1st SUMO User Conference 2013

Proceedings

Berichte aus dem DLR-Institut
für Verkehrssystemtechnik

Band 21

DLR Deutsches Zentrum
für Luft- und Raumfahrt

Reports of the DLR-Institute of Transportation Systems

Volume 21

Proceedings of the 1st SUMO User Conference SUMO2013

May 15 – 17, 2013
DLR, Berlin - Adlershof

Publisher:
Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Verkehrssystemtechnik
Rutherfordstraße 2, 12489 Berlin-Adlershof

ISSN 1866-721X

DLR-TS 1.21

Berlin, May 2013

Institute Director:
Prof. Dr.-Ing. Karsten Lemmer

Preface

Dear reader,

You are holding in your hands a volume of the series „Reports of the DLR-Institute of Transportation Systems". We are publishing in this series fascinating, scientific topics from the Institute of Transportation Systems of the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V. - DLR) and from his environment. We are providing libraries with a part of the circulation. Outstanding scientific contributions and dissertations are here published as well as projects reports and proceedings of conferences in our house with different contributors from science, economy and politics.

With this series we are pursuing the objective to enable a broad access to scientific works and results. We are using the series as well as to promote practically young researchers by the publication of the dissertation of our staff and external doctoral candidates, too. Publications are important milestones on the academic career path. With the series „Reports of the DLR-Institute of Transportation Systems / Berichte aus dem DLR-Institut für Verkehrssystem-technik"we are widening the spectrum of possible publications with a bulding block. Beyond that we understand the communication of our scientific fields of research as a contribution to the national and international research landscape in the fiels of automotive, railway systems and traffic management.

This volume contains the proceedings of the 1st SUMO User Conference (SUMO2013), which was held from 15th to 17th May 2013 in Berlin-Adlershof, Germany. SUMO is a well established microscopic traffic simulation suite which has been available since 2001 and provides a wide range of traffic planning and simulation tools. The conference proceedings give a good overview of the applicability and usefulness of simulation tools like SUMO ranging from new methods in traffic control and vehicular communication to the simulation of complete cities. Another aspect of the tool suite, its universal extensibility due to the availability of source code, is reflected in contributions covering parallelization and workflow improvements to govern microscopic traffic simulation results.

Several articles give short outlines of the general workflow when setting up a simulation with SUMO as well as an overview about the available tools for net and demand generation and for the evaluation of the results. Further features, include the simulation of private and public transport modes, person-based trip chains as well as the extension for the implementation of new behavioral models or remote control of the simulation using various programming environments. The conference's aim was bringing together the large international user community and exchanging experience in using SUMO, while presenting results or solutions obtained using the software. This collection should inspire you to try your next project with the SUMO suite as well or to find new applications in your existing environment.

Prof. Dr.-Ing. Karsten Lemmer

All contributions have been double refereed as abstract and full paper by the International Scientific Committee

Table of Contents

1 Summary on Publications citing SUMO, 2002-2012

Daniel Krajzewicz;
German Aerospace Center, Germany

1.1 Abstract

Having a first release after the success of internet search engines, and being a scientific tool which usage results are usually published, both the development as well as the usage of the open source traffic simulation package "SUMO" can be followed and exploited using commonly available tools to a high degree. In the following, an evaluation of publications which cite or name SUMO is given. Different aspects are shown, such as the development of the publications number over years or the publications' topics.

Keywords: Document Processing, open source Development, Bibliometrics, Traffic Science.

1.2 Introduction

While co-authoring an open source scientific software like the traffic simulation suite "SUMO" [1] [2], one may get interested in its usage, usability, and acceptance in the field it is designed for. Direct interaction with users is one possibility, but it often covers discussions on current issues only and the final results of the users' work are rarely disseminated using this channel. Several circumstances simplify, not to say allow, to collect information necessary for answering questions on a software's usage and acceptance, nowadays. First to name is of course the evolvement of internet search engines, in combination with the fact that SUMO was made available for the public after the search engines' omnipresence. The second is that SUMO mainly targets a scientific audience who presents obtained results in publications of different kind, usually – as a good practice – citing the used tools. Such publications are often indexed by search engines and can be found using appropriate search terms. Additionally, SUMO is a specific tool, reducing the audience and, by doing this, limiting the number of references to a manageable size.

Along the development years, the number of found publications naming SUMO – named "the collection" in the following – grew to a size which is assumed to be high enough to exploit them in a statistical way, not by discussing every paper for itself, but by looking at the tendencies in naming and using SUMO. The collection's basic bibliographic information already allows to examine the development of the number of publications or their types. Additionally, the collection's documents were classified manually by the addressed research topic, the organisations the authors belong to, and the role SUMO plays within the publication, allowing further evaluations. As most documents are available as .pdf-files, it was also partially possible to retrieve the text and to evaluate it.

