An Example of Microscopic Car Models Validation using the open source Traffic Simulation SUMO

Daniel Krajzewicz, Georg Hertkorn and Peter Wagner
German Aerospace Centre,
Institute for Transportation Research
Rutherfordstr. 2, 12489 Berlin, Germany
E-Mail: Daniel.Krajzewicz@dlr.de,
Georg.Hertkorn@dlr.de, Peter.Wagner@dlr.de

Christian Rössel
Centre for Applied Informatics Cologne
Weyertal 80, 50931 Köln, Germany
E-mail: roessel@zpr.uni-koeln.de

KEYWORDS
traffic simulation, road traffic, car following, model validation, microscopic, continuous, multi-modal, open source, car-driver model, traffic research

ABSTRACT

In [1] we presented an open source simulation software for road traffic simulation. Now we show one possible field of application, the validation of microscopic car/car-driver models. Our motivation is to awake the interest in using and extending the software, so this report will describe the software's usability but will not go into depth in interpreting the results.

INTRODUCTION

Traffic research as well as the process of traffic planning need tools to describe and forecast traffic and its changes. As traffic is influenced by many factors, some of which include random elements, such as the process of driving a car itself, one of the main tools used here are simulations.
Most traffic simulation software packages are 'black boxes' and do not allow to change the models they use or even their parameters. In contrast, the software named SUMO ("Simulation of Urban MObility") is distributed as open source, so everyone may change, extend and/or use it.
We give at first a short overview over the capabilities of the software, the models we use, the modules the software includes and then report about one of our current projects, the validation of the Krauß (Krauß 1998; Janz 1998) – car/driver model using SUMO.

THE SOFTWARE

Main Principles

SUMO is a microscopic, space continuous and time discrete traffic simulation.
In traffic research three or four classes of models are distinguished according to the level of detail of the simulation. In 'Macroscopic' models traffic flow is the basic entity. 'Microscopic' models, simulate the movement of every single vehicle on the street – mostly assuming that the behaviour of the vehicle depends on both, the vehicle's physical abilities to move and the driver's controlling behaviour (Krauß 1998; Janz 1998). 'Submicroscopic' models, also sometimes called 'nanomodels' regard single vehicles like microscopic simulations, but submodels are included, that describe the engine's rotation speed in relation to the vehicle's speed or the driver's preferred gear switching actions, for instance. This allows more detailed computations of the emissions produced by the vehicle compared to a simple microscopic simulation (Diekamp 1995; Schreckenberg and Wolf 1998; Helbig et al. 2001). However, submicroscopic models require large computation times. This restrains the size of the networks to be simulated.

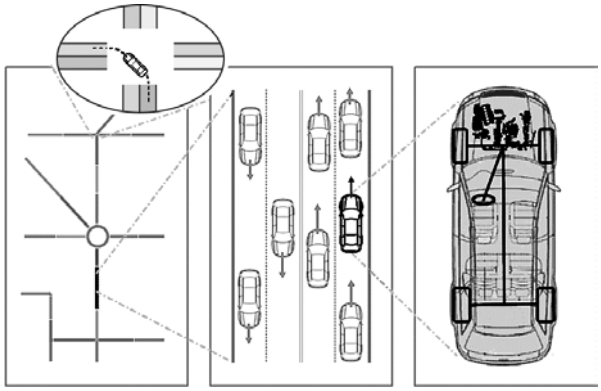

Figure 1: Different simulation classes (from left to right: macroscopic, microscopic, sub-microscopic; within the circle: mesoscopic)

A vehicle within a space-continuous simulation has a certain position described by a floating-point number. In contrast, space-discrete simulations are a special kind of cellular automata. They use cells and vehicles driving on the simulated streets "jump" from one cell to another (Brockfeld et al 2001)
In the near future, SUMO will be extended by other transport modes: busses and trains. Such simulations are called multi-modal. Then, the elementary part of the simulation will be the individual, who is described by a departure time and the route he/she takes, instead of vehicles as in pure traffic simulations with only one traffic mode.

Features

In the current version – 0.7 – SUMO has the following features:

- collision free vehicle movement
- different vehicle types
- multi-lane streets with lane changing
- junction-based right-of-way rules
- lane-to-lane connections
- XML-input
- Routing in dynamic networks

Traffic Lights

Traffic lights play an important role within the traffic management as they improve the traffic flow. Beside simple right of way rules each simulated junction may also be controlled by traffic lights.

