). Our goal was to create a streamlined user interface via the web-browser. When a user selects a certain map area in the web-browser by drawing a bounding box, the application handles the process of downloading OSM data creating the route network autonomously. To keep user interaction as simple as possible, we created a predefined configuration including a default driver profile and vehicle classes. They are kept simplistic, as the simulation algorithm was not the primary focus of this investigation. The user is able to create one or more entry points for simulations objects (i.e. vehicles) with predefined random distributions. To enable extensibility, every class in the framework is derived from a generic class. This allows the manual implementation/extension of almost every component like driving behavior or vehicle classes.

The simulation itself implements a simple vehicle following model and uses a time-stepped simulation approach. While simulation is handled on the server side, the user-interaction is handled client-sided through the web browser. The conceptual design of the web application is shown in Figure 8.

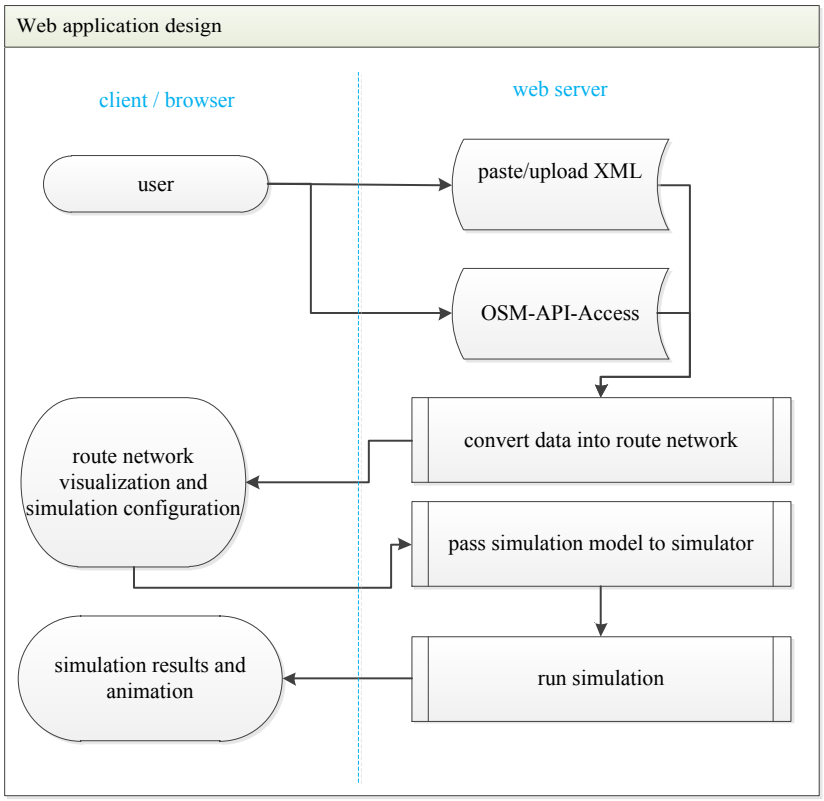

Figure 8: Conceptual web application design.

The simulation results are visualized in the web browser in a schematic visualization based on the generated graph structure of the route network (see Figure 9). It uses the canvas element of HTML5 for a client side generation of dynamic animations of moving entities, signals, etc. The implementation of the graph visualization is based on the SigmaJS framework (SigmaJS 2014). Furthermore several statistical key figures are presented using diagrams.

Finally, to demonstrate the concept of reverse geocoding information, we registered web service APIs by GeoNames.org and Google Maps. The prototype is capable of handling the service responses either in XML or JSON format. In future extensions this information could be used to have a direct effect on the underlying road network structure.

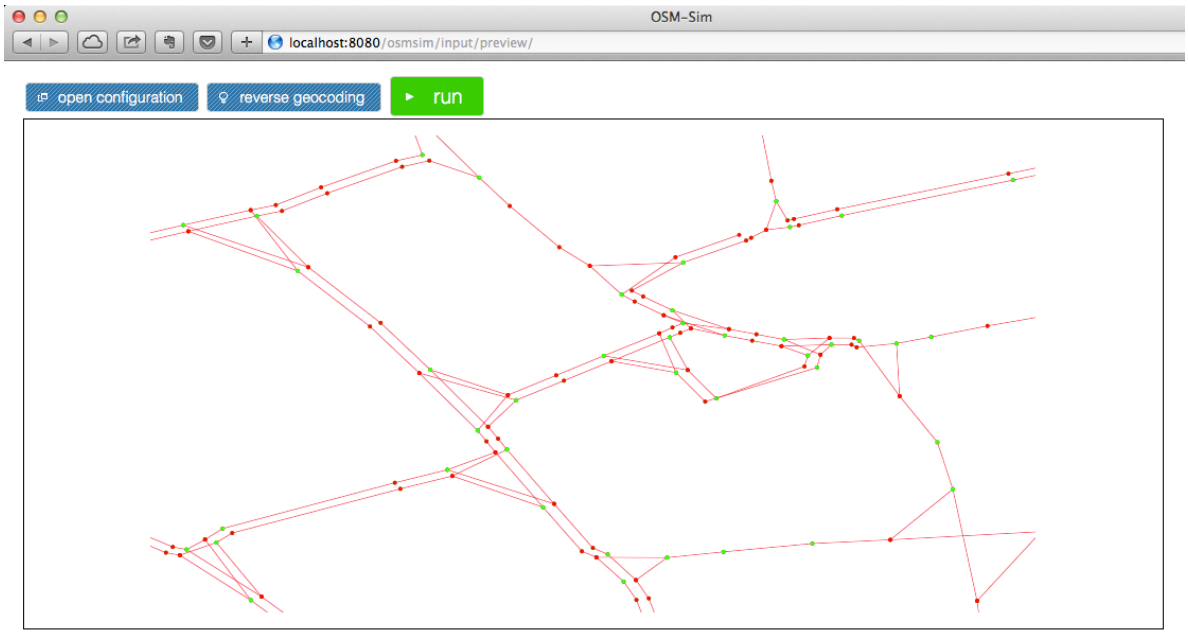

Figure 9: Visualization of the generated route network within the web browser.