This document concentrates mainly on listing the generated numbers, distributions, or classes that summarize the collection itself or the work described in the documents the collection consists of. To some degree, it is similar in both, the scientific context of work and in the used technique, to the evaluation of "IVC[1] Simulation Studies" performed by Stefan Joerer, Christoph Sommer and Falko Dressler [3].

It should also be stated that the work presented here is not intended to bring any further insight into topics such as document processing, traffic science, bibliometrics, or science theory, even these topics are touched and/or their tools are partially used. Trying to gain some knowledge about citation theories, the author stumbled across [4], what he thinks is worth to be read.

This document is structured as follows: at first, the collection is described, including the process of retrieving the documents it consists of, information how the documents were classified and information about the statistical relevance and the data quality. Then, different views on the documents are given, mainly split by the regarded information. This document ends with acknowledgements and a summary/outlook.

Figure 1-1: Frequency of words within the collection's titles, generated with Wordle™ [25]

1.3 The Collection

This section introduces the documents collection, describing how it was generated, first. The manual classification performed on the collection is presented afterwards. The section closes with a discussion about the statistical relevance and the assumed quality of the collection.

1.3.1 Collecting the Documents

A large part of the documents was collected by searching the web using Google [5] and other web search engines. The search was performed between the years 2002 and 2012 repeatedly, albeit spontaneously, not following any fixed frequency or any certain event. Usually, the search was performed using combinations of the terms "SUMO", "traffic simulation", "Simulation of Urban Mobility", and "Krajzewicz"[2], as neither "SUMO", "traffic simulation", nor "Simulation of Urban Mobility" yields in sets containing references to the object of research only. Starting in 2011, additional Google Scholar "alerts" [6] have been used. When established, a Google Scholar alert sends mails to the person who has set it up as soon as Google Scholar receives a new document containing certain words. Two Google

[1] IVC: inter-vehicle communication

[2] The author's name was used not only for some obvious personal reasons, but also because the author is the (co-)author of most publications about SUMO; additionally, the name is occurring relatively seldom, making it a good discriminator.

Scholar alerts were set up, one tracking the search term "'traffic simulation' SUMO", and one tracking "Krajzewicz".

Though SUMO users were asked to send information about their results and/or publications via the mailing list and this request is also included at SUMO's home page, only few citations were reported, which are also enclosed in the list. SUMO was also described by its authors. Publications authored by SUMO's core development team members[3] only are not included in the collection, as they do not resemble SUMO's visibility and usage in the user community. Publications, which the author has contributed to were not added to the collection explicitly. Nonetheless, the database contains five documents co-authored by him.

The links sent by Google Scholar often point to digital library portals, such as Springer Link [7], IEEE Xplore [8], ScienceDirect [9], or ACM Digital Library [10]. Usually, these portals do not only allow to download the publication itself, but also its bibliographic information. In such cases, the bibliographic information was stored as a BibTeX-entry [11] [12] into a BibTeX-database. The entry was afterwards extended by a reference to the downloaded full-text file. In some cases, the Google Scholar link points to the publication itself. In such cases, and for the documents collected before 2011, which were available as .pdf-files, the title of the publication was extracted and used as search term in Google Scholar [13]. From the results, the entry with the same title and the same authors as the original document was chosen as the one representing the document. No ambiguities were observed during this preparation step and the result was a set of BibTeX-databases containing the documents' bibliographic references.

From the database, eleven documents were removed, as it was not possible to obtain their full-texts, disallowing their further processing. Ten of those documents were published in the year 2012, the remaining one in the year 2011.

1.3.2 Manual Classification

Three classification schemes were applied to each of the documents in the collection. This work was done using JabRef [14], an open source literature management application working natively with BibTeX-files. The following classification schemes were applied:

1) SUMO's role in the reported research

The first classification scheme resembles the role of SUMO within the publication. Here, the following classes were used: "mentioned", "used", "extended", and "contributed", where "contributed" is a sub-set of "extended". The classification is assumed to be valid to a high degree, even despite the fact that it was not always possible to determine whether SUMO was extended or not. The classes were defined before classifying the documents.

2) The topic of the publication

The second classification scheme tries to sort the publications by their main research topic. It should indeed be named as a "try", as it is the semantically most weak of the used classification schemes. Imagine a document which reviews mobility models for VANET simulation (such as [15]). During the classification, the document was assigned to the subtopic "mobility models" of the major topic "V2X". On the contrary, a publication

[3] explicitly: Michael Behrisch, Laura Bieker, Jakob Erdmann, Marek Heinrich, Melanie Knocke, Daniel Krajzewicz, Yun-Pang Flötteröd, Peter Wagner

targeting on traffic simulations ([16], e.g.) was assigned to the subtopic "traffic simulations" of the major topic "mobility models". The main motivation behind this scheme was to quickly recognize in which scientific area SUMO is used or mentioned.