Simulation Output

By now, three different outputs are available. The first is a so-called „raw“ output which contains all edges (streets) and all lanes together with vehicles driving on them for every time step. The vehicles are described by their name, position and speed. This output is complete and may be used as input to post-processing tools for further evaluation. It is easy to use because of it's XML-compliant format. However, a large simulation produces a nearly unmanageable amount of data, so other outputs must be provided.
The first compact output that may be generated is a simulation of detectors, better known as induction loops, positioned on a certain lane on a certain position. These detectors are able to compute the flow, the average velocity on the lane and other values. The results of this computation are written into a file using the CSV or the GnuPlot-format. Every detector writes to another file.
The third type of output consists of common values used in traffic research such as the flow, the average speed etc. This output is aggregated for specified time intervals which may be changed by the user on the command line or within a configuration file. The simulation can write several files of the latter type with different sampling intervals.

THE SOFTWARE COMPONENTS

Software Modules

SUMO consists of more than a single application. Some other modules exist that allow to pre-process data needed for simulations and research.
The following modules are now being developed:

Sumo

SUMO itself is the simulation program. It reads data that describe the scenario to simulate and the simulation parameters like the starting and ending times. It performs the simulation of traffic flow saving the results in different types of output files. The GUI-version displays the movement of the cars on the screen.

Sumo-netconvert

The next picture illustrates the use of a net-building tool when a whole city is to be modelled. Neither the number of the streets nor the complexity of their relationships allows the processing of the network description by a human user. For this reason, SUMO-NETCONVERT was implemented.
During the process of network generation, SUMO-NETCONVERT reads in the available data, computes the input for SUMO: the streets, their lanes, right-of-way rules, lane-based street to street connections and traffic lights for junctions. The results are written to a XML-file.
By now, three different common formats may be converted into SUMO-networks:
- simple XML-data containing edge types, nodes and edges
- ASCII-Lists of nodes and edges
- VISUM-networks

We also implemented an ArcView import that needs some additional scripts.

Figure 2: The city of Berlin built from high-quality digital map data

Sumo-router

Beside the static part – the network – the simulation consists of moving vehicles. Usually departure times and a route's origin and destination are given, but the routes themselves must be computed. To avoid their time consuming online-computation during the simulation itself, a separate module, the SUMO-ROUTER generates and saves them and they can be loaded by the simulation.
This module receives the departure times, origins and destinations for a set of virtual humans that shall be simulated and computes the routes through the network using the well-known Dijkstra routing algorithm (Dijkstra 1959). For the routing no sophisticated algorithms like hierarchical networks have been implemented, yet.
As the speed on the streets changes with the traffic load and so a previously best route may be slower than others when used by too many vehicles, the routing is

based on the Dynamic Traffic Assignment approach developed by Christian Gawron (Gawron 1998) where routing and simulation are repeated several times to achieve a real-world behaviour of drivers.
The router takes into account that the load on the streets changes during the day.

Sumo-GUI

The GUI-version of SUMO was realised by extending the original simulation package by a windowing and graphical interface. It is based on the Qt Windowing (Trolltech 2002) library and openGL (OpenGL 2002) for fast rendering of the geometry. This approach allows the installation on the most common operating systems: UNIX, Windows and Linux.

Software Development

SUMO is implemented in C++. During the development, we use only standardised parts of the language. One of them is the STL which – when not coming directly with the compiler – may be downloaded for free (STLPort 2001). The code is well documented and formatted and some of the Ellemtel-guidelines (Ellemtel 1990-1992) are followed which assure portability to most systems.
Due to this, our software compiles on most platforms and we validated this for the following environments:

- Windows using MSVC++
- Solaris using SUN-C++-compiler and STL-Port
- Linux using gcc

Extensibility

Hoping for the participation of other interested persons, we try to supply potential developers with all information needed to extend and modify the program (SUMO 2002).
Beside the possibility to add new elements such as intelligent traffic lights, models of cars equipped with ACC systems, etc. inside a fully developed environment, we hope to supply interested researchers with a common platform to test their microscopic models with.

THE MODEL VALIDATION PROJECT

The Object of Interest

Our object of interest are two parts of American highways, one located at the I-80 highway and the other on the I-880. We will now show how to build the network, run the simulation and some of our results, all for the I-880 as the steps are the same for both scenarios.

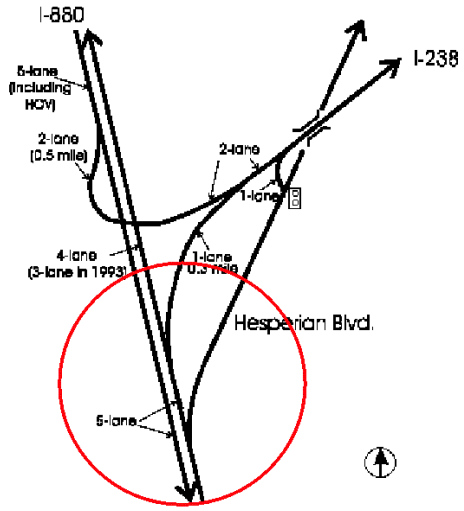

Figure 3: Picture of the I-880.

