GIS-Based Traffic Simulation Using OSM

Jörg Dallmeyer, Andreas D. Lattner, and Ingo J. Timm

Abstract This chapter demonstrates how to build up a traffic simulation on the base of a Geographic Information System (GIS). Maps from the OpenStreetMap (OSM) initiative have shown to be appropriate for usage in this field. Essential steps from OSM over GIS to a graph data structure for use in traffic simulation are described. The work is done with the focus on urban scenarios. The crucial decision, which types of road users to integrate into a simulation and how to model them, is discussed. A case scenario shows the utility of data mining techniques in the field of traffic simulation. The scenario aims at predicting traffic jams in the city of Frankfurt am Main with help of a learned classifier. Our results show that taking into account simple and partial information about the traffic situation can lead to a huge gain of knowledge when using data mining techniques in the face of predicting of traffic situations.

Keywords Traffic simulation • Machine learning • Jam forecasting • Urban traffic

1 Introduction

Simulations are commonly used for understanding and prediction of traffic phenomena, traffic densities and traffic flows. It is much cheaper to test a new scenario in simulation before implementing it in reality. For a realistic simulation of urban

J. Dallmeyer (✉) • A.D. Lattner
Information Systems and Simulation, Institute of Computer Science, Goethe University Frankfurt, P.O. Box 11 19 32, 60054 Frankfurt, Germany
e-mail: dallmeyer@cs.uni-frankfurt.de; lattner@cs.uni-frankfurt.de

I.J. Timm
Business Informatics I, University of Trier, D-54286 Trier, Germany
e-mail: ingo.timm@uni-trier.de

G. Cervone et al. (eds.), *Data Mining for Geoinformatics: Methods and Applications*,
DOI 10.1007/978-1-4614-7669-6_4,

scenarios, two elementary problems need to be solved. At first, the road structure needs to be modeled in a simulation graph. Second, the behavior of road users and traffic rules need to be modeled. According to the focus of the simulation, additional information needs to be integrated into the model (e.g., a digital terrain model or a model for gas consumption and CO_2 emissions).

Today, for almost any major city, very detailed road models exist in layers for Geographic Information Systems (GIS). GIS are very powerful tools in order to deal with heterogeneous data sources particularly with regard to building road models. With help of the renderer of a GIS, it is easy to show the effects in simulation directly. Therefore, a couple of GIS layers, specifying different types of geometry (e.g., roads and polygons), are rendered on top of each other and a typical map is produced.

The simulation of multimodal traffic is a challenge for traffic simulation systems. Interdependencies from, e.g., bicyclists with cars or cars with pedestrians need to be modeled. Most traffic simulations therefore either simulate traffic in a very simplified way and cannot be adapted realistically for urban scenarios (e.g., Transims (Nagel et al. 1999), MATSim (Rieser 2010)) or are high fidelity simulations which are not sufficient for large scenarios (e.g., VISSIM (Fellendorf 1994)). Simulations differ widely in the needed time effort for setting up a simulation.

After having built a traffic simulation system, traffic engineers need to evaluate scenarios with help of the system. Different output values can be used in order to compare different scenarios (e.g., traffic flows, mean velocities, travel times, time spent in front of red traffic lights, number of accidents, ...). A manual analysis of such data is often not feasible because of the high amount of information and numerous attributes. Methods from machine learning can be used to find interesting rules like "When the traffic density in the area x is bigger than v, traffic will get stuck in area y". Apparently, such rules are not trivially to be generated, but steps into this direction have been taken (e.g., Lattner et al. 2011).

The *OpenStreetMap* (OSM) project has grown constantly in the last years and is today a useful source for road maps for different purposes. It would be a benefit for traffic research to build traffic simulation models upon OSM. OSM maps are not directly usable for GIS. Thus, a transformation step has to be established.

This chapter gives an introduction on how to use OSM data for traffic simulation (Sect. 2) and how to build up a simulation graph from GIS information (Sect. 3). It also introduces the developed traffic models (Sect. 4) and a short introduction to machine learning for traffic simulation is provided (Sect. 5). A case scenario is discussed (Sect. 6) and the chapter ends with a short summary of the content and perspectives for future investigations (Sect. 7). The discussed architecture and models are part of the traffic simulation system MAINS^{2}IM (MultimodAl INnercity SIMulation).[1]

[1] http://www.mainsim.eu

2 OpenStreetMap for Traffic Simulation

This section discusses OpenStreetMap[2] (OSM) and how to obtain data from the project. The design of cartographical material from OSM is described and an idea about how to extract important information from OSM is given.

OSM is a free of charge project giving access to cartographical material from all over the world. The maps are created by voluntary cartographers in the manner of Wikipedia. The data is licensed currently under Creative Commons Attribution-Share Alike 2.0 (CC-BY-SA),[3] but this could change in the future (Bennett 2010).

OSM has grown in the last years and has reached a level of detail and quality enabling it to be used for different purposes even though the data quality differs across different areas (Haklay 2010; Zielstra and Zipf 2010). For example, OSM maps are used in different smartphones for navigation purposes. In the area of Transportation Simulation, OSM is used as a data source in order to built up simulation graphs (Dallmeyer et al. 2011; Zilske et al. 2011).

OSM uses an XML-format. At first, an amount of `nodes` is defined. LineString and polygon geometries are built with help of `way` elements. These can be, e.g., roads, rivers, forests and local areas. `ways` store references to the `nodes` they consist of. The strengths of the format are its simplicity, memory efficiency, and extensibility. Additional information can be stored via usage of `tag` elements, which use a `key` and a `value`. In addition to `nodes` and `ways`, `relations` can be created, which store, e.g., bus routes or bicycle tracks. `relations` are always the last section of an OSM file. Figure 1 shows the basic elements.