In several cases, a document was assigned to more than one topic. An example may be a thesis, where a simulation system is presented, which is used in subsequent steps to evaluate a V2X[4] message routing protocol as well as applications based on the communication. Such a document would be assigned the subtopics "simulation software", "routing protocols", and "applications" of the major topic "V2X". Although a coarse overview on the topics SUMO is used for was known before the classification, the topics were not defined before the document's classification. Instead, new topics or sub-topics were added during the classification process, if needed. After classifying all of the collection's documents, the topics were revisited for balancing them, mainly by joining less occupied topics into classes named "other" at different depths of the topics tree. The chosen topics, sub-topics as well as the terms used to describe them are surely dictated by the author's experience.

3) Organisation(s) the author(s) belong to

Finally, the organisation or organisations the authors of a publication belong to was determined, which was almost always possible using the information contained in the document itself. The classification is hierarchic: on the first level, the country an organisation is located in, was used. On the second level, the organisation – mostly an university – was given. On the third level, if available, the department and/or institute was used. For five publications, it was not possible to completely assign the authors to institutes. These publications were assigned to the group "unknown". Three reports, all from projects co-founded by the European Commission are not indexed, mainly because resolving the partner organisations' abbreviations given in the document was assumed to be too time consuming.

1.3.3 Statistical Relevance

One should pose the questions whether the collection contains all publications citing SUMO or represents the set of all publications citing SUMO properly. The first question should be negated, starting with the fact that the core developer's publications are not included. But, e.g., Master theses are not always made publicly available and so cannot be found using web search engines. This is assumed to count for other types of publications as well.

The second question can be hardly answered without knowing all publications. Nonetheless, based on following SUMO's usage via both, users' publications, as well as the interaction on the mailing list, it is assumed that it is a fair overview of how SUMO is used. In addition, the documents collecting steps were performed using unbiased search terms, concentrating on finding information about SUMO only.

Of a rather academic nature is the question whether all groups that use SUMO, cite it. This is probably not the case, starting with the fact that SUMO is often used as a part of multi-simulator network architectures, such as "Veins" [17], "MOVE" [18], "VSimRTI" [19], or "iTETRIS" [20].

[4] The term "V2X" is a simplifying abbreviation of "vehicle-to-vehicle and vehicle-to-infrastructure"

1.3.4 Data Quality

The BibTeX-descriptions obtained from library portals are of good quality. The bibliographic information from Google Scholar was often erroneous. Both, Master and Doctoral theses were very often not assigned to the proper BibTeX-type. This was corrected manually. Often, the publication year was not given or one could find the name of a month in the according field, instead. This was corrected manually, but the information could not be found for six documents.

Another predictable issue were the author names. Dots were missing at the first name abbreviations, non-Latin characters as German Umlaute (äöü) were encoded in different ways, first names were only sometimes abbreviated, and middle names not always given. This was corrected manually by inspection of similar names and/or from knowing the listed authors. It is assumed to be not reliable to 100%.

The collection includes 362 documents. Table 1-1 summarizes the known quality issues.

Table 1-1: Summary on the collection's quality issues.

Property	Given	Estimated Correctness
document number	362	
BibTeX: type	362	Is mandatory in BibTeX, but is probably not correct for many of the documents as discussed.
BibTeX: Publication Year	356	Assumed to be correct
BibTeX: Journal / BibTeX: Booktitle		Assumed to be incorrect and complicated and time consuming to be corrected; neglected
BibTeX: Authors	362	Assumed to be correct to a large degree
Classification: Role	362	Assumed to be correct to a large degree
Classification: Topic	362	Assumed to be correct
Classification: Institutions	359(-5)	Assumed to be correct to a large degree, despite the limitation described in section 1.3.2.

1.4 Evaluation

In the following, the collection's properties are given. The evaluation was done using the Python [21] programming language and the matplotlib [22] module for visualisation.

1.4.1 Titles

A very coarse, statistical view at the titles, shown in Figure 1-1, already implies what will be discussed in section 3.5 about research topics of the collection: the dominance of research on vehicular communications. Figure 1-1 was generated by collecting the words from titles, for which the stem was determined using Porter stemming [23] [24]. As stems obtained from the Porter algorithm are usually pruned so much that no real English word is obtained, a mapping from the stem to the shortest complete word that was found within the titles was used to get existing English words. The resulting list of occurrences per word was then visualised using Wordle™ [25]. For stemming, the Python Porter stemming implementation from Vivake Gupta and Danny Yoo was used [26].

1.4.2 Development over Time

Figure 1-2 shows the development of the annual publications number over time, distinguishing the publications' types. It shows a fair increase over the years, albeit with some dents.