Figure 3 shows a picture of the I-880. We simulate the parts within the circle. Notice that we only consider the direction to the north.
This part of the highway is especially worth to look at for two reasons. First, there are frequent breakdowns, second, it is equipped with many detectors. All of them are double induction loops which allow to measure the mean speed of passing vehicles. The positions of the detectors may be seen on figure 4. The circles in this figure are the junctions („nodes“) every road segment starts and/or ends at.

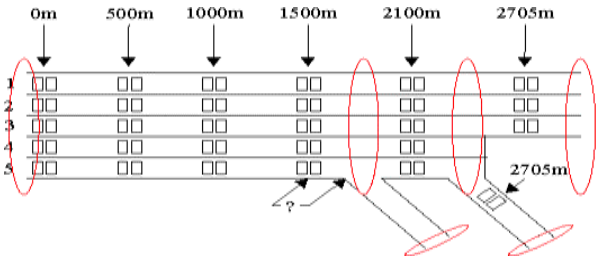

Figure 4: Schematic view of the detectors

Network Generation

The network was modelled by hand as the number of the street segments is small.
The first thing to do was to generate a list of nodes where each of the edges starts or ends at. The resulting XML-file is very compact – it is only 10 lines long. The same was done for the edges where an edge is specified by it's source and it's destination node. The resulting files contains 10 lines, too.

Both files were converted into a fully connected SUMO-network using the SUMO-NETCONVERT – tool. The interconnections between lanes causing a certain behaviour of the traffic flow are shown on the left side of figure 5. They are automatically generated by the SUMO-NETCONVERT module. The right side of the figure shows the network as it appears in the main window of the GUI-version.

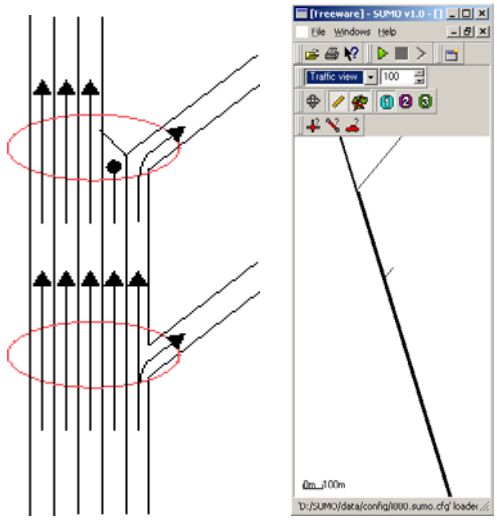

Figure 5: The lane connections (left) and the network as displayed by the GUI

Further Input

Detectors

Trying to validate our model's behaviour, we used measured data and mapped it to our own, simulated detectors. For this, we built a file which describes where to set the detectors onto the network and according to the positions shown in figure 4. The detectors are named „i880_dXlY" where X is the number of the detector on a lane (from left to right on the first picture) and Y is the number of the lane (from rightmost lane to the leftmost lane or from bottom to top on the first picture). There are no detectors with the number 1 as the first detector column – the column at 0m – is used for entering vehicles into the simulation.
There is another detector on the eastbound off-ramp, called „eastb_d6".
Every detector runs at 1Hz and is generating a file that has the detector's name.

Sources

Sources are used to let vehicles enter the simulation. Within SUMO three types of sources are implemented:

- periodic sources
  Vehicles start every n seconds.
- triggered sources
  Vehicles start at certain time steps which are read from additional files.
- stochastic sources
  The vehicles start with a certain probability.

Each vehicle is characterised by a number of parameters: initial speed, type and route to take.
The real-world column of the real-world induct loops is implemented within our simulation as triggered sources what allows to put in vehicles according to the real-world data. More precisely, the simulation generates a vehicle at one of the sources being positioned at 0m every time, a vehicle has passed the corresponding detector in the real situation.

The Car-Driver-Model

The model used currently within SUMO is the Gipps-model extension developed and described by Stefan Krauß (Krauß 1998, Janz 1998). The model reproduces the main features of traffic, namely free and congested flow.
At each time step the vehicle's speed is adapted to the speed of the leading vehicle in a way that results in a collision-free system behaviour within the following simulation step(s).