OSM files can be exported directly from the OSM website. Only small areas can be exported. Another possibility is to download a map from a whole federal

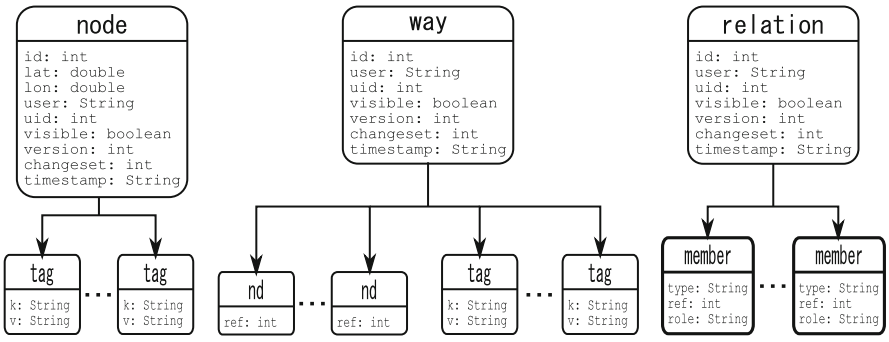

Fig. 1 Basic OSM elements

[2] http://www.openstreetmap.org

[3] http://www.openstreetmap.org/copyright, last visited December 15th, 2011

state from an external provider.[4] The files can easily have a size of more than 1 GB. Thus, we developed a way to clip the map by oneself to a polygon P. The strict order `nodes` → `ways` → `relations` can be used to do the clipping efficiently. The XML elements are parsed via a SAX parser and each node which is covered by P is stored in the clipped new output file. The remainder is dropped. Afterwards, each `way`, which contains only references to `nodes`, which are in the output file are attached to the clipped file, the rest is dropped. The same way, relations can be filtered.

In order to keep each `way` which is covered by P or crosses P, the OSM file has to be processed a second time to amend the `nodes` that have been dropped during the first parsing of the file though being used in one of the `way` elements, that are not fully covered by P.

For the purpose of traffic simulation, it is much easier to work with a layer based GIS than with a whole XML tree. This gives the advantages of a powerful rendering method for visualization of the simulation and to divide information which is used for simulation from information which is only used for rendering. Layers are a logical way of grouping associated information. The clipped OSM document is converted to a number of Shapefile layers.[5] This is done with help of the open source GIS toolkit *GeoTools*.[6]

GeoTools is written in Java. It provides classes for the handling of geoinformation and can be extended arbitrarily. In order to build up a traffic simulation system, it is necessary to extract a simulation graph data structure from the given geographical map. The interplay of different layers and specific characteristic for traffic simulation are discussed in the next section.

3 Graph Generation from GIS Layers

For the basic generation of a graph, *GeoTools* provides a tool to generate graphs from LineString geometries. The basic graph is not suitable for simulation because of specific characteristics of road networks, which are not covered by *GeoTools*, e.g., missing nodes at road intersections or faulty set nodes at the intersection between a bridge and a regular road. Thus, this section provides a compilation of analysis and modification steps to model whole map sections in a data structure `ExtendedGraph`.

As a first step, a faultless graph has to be extracted, taking account for bridges, tunnels, right of way rules, traffic circles, numbers of lanes and velocity restrictions (Sect. 3.1). Then, additional information has to be extracted from the calculated

[4] http://www.geofabrik.de/en/data/download.html

[5] Shapefile is a standard format for geographical information.

[6] http://www.geotools.org/

graph, assigning which road may be used by which type of road user, finding bus routes, calculating ways in the graph and assigning edge usage probabilities (Sect. 3.2).

3.1 Generating an ExtendedGraph

At first, all basic edges and nodes of the graph are converted into EdgeInformations (`EI`) and NodeInformations (`NI`), which use as much information from OSM as possible (e.g., number of lanes, type of road, velocity restrictions). Unavailable information is amended via lookup tables, with respect to the type of road. Typical values for a residential road would be #lanes $= 1$ and $v_{\max} = 30\ \mathrm{km} \cdot \mathrm{h}^{-1}$. The connections between `EI`s are set: each `EI` is connected to two `NI`s $\mathtt{NI}_a$ and $\mathtt{NI}_b$ and each `NI` has at least the connection to one `EI`.

`EI`s are grouped in a structure called EdgeInformationCollection (`EIC`), `NI`s in a NodeInformationCollection (`NIC`). The collections give methods for selecting specific elements and manipulating `EI`s and `NI`s for different purposes.

At the current stage, each road is modeled with one geometry. Roads do not cross each other when they proceed after the crossing point. Therefore, intersections between `EI`s are detected and the corresponding `EI`s are split up at the intersection point. A new `NI` connects the crossing parts.

We are using a graph structure which stores one `EI` per road segment. The stored `nodes` of an OSM `way` are sorted in the direction of the course of the road. If a road is oneway, this attribute will be stored in the corresponding `EI` and the `EI` can only be used from $\mathtt{NI}_a$ to $\mathtt{NI}_b$ and not in the opposite direction. This means that the position of a simulated road user ru on the road depends on the `NI`, ru has visited last. In OSM, small traffic circles or turning areas are often stored with only one edge. In this case, `EI`s exist which have $\mathtt{NI}_a = \mathtt{NI}_b$ and it is impossible for ru to decide in which direction it drives on the corresponding road. Thus, such `EI`s are split in the center of their geometries and new `NI`s are inserted.

`NI`s being connected to `EI`s representing bridges or tunnels are checked, whether they do connect `EI`s which are bridges or tunnels with `EI`s which are not. In this case, a new `NI` is inserted at the same position of the existing `NI` and the `EI`s which are bridges or tunnels are connected to the new `NI` and are removed from the old one. In our representation, crosswalks and traffic lights are stored in `NI`s and therefore, `EI`s are split whenever a traffic control has to be inserted. The corresponding positions are read from a GIS layer representing points.