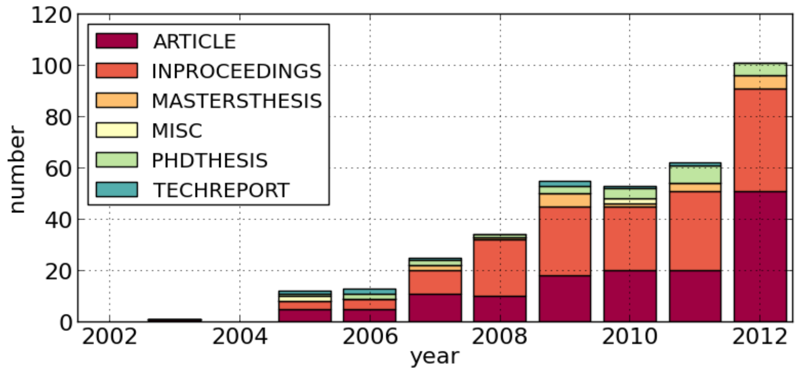


Figure 1-2: The development of publications over the years, divided by publication type

1.4.3 Authors and Authorship

The publications are authored by 863 persons in sum; the development is shown in Figure 1-3. "all" denotes the set of all authors that have co-authored a document in a given year, "new" is the subset of "all" containing authors who have not (co-)authored an earlier publication from the collection and "single" is a subset of "new", which contains authors who have contributed to one publication within the collection only.

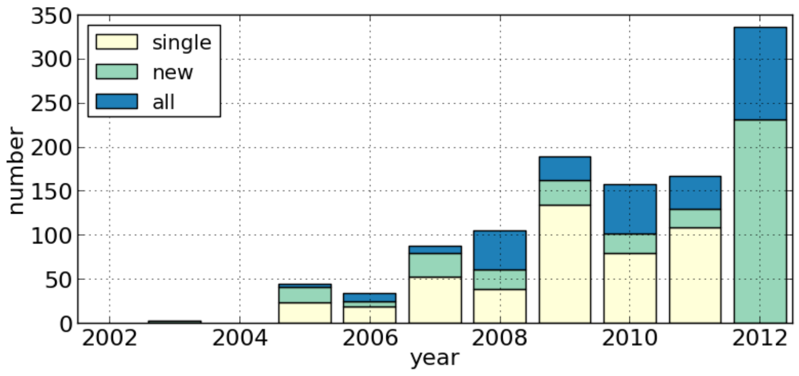


Figure 1-3: The development of authors over the years, classified by the continuity of their publications from the collection

Albeit the number of "all" authors is growing, the majority of the authors has participated in one publication only. This could be interpreted as a minor acceptance, but is rather supposed to have its reasons in a high number of co-authors. Often, a document describes a larger project and only few of the documents' authors have used SUMO by themselves.

1.4.4 SUMO's Role

The manual classification of documents by the role of SUMO within the reported research is maybe the best indicator for SUMO's acceptance in the scientific community. As shown in Figure 1-4, not only the number of documents which report about using SUMO is increasing over time, but also their percentage, in comparison to documents which mention SUMO only.

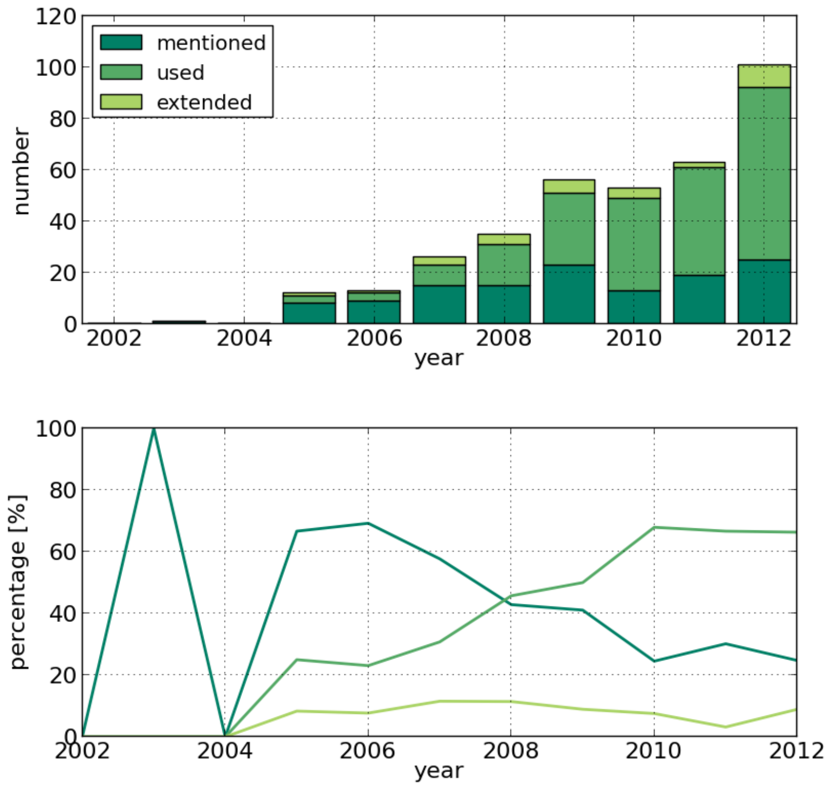


Figure 1-4: Role of SUMO within the evaluated publications. Top: absolute numbers, bottom: percentage, both along the years

Also remarkable is the almost constant percentage of documents where extensions to SUMO are reported. It should be noted, that after introducing the TraCI application control interface in 2008, which allows an on-line interaction with the simulation, the need to extend SUMO was reduced as the logic to evaluate can be embedded into an external application since that time. Although one could assume a reduction of extensions to SUMO since that time, this is not visible within the collection.