6 CONCLUSIONS AND FUTURE WORK

We tested our prototypical implementation by inserting OSM extractions from multiple German cities. The prototype was able to cover the fundamental structures of the route network and to successfully run basic microscopic traffic simulations within the generated layout. Whereas overall data quality and correctness of the route network depends on the underlying OSM data, we were able to show that automatically creating a route network from a crowd-sourced data basis is possible. Due to the frameworks generic data structure, we were able to compensate missing information by utilizing default modeling assumptions.

Certain limitations resulting from the form of the input data exist. Most prominently, OSM data does not include information on multi-lane road layouts, which is mainly due to technical restrictions regarding the accuracy of GPS tracking devices.

Related to that issue, it is not possible to exactly retrieve the layout of intersections, for example when roads with multiple dedicated or mixed turning lanes come into play. Although our approach provides educated guesses to mitigate the effects of these issues, in many cases those details of a specific intersection can have significant influence on the results of a microscopic traffic simulation study.

We therefore suggest that it is advisable to combine the presented auto generating approach with adequate editing functionality in order to review and manipulate the generated road network manually. Further, editing functions come in handy when default assumptions need to be specialized or overwritten. However, even if a post processing through the end user is necessary, an auto generated foundation of the road network significantly reduces modeling effort compared to building the model from scratch in any case.

We further presented a concept of gaining additional information through reverse geocoding web services. One possible direction of future research may be the investigation on how such kind of information can be integrated in order to have tangible effects on the road network generation and simulation parameters.

REFERENCES

Bergmann, S., and S. Strassburger.2010. "Challenges for the Automatic Generation of Simulation Models for Production Systems." In: *Proceedings of the 2010 Summer Simulation Multiconference*, 545-549. July 11-14, 2010. Ottawa, Canada.

Burghout, W., H. N. Koutsopoulos, and I. Andreasson. "A Discrete-Event Mesoscopic Traffic Simulation Model for Hybrid Traffic Simulation." In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference*, 1102-1107. Toronto, Canada, September 17-20, 2006.

Lechler, T., and B. Page. 1999. "DESMO-J: An object oriented discrete simulation framework in Java." In *Proceedings of the 11th European Simulation Symposium*, edited by G. Horton, D. Möller, and U. Rüde, 46-50. Erlangen: SCS Publishing House.

Meyer, T., M. Trojahn, and S. Strassburger. 2013. "Using crowdsourced geographic information from OpenStreetMap for discrete event simulation of logistic systems." In *Proceedings of the 46th Annual Simulation Symposium,* 11-18. San Diego, CA, USA, April 7 - 10, 2013. Redhook, NY: Curran.

OSM Wiki. 2014. OpenStreetMap Editing API v0.6. Accessed March 27, 2014. http://wiki.openstreetmap.org/wiki/API_v0.6

Olstam, J.J., and A. Tapani. 2004. Comparison of Car-following models. Report of the Swedish National Road and Transport Research Institute. Accessed March 3, 2014. http://www.vti.se/en/publications/pdf/comparison-of-car-following-models.pdf

Page, B., and W. Kreutzer. 2005. The Java Simulation Handbook: Simulating Discrete Event Systems with UML and Java. Aachen: Shaker.

Schulze, T., and T. Fliess.1997. "Urban Traffic Simulation with psycho-physical vehicle-following models." In *Proceedings of the 1997 Winter Simulation Conference*, edited by S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, 1222-1229. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Schulze, T., S. Straßburger, and U. Klein. "On-line Data Processing in Simulation Models: New Approaches and Possibilities through HLA." In *Proceedings of the 1999 Winter Simulation Conference*, edited by P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, pp. 1602-1609. December 5-8, 1999. Phoenix, Arizona.

SigmaJS. 2014. JavaScript graph drawing library. Accessed March 31, 2014. http://www.sigmajs.org

Treiber, M. and A. Kesting. 2013. *Traffic Flow Dynamics.* Berlin, Heidelberg: Springer.

Wiedemann, R. 1974. "Simulation des Straßenverkehrsflusses." In: *Schriftenreihe des Instituts für Verkehrswesen der Universität Karlsruhe*, Heft 8, Karlsruhe, Germany.

AUTHOR BIOGRAPHIES

NICLAS FELDKAMP holds bachelor and master degrees in business information systems from the University of Cologne and the Ilmenau University of Technology, respectively. He is currently working as a doctoral student at the Department of Industrial Information Systems of the Ilmenau University of Technology. His email address is niclas.feldkamp@tu-ilmenau.de.

STEFFEN STRASSBURGER is a professor at the Ilmenau University of Technology and head of the Department for Industrial Information Systems. Previously he was head of the "Virtual Development" department at the Fraunhofer Institute in Magdeburg, Germany and a researcher at the DaimlerChrysler Research Center in Ulm, Germany. He holds a Ph.D. and a Diploma degree in Computer Science from the University of Magdeburg, Germany. He is further an associate editor of the Journal of Simulation. His research interests include distributed simulation, automatic simulation model generation, and general interoperability topics within the digital factory context. He is also the Vice Chair of SISO's COTS Simulation Package Interoperability Product Support Group. His web page can be found via www.tu-ilmenau.de/wi1. His email is steffen.strassburger@tu-ilmenau.de.