Because $v_{\max}$ of each `EI` is set in relation to the type of road when no information is given, highways crossing towns result in higher values of $v_{\max}$ than in reality. From the layer of polygons, each inner-city area ica is checked for `EI`s being located in ica. The value of the maximum velocity is adjusted to the minimum of $v_{\max}$ and $50\,\mathrm{km} \cdot \mathrm{h}^{-1}$, which is a typical value for the standard inner-city velocity. `EI`s being located in parking lots are detected identically.

In urban scenarios, traffic circles have gained an important role. The problem at this point is, that in OSM not all traffic circles are marked as such and therefore an analysis step is necessary to detect traffic circles. All parts of the traffic circle have identical values in respect to name, road type, maximum velocity, number of lanes and traffic circles are always oneway streets. The geometries of the parts of a traffic circle form a circle in anticlockwise direction. With the determined characteristics, traffic circles can be found in the graph.

OSM does not store information about which type of road user may use a road. A further lookup table is used to define this information in relation to different road types. In urban scenarios, the right of way is regulated by the priority of the road and the rule "right before left".[7] Each `NI` stores information about all connections over this `NI`. A connection has a direction (left, right, ahead) and an angle. The priority of a road is stored in the corresponding `EI` with help of an additional lookup table for the type of the `EI`. The higher the priority value, the higher the right of way of `EI`. The highest priority is given to traffic circles without respect to their road type.

Up to this point, a consistent `ExtendedGraph` is built. The following subsection discusses additional information, which needs to be calculated, according to the subject of simulation.

3.2 Determining Additional Information

Bus routes can be stored in OSM via usage of relations. The problem at this point is, that routes are usually not stored from a starting point to an end point under usage of way objects in the sequential arrangement of their appearance on the route. But at least, the used ways can be read out and a heuristic algorithm which rearranges the parts of the route to a plausible aggregate route can be applied. Buses can be used in the simulation to let groups of pedestrians appear at bus stops and also for simulating public transportation where buses stop at bus stops. This leads to abruptly changing traffic situations in areas surrounding bus stops, because of pedestrians using a crosswalk, pushing the button of a traffic light or crossing the road at an insufficient gap.

Assuming that the simulation graph will not alter significantly during the simulation, all possible routes in the simulation graph are calculated. From each $\texttt{NI}_{\text{start}} \in \texttt{NIC}$, Dijkstra's algorithm (e.g., Cormen et al. 2001) is performed. The ID of the first $\texttt{NI}_1$ on the way from $\texttt{NI}_{\text{start}}$ to each $\texttt{NI}_{\text{dest}} \in \texttt{NIC} \setminus \texttt{NI}_{\text{start}}$ is stored. After performing this algorithm, each $\texttt{NI}_{\text{start}}$ holds an array, where the n-th entry is the ID of the first `NI` on the way to the $\texttt{NI}_{\text{dest}}$ with ID n.

[7] The described method implements rules for right hand traffic. For left hand traffic, the right of way priorities need to be switched.

The method is performed for the three basic routing characteristics “car”, “bicycle” and “pedestrian”. Space reduction is achieved due to creating lookup tables for each `NI`, giving the stored IDs shorter values. This is possible due to the fact, that each `NI` has only as much different IDs to store, as it has connected `EI`s.

In order to have random behavior in routing, the calculated routes are passed to an analysis step. All routes are run from $\texttt{NI}_{\text{start}}$ to $\texttt{NI}_{\text{dest}}$ and each `NI` counts, how often its connected `EI`s is used on the runs. Edge usage probabilities are set up with respect to these counts. Three different routing mechanisms can now be used during simulation: Precalculated routes, online calculated routes (which need more computational time, but can deal with changes in the `ExtendedGraph`) and random walks with respect to precalculated edge usage probabilities (Dallmeyer et al. 2011).

The computed `ExtendedGraph` can be stored into a file and directly used as input for simulation runs.

After having calculated the `ExtendedGraph`, it is necessary to decide, which kind of road users to simulate and how to model, e.g., cars and pedestrians. The next section addresses this crucial factor.

4 Simulation Models

The focus of this work is the simulation of multimodal traffic in urban scenarios. Each group of road users needs different behavioral models. This section gives a brief introduction of how our models work. It is important to model cars (e.g., passenger cars, trucks and buses; Sect. 4.1), bicycles (Sect. 4.2) and pedestrians (Sect. 4.3).

4.1 Space Continuous Car Model

The simulation of car traffic is often done with help of the Nagel-Schreckenberg model (NSM) (Nagel and Schreckenberg 1992), which is a cellular automaton model for freeway traffic. It is the de facto standard, because of its simplicity and, nevertheless, capability to model freeway traffic in a realistic way with respect to macroscopic properties. NSM divides a road into cells with length 7.5 m. Cars may have discrete velocities in the range $v \in \{0 \cdots 5\} \left[\text{cell} \cdot \text{s}^{-1}\right]$. The model is discrete in time. One simulation iteration corresponds to 1 s real time.

Each simulated car performs the following steps in parallel. At first, try to accelerate ($v \leftarrow v + 1$), if $v < 5$. Determine the gap γ to the preceding car. Set $v \leftarrow \min(v, \gamma)$. Dally with probability p. When dallying, set $v \leftarrow \max(0, v - 1)$.

It was shown that emerging phenomena through interdependencies of many cars like *phantom jams*[8] can be simulated with NSM. Furthermore, the model could be calibrated to measurement data from real freeways. Many work has been done in the community of traffic simulators, e.g., extending the model with brake lights (Hafstein et al. 2003), a slow start rule (Helbing 1997), smaller cell sizes (Krauss et al. 1996) and multi lane traffic (Knospe et al. 2002).

Although NSM provides many extensions, it has been reasonable to remove the cells from the roads and to build a space continuous car model which uses the basic functionalities of NSM, but being able to build arbitrary acceleration and deceleration functions. One main reason for space continuity are different velocity spreadings for different types of road users. For example, a bicycle usually moves much slower than a car. Any length of car can be modeled. Interaction on the roads can be simulated in more detail. This could only be done in NSM when using very small cells, but this would decrease the computational advantages of NSM significantly.