1.4.5 Research Topics

As described in section 1.3.2, a document may be assigned to more than one research topics. For the discussion, it may be interesting to know which and how many topic combinations exist within the collection. Figure 1-5 shows the co-occurrence of topics. Topics which do occur in combination with other topics are not shown, herein.

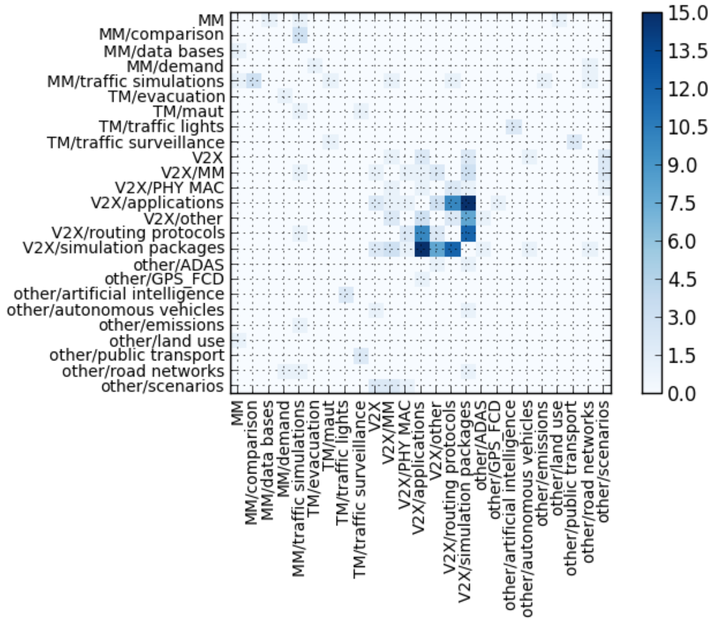


Figure 1-5: Co-occurrences of topics within the collection. Topics with no co-occurrence are not shown. The abbreviations are: MM: mobility models, TM: traffic management

The development of addressed topics within the collection over time is shown in Figure 1-6. Clearly evident is the domination of documents which describe work on “V2X”. In sum, about 70 % of the collection were classified into this topic, about 12 % present work on “mobility models” and each of both topics “traffic management” and “other” is discussed in about 9 % of the publications.

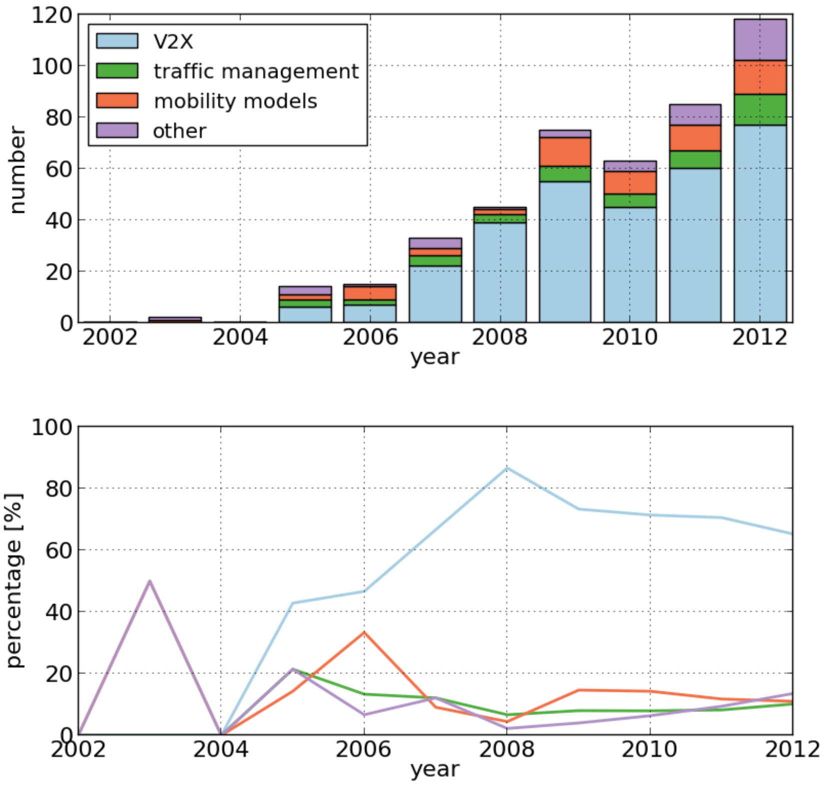


Figure 1-6: Development of the publications’ major topics. Top: absolute numbers, bottom: percentage, both along the years

The dominance of V2X research within the SUMO user community was already observed and reported, see [1]. The evaluation of the topics proves it. It may be also noted, that, vice-versa, SUMO is the most often used traffic simulation tool in V2X research, as reported in [3]. The reasons can only be guessed. The first documents targeting this topic occur in the year 2005. One of those, “MOVE: A MObility model generator for VEhicular network” by Feliz Kristianto Karnadi, Zhi Hai Mo, and Kun-Chan Lan [28], may be the reason for SUMO’s prominence in this research field. [28] presents “MOVE”, a complete system for generating vehicular traces that can be used in the ns2 simulator, which was the state-of-the-art for communication simulation at that time. Screenshots of different scenarios show the system’s variability in use.