Calibration and Validation

As our car-driver model contains some variable parameters, it may be calibrated for the simulation at hand. For this purpose, a small programme was built which prepares the vehicle types and parameter definitions used within the simulation and writes them into a file that is read by the simulation programme. In addition to this file, a meta-file containing the description of the calibration process is built by this tool.
The project settings are similar to the settings used within the original car-model validation project and in fact, even the used approximation programme was the same with minor changes needed to generate the needed input. See (Brockfeld , Kühne, Skabardonis, Wagner 2002) for more details.
After each simulation, this program is restarted and continues with the calibration using the previously saved information about the process as shown by the fat arrows in figure 6.
The calibration uses the comparison of the simulated travel times saved on the hard disc by our simulated detectors to the detector data available from the real-world I-880. The error function to minimise is:

$$e(p) = \frac{\left\langle \left| T_{sim}(p) - T_{obs} \right| \right\rangle}{\left\langle T_{obs} \right\rangle} \quad , \qquad (1)$$

with:
T_{sim}: simulated travel time
T_{obs}: observed travel time

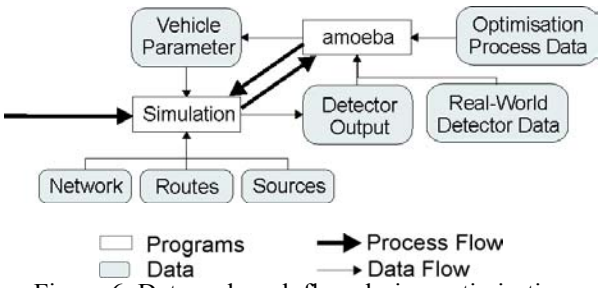

Figure 6: Data and work flow during optimisation

As approximation algorithm, the amoeba-algorithm from Numerical Recipes was used.
This approach also needed the implementation of a triggered traffic light which turns red when a too large amount of cars passes it.

RESULTS

Calibration and Simulation Results

During the calibration phase, the model's quality improved from an error around 40% to an error around 15%.

CONCLUSIONS

Simulation Benchmarks

The simulation is capable to simulate around 1 million of vehicle movements per second depending on the network complexity. Further optimisations will follow.

Project Results

We showed that SUMO is capable to be used for microscopic traffic model calibration with small effort. The need for open interfaces, both on the side of model parameterisation as well as on the output side seems evident.
As we are currently working on the simulation itself, we did not have the time to implement other models like it was done within the original car-model validation project and hope to find co-workers who want to embed their models into a complete street environment that includes all needed structures to model real-world traffic. Until now, no such extendable simulations were available.

FUTURE WORK

Beside implementing and testing other car/driver models, the calibration tool will be extended and supplied as a part of the project. For this, new optimisation approaches and an easier handling should be implemented.
Some projects have shown the need to integrate the software into other software packages or to build interfaces between the software and other programming languages. Within this context, the simulation may be wrapped to be usable from other programming languages such as Perl or Python as a library.
The high complexity of the simulation should also allow to calibrate models of traffic control systems such as traffic lights.
The amount of traffic to simulate is growing and also some research experiments need several simulations to compute a single value. The second, for instance, is the case in traffic light optimisation where several timing schemes must be tested. So, an extension of the system that allows the usage of computer cluster should be kept in mind.
We also hope to gain help from other persons or institutes who are interested in traffic simulation and want to participate on SUMO's development by extending, improving or just using it.
We invite you to visit the SUMO-pages at http://sumo.sourceforge.net.

BIBLIOGRAPHY

E. Brockfeld et al. 2001. „Optimizing Traffic Lights in a Cellular Automaton Model for City Traffic". In: Physical Review E 64, 056132

E. Brockfeld , R. Kühne, A. Skabardonis, P. Wagner. 2002. „Towards a Benchmarking of microscopic Traffic Flow Models". TRB, in submission

E. W. Dijkstra. 1959. „A note on two problems in connection with graphs". In: Numerische Mathematik, 1:269-271.

Ellemtel. 1990-1992. Coding Standards Page. http://membres.lycos.fr/pierret/cpp2.htm

Christian Gawron. 1998. „Simulation-Based Traffic Assignment". Inaugural Dissertation.

Stefan Janz. 1998. „Mikroskopische Minimalmodelle des Straßenverkehrs". Diploma Thesis.

D. Krajzewicz, G. Hertkorn, C. Rössel, P. Wagner. 2002. „SUMO (Simulation of Urban MObility) - An open-source traffic simulation". In: MESM2002 Proceedings. Coming out Okt. 2002

Stefan Krauß. 1998. „Microscopic Modelling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics". Hauptabteilung Mobilität und Systemtechnik des DLR Köln. ISSN 1434-8454

K. Nagel, M. Schreckenberg. 1992. Journal of Physics I 2, 2221

OpenGL.org. 2002. OpenGL.org Homepage. http://www.opengl.org/

STLPort. 2001. Company Homepage. http://www.stlport.org/index.html

SUMO: G. Hertkorn, D. Krajzewicz, C. Rössel. 2002. SUMO Homepage. http://sumo.sourceforge.net

TrollTech. 2002. Troll Tech Homepage. http://www.trolltech.com

Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling. 1988. "Numerical Recipes in C: The Art of Scientific Computing," William H. Press, Cambridge University Press, New York, ISBN 0-521-35465-X