The model uses the same update steps as NSM but with variable velocity functions. Different modifications and enhancements, described in literature are built in our model, too. We could show, that the basic model reproduces macroscopic data on freeways (Dallmeyer et al. 2011). Cars are able to change lanes and to choose lanes for turning maneuvers.

4.2 Bicycle Model

Bicycles are currently not under detailed investigation in science. But there are publications about how bicycles behave in urban traffic (e.g., Johnson et al. 2011; Larsen and El-Geneidy 2011). The investigated bicycle model is built on determined velocity distributions. Bicycles have mainly the same behavioral model as cars, but with different dallying properties and, of course, different routes. Bicycles are interesting for traffic simulation because of the interaction with cars.

Cars may overtake bicycles when it is safe. Each car determines, what is seen to be *safe*. Overtaking is done with respect to the desire to overtake, the gap to oncoming traffic, the possibility to get back to the right lane after overtaking (there needs to be a gap, too), the distance to the next road junction and the width of the road (with respect to the width of the car, the oncoming car and the bicycle).

Another occurrence in urban traffic is the pushing to the front of cyclists to red traffic lights, overtaking on the left, on the right or both. This seems to be intended by traffic planners in some cases (e.g., if there is a specific region for cyclists in front of the stop line for cars). When the light switches to green, cars need to overtake the bicycles again. Not all bicycles overtake at red lights. This can be modeled with help of a probability function.

[8] A phantom jam is a jam which occurs without an obvious reason. It is build up from dallying and overdosed braking in NSM and has likewise reasons, in reality (Helbing 2001).

4.3 Pedestrian Model

The simulation of pedestrians has been given little attention in the past (Ishaque and Noland 2008), even though every human is a pedestrian. Pedestrians walk on sidewalks and cross roads, when it is necessary. Traffic lights and crosswalks are preferred for crossing the road (crossing at `NIs`) compared to crossing without right of way (crossing at `EIs`). It is assumed that pedestrian velocity will average over time and thus, that the mutual influence of pedestrians while walking on sidewalks (`EIs`) can be neglected. It is more interesting to determine, how long the crossing of a road will take. A model accounting for interaction between pedestrians is used for crossing roads at `NIs`.

The cellular automaton model for pedestrian movement, presented in Blue and Adler (2001) is used as basis for a space continuous model enabling pedestrians to avoid other pedestrians at `NIs`, walk with velocities in relation to the free space on `NIs` and to choose lanes. For example, a crosswalk has a width w_c and each pedestrian has a width w_p. The crossing then has $\left\lfloor w_c \cdot w_p^{-1} \right\rfloor$ lanes. The underlying principles are used for individual velocities with respect to different types of pedestrians (e.g., children, grown-ups and seniors). The velocities of pedestrians vary over different situations (e.g., walking on the sidewalk, crossing a road without right of way or crossing the road on a crosswalk).

Each simulated pedestrian accepts individual gaps in road traffic for crossing a road. This is done after the concept of Estimated Crossing Time (ECT) (Ottomanelli et al. 2009), which takes into account of individual aggressiveness of pedestrians. It may happen that a pedestrian misjudges a gap and is not able to finish crossing a road before a car or bicycle arrives at its position. Pedestrians are visible for road traffic only when crossing a road and then are treated like normal standing cars. An approaching car will slow down and in extreme cases come to standstill until a crossing pedestrian has left the road.

On the other hand, pedestrians cross `NIs`. They can do so without having the right of way. They decide whether crossing is appropriate with respect to traffic. Whenever a pedestrian is crossing an `NI` at one side, all road users planning to cross this side, will wait until the crossing is done. Pedestrians pass crosswalks without respect to traffic. Pedestrians are able to virtually push the buttons of traffic lights, forcing them to turn red for road traffic during the next simulated minute. The mentioned interactions influence urban traffic. A detailed description of the pedestrian model will be available in a separate publication (Dallmeyer et al. 2012a).

Different parameters can be measured during simulation runs, e.g., average velocities, traffic densities or traffic flows. In order to spot relationships between different parameters, appropriate techniques have to be applied to the system. The next section gives a brief introduction on machine learning and how to use it for traffic simulation.

5 Learning in Traffic Scenarios

Different learning paradigms can be applied to simulated traffic scenarios in order to identify regularities or useful behaviors. In this section, we discuss different options following the classification of learning approaches regarding learning feedback, namely supervised learning, unsupervised learning, and reinforcement learning (see, e.g., Russell and Norvig 2003).

The evaluation scenario in the subsequent section addresses the question how to predict if a congestion occurs at a certain road segment using information about the number of road users at different regions in the city. We also refer to this example in order to illustrate the different learning approaches.

In supervised learning, the desired output of the concept to be learned is known in advance. In the case of supervised learning from examples, for instance, each example in the training data consists of a set of features as well as a corresponding class. Referring the aforementioned scenario, features can be, e.g., the number of cars in different residential areas and the target class, if this situation has lead to a congestion at a specific road segment. The learning task is to generate a hypothesis which – hopefully – represents well the true underlying concept of the data. Learning results in a classifier which can be used to classify (previously unseen) examples. Having captured the number of cars in the different residential areas, the classifier can predict if a congestion is expected at the point of interest. It is aimed at generating a general classifier which does not only cover the training data well but additionally performs well on unseen examples. Approaches to supervised learning are, among others, decision tree learning, decision rule learning, learning in neural networks, and Bayesian learning. We focus here on symbolic learning approaches (decision tree learning and decision rule learning) as the generated classifiers lead to a comprehensible representation. Figure 2 illustrates the process of supervised learning from traffic simulation experiments.