Whether [28] was the seed to SUMO’s popularity within the research on V2X or not, may be provable by evaluating the citations of following papers, trying to determine how often [28] is cited. This was not done so far.

As “V2X” covers about 70 % of the publications, it may be investigated using the same approach. Figure 1-7 shows the development of sub-topics of “V2X” along the years. The overall frequencies of “V2X” sub-topics are as following: “simulation packages”: 30.9 %, “routing protocols”: 24.7 %, “applications”: 22.7 %, “PHY/MAC” and “mobility models”: 5.2 %, and “other” cover 11.3 %.

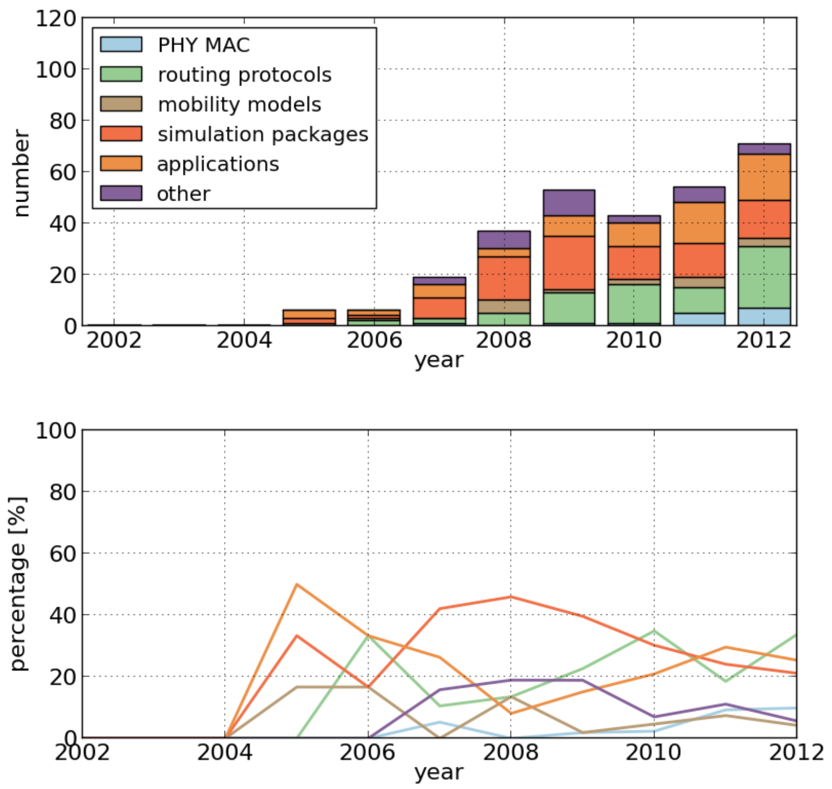


Figure 1-7: Development of the publications’ sub-topic within the “V2X” topic. Top: absolute numbers, bottom: percentage, both along the years

It is interesting to note that “simulation packages” – presentation of tools for research – is dominant to such a high degree. On the other hand, one should take into regard that the presentation of a “simulation package” occurs often in combination with some further evaluation of V2X functionality, may it be a “routing protocol”, or an “application”, see also Figure 1-5. Nonetheless, the author finds the number of reports on “simulation packages” quite high. Whether such “simulation packages” find their way into a broader use, or are only used once, and whether the necessity to develop new ones exist, given the high number of existing ones, is matter of a different kind of research.

A promising fact is the increase in using SUMO for evaluating V2X-based applications, as in most cases, such research requires a joint operation of a traffic simulation and a communication simulation, usually employing a middleware instance. The increase of publications on this topic shows that available middleware solutions are accepted and can be used for scientific work. Within the work on "routing protocols", SUMO is usually used to generate vehicular "traces" only, which are then exported into a format readable by the used communication simulator.

The document sets of the remaining major topics are too small for a meaningful insight into the development over time. Figure 1-8 shows the distribution of the sub-topics within the major topics "mobility models" and "traffic management".