In contrast, in unsupervised learning no information about the desired output is provided. The learning task is to identify regularities in the data (without target concept) or to group examples with respect to their similarity. In the first case, (sequential) association rules can be identified (e.g., Agrawal and Srikant 1994; Pei et al. 2001): If there is a high traffic density in area 1 and area 2, there will also be a high traffic density in area 3. In the latter case, the data would be grouped in a way that a classification scheme is generated, for instance, grouping similar traffic situations using a specified similarity measurement. This process is called clustering (see, e.g., Jain et al. 1999). Getting an overview of different groups in the data can help to analyze and to come up with certain strategies for the identified groups.

No direct feedback is provided to the learner in reinforcement learning. The general setting in reinforcement learning is, that an agent can perceive the current situation and can decide what action to take. In dependence of the actions, a reward will be provided to the agent. However, this reward is not necessarily instantaneous and might also depend on further activities or events beyond control of the agent. Referring to the traffic scenario once again, a reinforcement learning setting could

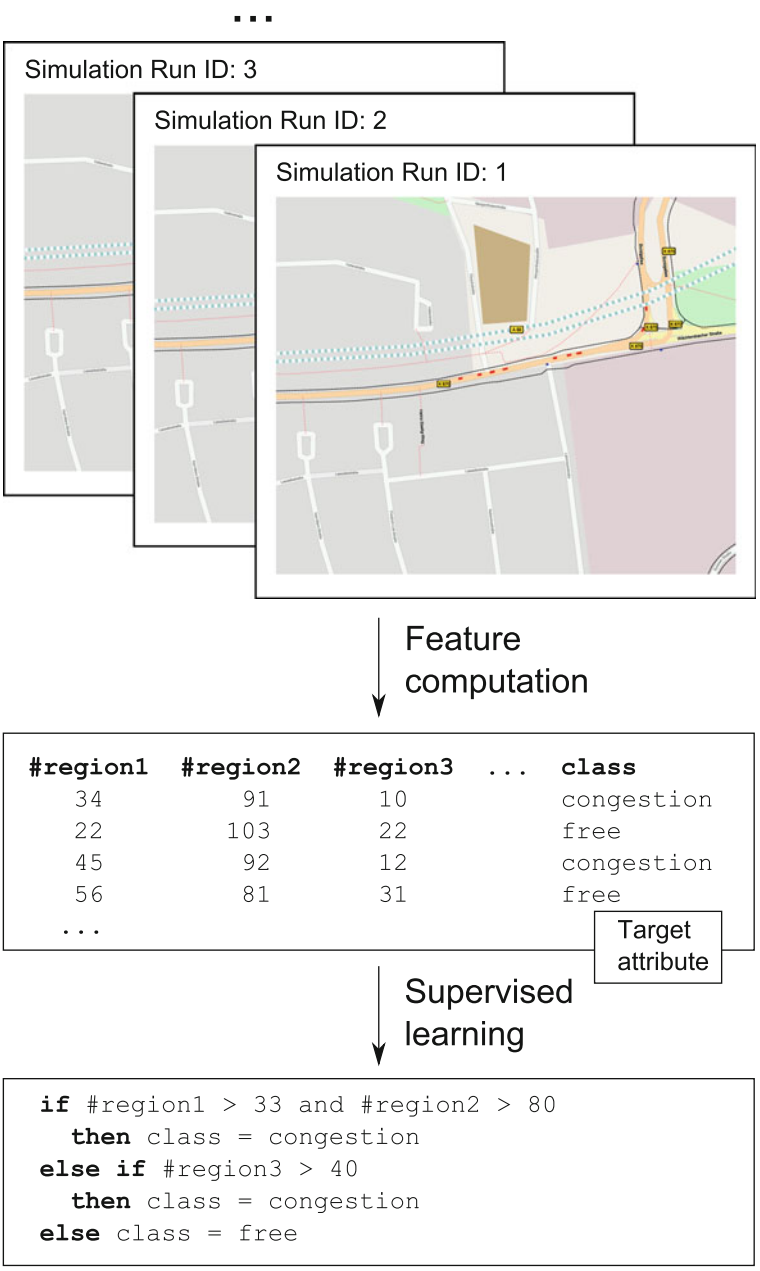

Fig. 2 Supervised learning in traffic simulation

be to reach a target position as fast as possible. Perceiving the current traffic situation (maybe even about remote positions via radio traffic service) then provides the basis for the decision how to behave, e.g., to keep the route or to re-plan. A discussion of reinforcement learning approaches in the context of traffic scenarios can be found, e.g., in Bazzan (2009).

In this work, we focus on the first learning paradigm, namely supervised learning. In the following section, we describe the supervised learning task as well as experimental results.

6 Case Scenario

The focus of this work is to generate understandable rules to predict situations, when traffic will get stuck. Figure 3 shows an excerpt of Frankfurt am Main. In the morning, most traffic pours into the city from the motorway A66 in the east of Frankfurt am Main. The area where cars enter the city is magnified. Most cars drive to the direction of the city center which results in a huge amount of cars turning left at this point. A part of the cars parks in a park-and-ride (P+R) parking garage, also magnified. Most of the cars drive on to the third magnified area, presented dashed. This area often has slow-moving traffic.