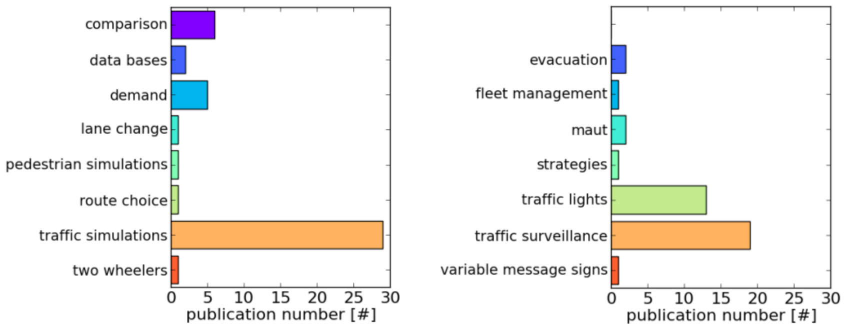


Figure 1-8: Subtopics and their relative frequency for the topics "mobility models" (left) and "traffic management" (right)

1.4.6 Countries and Organisations

The categorization into the institutions the authors belong to is shown on the top-most, national level only, herein. Figure 1-9 shows a matrix of international co-authoring of the collection's papers, whereas Figure 1-10 shows the numbers of publications per country.