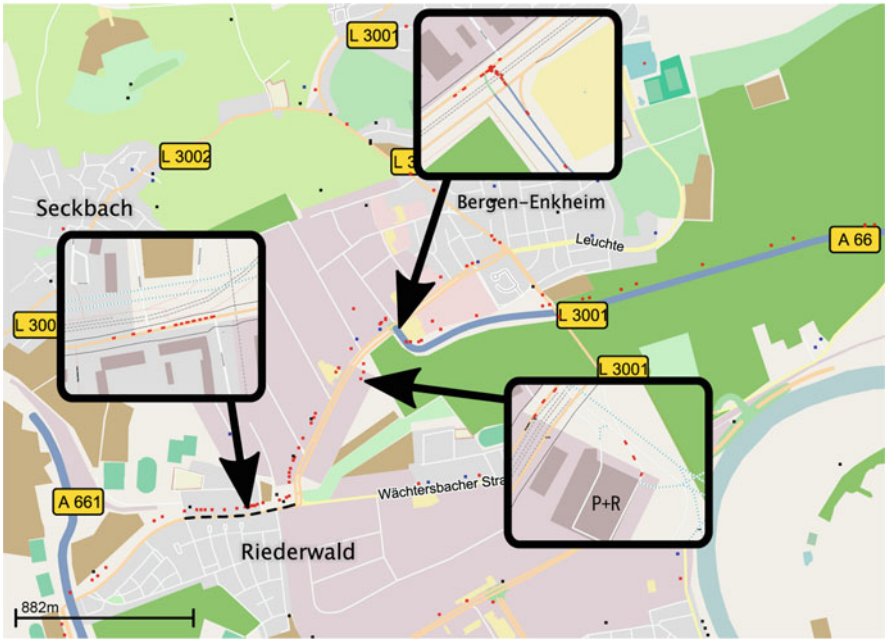

Fig. 3 Case Scenario: OSM excerpt of Frankfurt am Main. The area of velocity measurement is shown *dashed*

This results from a huge amount of unsynchronized traffic lights and the exceeded road capacity limit in this area. The traffic flow is influenced by pedestrians pushing the buttons of traffic lights. It is assumed to be possible to count the number of cars in a given area. This could be done with help of induction loops at central crossing points and extrapolation of these counts for a whole area. It is more difficult to count the number of bicycles, because of the possibility to additionally use cycle ways and paths. Pedestrians cannot be counted easily. It would be possible to count how often pedestrians push the buttons of traffic lights during a given time interval in a defined area.

In the case scenario, only cars are counted. The urban districts "Bergen-Enkheim", "Seckbach" and "Riederwald" are observed independently. In addition, the motorway A66 in the direction of the city of Frankfurt am Main gets observed. After a warm-up phase of the simulation (2,000 iterations), the number of cars in the measurement areas are counted ($\#Riederwald, \#Bergen-Enkheim, \#A66, \#Seckbach$) and the sum of them is calculated. These values are used as snapshots. In traffic scenarios, *time* is always an important parameter. The situation at a time t might influence the situation at time $t + \Delta$ in another region. The value of Δ is chosen $\Delta = 300\,\mathrm{s}$, which is 5 min real-time. The value is chosen arbitrarily and is used as a first try, estimating the travel time from the areas of measurement to the point of interest. After Δ, the mean velocity $\overline{v}$ of all cars in the dashed jamming area is measured over a time of 60 s (1 min). Note that the traffic conditions of the dashed road are not part of the above traffic counts.

Pedestrians and bicycles influence the outcome of the simulation. Nevertheless, they are not observed for the calculation of the situation description for learning the classifier. The main function of these groups of road users is to increase the diversity of results in the training data and to model the most important types of road users occurring in reality.

Two classes are used to train a classifier.

$$class = \begin{cases} \text{congested} & \text{if } \overline{v} < 3.6\ \mathrm{m} \cdot \mathrm{s}^{-1} \\ \text{free} & \text{else} \end{cases}$$

We performed 20,000 simulation runs to train a classifier and 5,000 to measure its performance on unseen data. To define the settings of each simulation run, let $\mathcal{N}_a^b(\mu, \sigma) = \min(\max(\mathcal{N}(\mu, \sigma), a), b)$ be Gaussian distributed random number with μ and σ bounded to the interval $[a \cdots b]$. Table 1 shows the simulation parameters, calculated prior to each simulation run.

Figure 4 shows the distribution of training data. The data is separable and thus a classifier should be able to determine the classes.

The case scenario is a supervised learning scenario. The free machine learning software *WEKA*,[9] is used to analyze the training data and to build classifiers.

[9] http://www.cs.waikato.ac.nz/ml/weka/, see also Bouckaert et al. (2010).

Table 1 Different probabilities for each simulation run resulting in different volumes of traffic

$p_{\text{create}} = \mathcal{N}_{0.3}^{0.7}(0.5, 0.1)$	Probability to create a road user for each simulation iteration
$p_{\text{car}} = \mathcal{N}_{0.5}^{0.8}(0.75, 0.1)$	Probability that a created road user is a car
$p_{\text{bicycle}} = \mathcal{N}_{0.5}^{0.9}(0.75, 0.1)$	Probability, that it is a bicycle, when it is no car. Otherwise, it will be a pedestrian
$p_{\text{A66}} = \mathcal{N}_{0.67}^{0.9}(0.75, 0.1)$	Probability, that car will be put on the motorway A66
$p_{\text{fixDest}} = \mathcal{N}_{0.67}^{0.88}(0.75, 0.1)$	Probability to have one of the predefined destinations, otherwise the destination is a random point in the city
$p_{\text{destInnerCity}} = \mathcal{N}_{0.7}^{0.9}(0.8, 0.05)$	When having one of the predefined destinations: Probability to have the destination in the inner city of Frankfurt am Main, leading to use the dashed road. Otherwise the destination will be the P+R parking garage

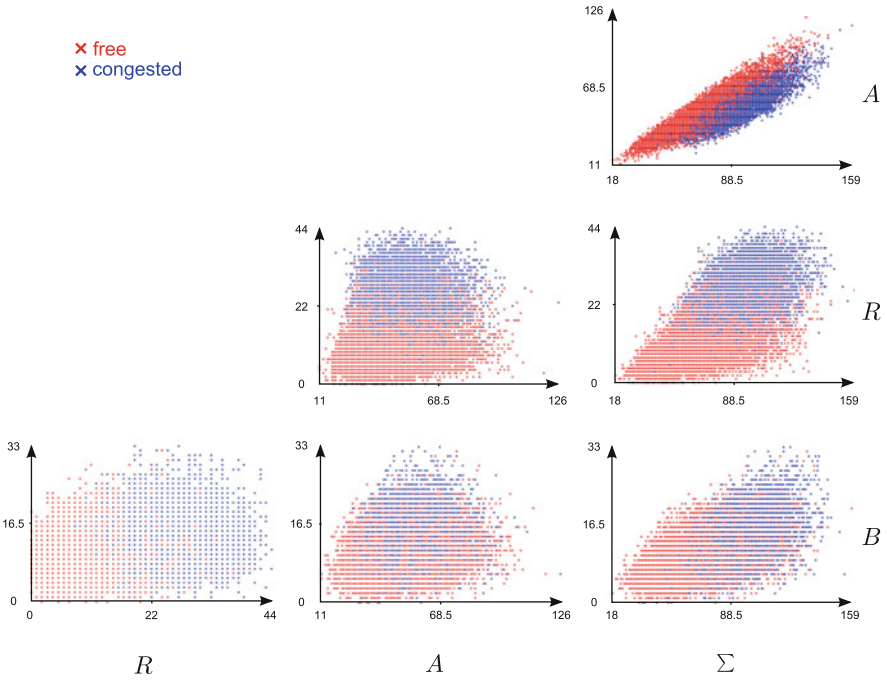

Fig. 4 Distribution of training data

For learning, we use WEKA's J4.8 implementation of the C4.5 algorithm (Quinlan 1993) and WEKA's JRip implementation of RIPPER (Cohen 1995), as they produce comprehensible classifiers.

The trained classifiers are shown in Figs. 5 and 6. RIPPER and C4.5 present the results in different ways. RIPPER gives one logical formula, presented in a summarized version and C4.5 generates a decision tree.

$$
\begin{aligned}
\text{congested} \;=\; & (R \geq 22) \\
& \vee (R \geq 19) \wedge (B \geq 10) \\
& \vee (R \geq 17) \wedge (B \geq 15) \\
& \vee (R \geq 21) \wedge (B \geq 6) \wedge (43 \leq A \leq 51) \\
& \vee (R \geq 20) \wedge (B \geq 7) \wedge (58 \leq A \leq 64) \wedge (\Sigma \geq 72) \\
& \vee (R \geq 16) \wedge (B \geq 8) \wedge (42 \leq A \leq 50) \wedge (S \leq 0) \wedge (72 \leq \Sigma \leq 75) \\
& \vee (R \geq 14) \wedge (B \geq 12) \wedge (A \leq 62) \wedge (S \geq 1) \wedge (\Sigma \geq 91) \\
& \vee (14 \leq R \leq 16) \wedge (11 \leq B \leq 17) \wedge (A \leq 50) \wedge (S \leq 1) \wedge (\Sigma \geq 78)
\end{aligned}
$$

$$
\begin{aligned}
\text{with } R &= \#\mathit{Riederwald} \\
B &= \#\mathit{Bergen{-}Enkheim} \\
A &= \#\mathit{A66} \\
S &= \#\mathit{Seckbach} \\
\Sigma &= B + R + S + A
\end{aligned}
$$

Fig. 5 Learned decision rule using RIPPER

GIS-based Traffic Simulation using OSM

```
Riederwald <= 16: free
Riederwald > 16
|  Riederwald <= 22
|  |  Bergen-Enkheim <= 14
|  |  |  Riederwald <= 19
|  |  |  |  Bergen-Enkheim <= 9: free
|  |  |  |  Bergen-Enkheim > 9
|  |  |  |  |  Seckbach <= 1
|  |  |  |  |  |  Riederwald <= 18: free
|  |  |  |  |  |  Riederwald > 18
|  |  |  |  |  |  |  Bergen-Enkheim <= 13: free
|  |  |  |  |  |  |  Bergen-Enkheim > 13: congested
|  |  |  |  |  Seckbach > 1
|  |  |  |  |  |  Riederwald <= 18
|  |  |  |  |  |  |  A66 <= 50
|  |  |  |  |  |  |  |  sum <= 74
...
```

Fig. 6 Excerpt of the learned tree using C4.5

Both classifiers perform well on unseen data. Table 2 displays the confusion matrices of the learned classifier on the test set of 5,000 further simulation runs. The results are on par with each other: RIPPER achieves a predictive accuracy of 83.66% and C4.5 of 83.48%, respectively. The classifiers perform better than the simple solution to always choose "free" as a result, because it is the class with the highest frequency of occurrence, leading to a performance of 58.38%.

The small example shows the feasibility to use data mining techniques for traffic scenarios. In a next step, the cars coming from the motorway A66 and from Bergen-Enkheim could be advised to change destination to the P+R parking garage, when it is likely that a traffic jam will occur.

Table 2 Confusion matrices of the trained classifiers gained on the test set

RIPPER			C4.5		
Classified as →			Classified as →		
Correct result ↓	Congested	Free	Correct result ↓	Congested	Free
Congested	1,768	313	Congested	1,731	350
Free	504	2,415	Free	476	2,443
Correctly classified: 4,183 (83.66%)			Correctly classified: 4,174 (83.48%)		

7 Summary and Perspectives

An introduction on how to build up an executable traffic simulation from OSM cartographical material has been presented. The use of GIS technologies offers the opportunity to enhance the simulation model with data from various sources. For example, a digital terrain model could influence the routing of bicycles or a layer giving information about rain on the map could decrease the amount of bicycles and pedestrians in the simulation and make the cars drive more slowly.

The requirements for simulation of urban traffic have been discussed and three basic types of road users – car, bicycle and pedestrian – have been identified. Microscopic simulation models taking account for the interactions between these groups have been presented.

A case scenario has shown a method how to predict traffic jams in urban scenarios on the example of Frankfurt am Main. Machine learning techniques have been applied to train a classifier on the results of simulation runs. The classifier performs well on unseen data (approximately 83.5% correctly classified samples).

As one next step, a model for gas consumption and CO_2 emission can be applied to the car model as proposed in Dallmeyer et al. (2012b). Then, the influence of bicycles and pedestrians on this dimension could be analyzed. The influences of actions like advising to park the car in a P+R parking garage on traffic jams and on gas consumption then could be investigated.

Different types of classification algorithms could be compared in future work. It might also be advantageous to use different intervals than $\Delta = 300\,\text{s}$.

Acknowledgements This work was made possible by the *MainCampus* scholarship of the *Stiftung Polytechnische Gesellschaft Frankfurt am Main*.

References

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th international conference on very large data bases, VLDB, Santiago de Chile, Sept 1994, pp 487–499

Bazzan ALC (2009) Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Auton Agents Multi-Agent Syst 18:342–375

Bennett J (2010) OpenStreetMap. Packt Publishing, Olton Birmingham, GBR. ISBN:978-1-84719-750-4

Blue VJ, Adler JL (2001) Cellular automata microsimulation for modeling bi-directional pedestrian walkways. Transp Res Part B Methodol 35(3):293–312

Bouckaert, RR Frank E, Hall MA, Holmes G, Pfahringer B, Reutemann P, Witten IH (2010) WEKA – experiences with a Java open-source project. J Mach Learn Res 11:2533–2541

Cohen WW (1995) Fast effective rule induction. In: Proceedings of the 12th international conference on machine learning, Lake Taho

Cormen TH, Stein C, Rivest RL, Leiserson CE (2001) Introduction to algorithms, 2nd edn. McGraw-Hill, New York

Dallmeyer J, Lattner AD, Timm IJ (2011) From GIS to mixed traffic simulation in Urban scenarios. In: Liu J, Quaglia F, Eidenbenz S, Gilmore S (eds) 4th international ICST conference on simulation tools and techniques, SIMUTools'11, Barcelona, 22–24 Mar 2011. ICST, Brüssel, pp 134–143. ISBN:978-1-936968-00-8

Dallmeyer J, Lattner AD, Timm IJ (2012a) Pedestrian simulation for urban traffic scenarios. In: Bruzzone AG (ed) Proceedings of the summer computer simulation conference 2012. 44rd summer simulation multi-conference, Genoa, 8–11 July 2012, S. 414–421. Curran Associates Inc

Dallmeyer J, Taubert C, Lattner AD, Timm IJ (2012b) Fuel consumption and emission modeling for urban scenarios. In: Troitzsch KG, Möhring M, Lotzmann U (eds) Proceedings of the 26th European conference on modelling and simulation (ECMS 2012), Koblenz, pp 574–580

Fellendorf M (1994) Vissim: a microscopic simulation tool to evaluate actuated signal control including bus priority. In: 64th institute of transportation engineers annual meeting, Dallas

Hafstein SF, Pottmeier A, Wahle J, Schreckenberg M (2003) Cellular automaton modeling of the autobahn traffic in north rhine-westphalia. In: Proceedings of the 4th MATHMOD, Vienna, pp 1322–1331

Haklay M (2010) How good is volunteered geographical information? A comparative study of OpenStreetMap and ordnance survey datasets. Environ Plan B Plan Des 37(4):682–703

Helbing D (1997) Empirical traffic data and their implications for traffic modeling. Phys Rev E 55(1):R25–R28

Helbing D (2001) Traffic and related self-driven many-particle systems. Rev Mod Phys 73(4):1067–1141

Ishaque MM, Noland RB (2008) Behavioural issues in pedestrian speed choice and street crossing behaviour: a review. Transp Rev Transnatl Transdiscipl J 28(1):61–85

Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323

Johnson M, Newstead S, Charlton J, Oxley J (2011) Riding through red lights: the rate, characteristics and risk factors of non-compliant urban commuter cyclists. Accid Anal Prev 43(1):323–328

Knospe W, Santen L, Schadschneider A, Schreckenberg M (2002) A realistic two-lane traffic model for highway traffic. J Phys A Math Gen 35(15):3369–3388

Krauss S, Wagner P, Gawron C (1996) Continuous limit of the nagel-schreckenberg model. Phys Rev E 54(4):3707–3712

Larsen J, El-Geneidy A (2011) A travel behavior analysis of urban cycling facilities in Montréal, Canada. Transp Res Part D Transp Environ 16(2):172–177

Lattner AD, Dallmeyer J, Timm IJ (2011) Learning dynamic adaptation strategies in agent-based traffic simulation experiments. In: Klügl F, Ossowski S (eds) Ninth German conference on multi-agent system technologies (MATES 2011). LNCS 6973. Springer, Berlin, pp 77–88

Nagel K, Schreckenberg M (1992) A cellular automaton model for freeway traffic. J Phys I 2(12):2221–2229

Nagel K, Beckman RJ, Barrett CL (1999) Transims for urban planning. Los Alamos unclassified report LA-UR 98-4389

Ottomanelli M, Caggiani L, Giuseppe I, Sassanelli D (2009) An adaptive neuro-fuzzy inference system for simulation of pedestrians behaviour at unsignalized roadway crossings. In: 14th online world conference on soft computing in industrial application, 14. http://link.springer.com/content/pdf/10.1007%2F978-3-642-11282-9_27.pdf

Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M-C (2001) PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 12th IEEE international conference on data engineering, Heidelberg, pp 215–224

Quinlan JR (1993) C4.5 – programs for machine learning. Morgan Kaufmann, San Mateo

Rieser M (2010) Adding transit to an agent-based transportation simulation concepts and implementation. Dissertation, Technische Universität Berlin

Russell S, Norvig P (2003) Artificial intelligence: a modern approach, 2nd edn. Prentice Hall/Pearson Education, Upper Saddle River

Zielstra D, Zipf A (2010) A comparative study of proprietary geodata and volunteered geographic information for Germany. In: 13th AGILE international conference on geographic information science 2010, Guimarães

Zilske M, Neumann A, Nagel K (2011) OpenStreetMap for traffic simulation. In: State of the map Europe (SOTM-EU), Vienna