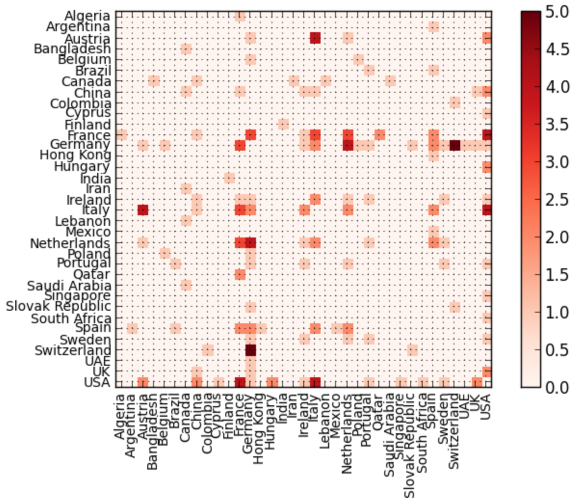


Figure 1-9: Co-occurrence of countries within the collection

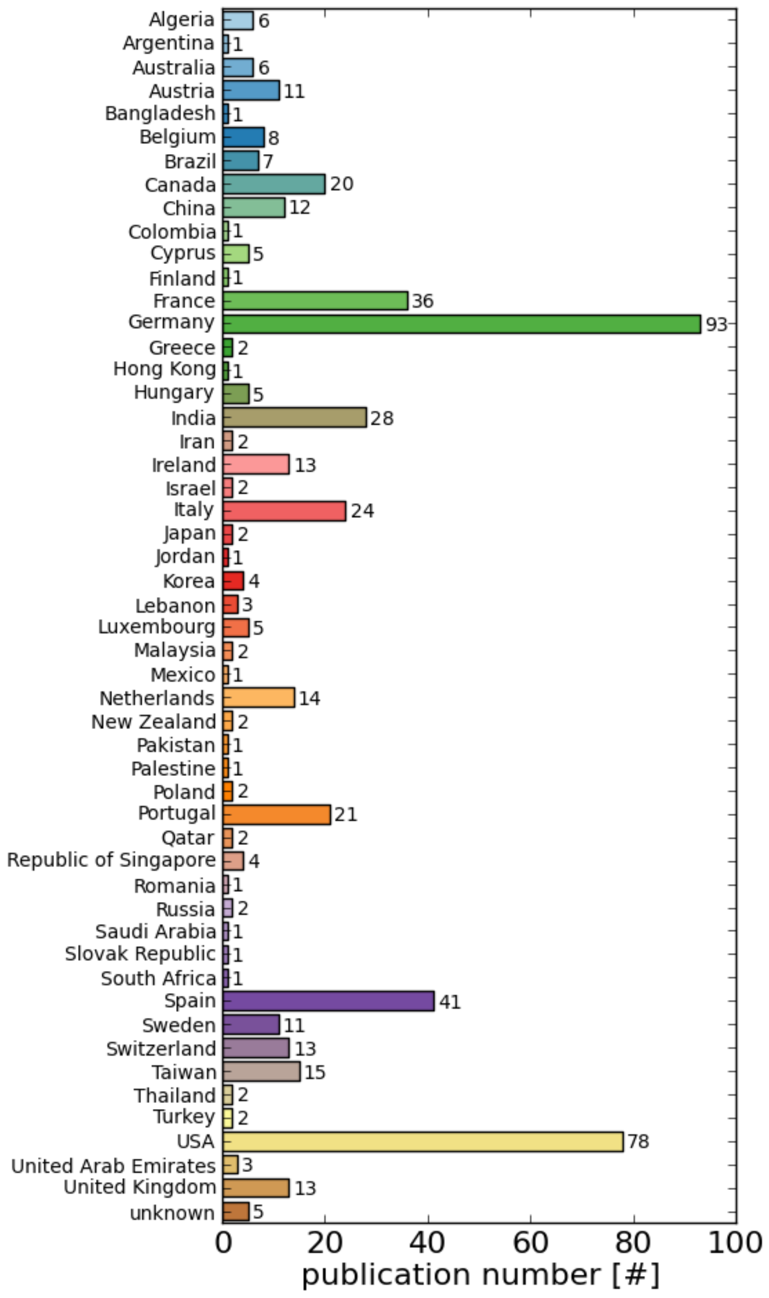


Figure 1-10: The number of publications per country

1.4.7 Content Statistics

Parallel to classification of the documents, the reference to the pdf-document was added to each entry. In a later step, "pdfminer" [29], an open source Python library for pdf parsing, was used to retrieve an xml-representation of the document as well as the text contained in it. At the current time, only some coarse values were extracted, including the number of pages, the number of word, and the number of characters. Figure 1-11 shows these values – the colouring is based on publication type as used in Figure 1-2. More in-depth evaluation of the contents may be the topic of later work.

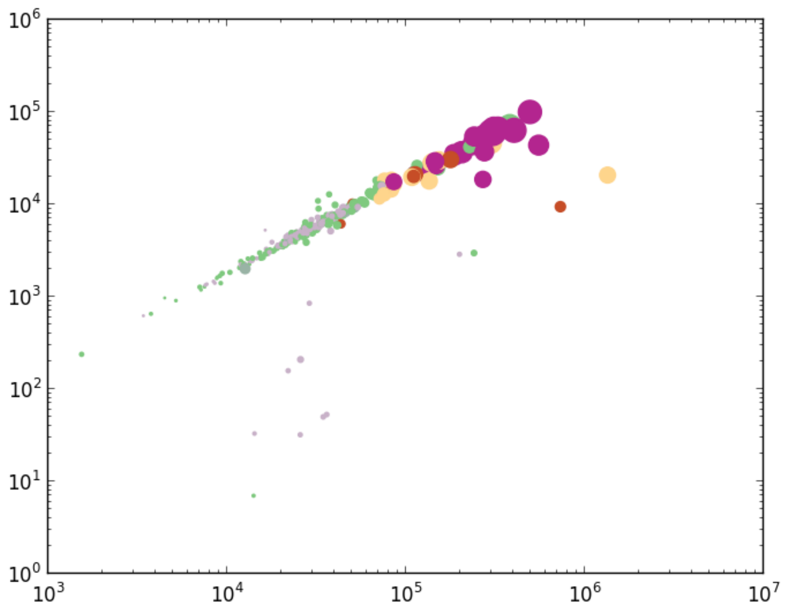


Figure 1-11: Word number (y) over character number (x) with log-scales for x- and y-axes; colour: BibTeX-type as in previous figures; size: page number

1.5 Summary and Outlook

Different views on a collection of documents which name the open source traffic simulation SUMO were presented. The document's bibliographic references, a manual classification performed on them and text processing were used to obtain the presented results. The evaluations show the progress of using SUMO, the major topics it is applied for, as well as other aspects.

Summarizing, it is possible to state that the number of publications increases and that no hints for a change in this development are visible. SUMO itself is accepted as a tool useful for research, indicated by the growing number of work that report its usage. The researches which cite or use SUMO come from all over the world, albeit European countries and the USA dominate. Research on vehicular communication is the major application topic to be found within the papers and, even if extrapolating the numbers, one should assume that it states at this position for the next time.

The shown distributions and developments over time are a coarse look at the data set only. It is assumed that the data set allows further evaluations, but most of such would be based on per-author evaluations what was not wanted for the presentation performed here.

As one could assume, the collection itself is of a high value, pointing to already done, classified work, or to parties working on certain topics. Nonetheless, the work on the collection – collecting, scanning, reading, and classifying the documents – gave probably more insight and surprises than a view on the complete collection presented here.

1.6 Acknowledgements

The author wants to thank all authors of the evaluated documents, and especially those who informed the SUMO development team about their work. Further thanks go to Alexandra Eßl for her work on the collection.

The work would be not possible without free search engines and library portals mentioned earlier. Also, acknowledgements go to the developers of the open source tools used in this research, namely JabRef, Python, matplotlib, and pdfminer.

1.7 References

[1] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent *Development and Applications of SUMO - Simulation of Urban MObility.* In: International Journal On Advances in Systems and Measurements, 5 (3&4), pp.128-138. ISSN 1942-261x (2012)

[2] DLR and contributors: SUMO homepage. http://sumo.sourceforge.net/ (2013)

[3] Joerer, S., Sommer, C., Dressler, F.: *Towards Reproducibility and Comparability of IVC Simulation Studies--A Literature Survey.* In: Communications Magazine, IEEE, 50 (10), pp.82-88. ISSN 0163-6804 (2012)

[4] Smith, L. C.: Citation analysis. Library trends, 30. Jg., Nr. 1, pp. 83-106 (1981)

[5] Google. Google web search engine. http://www.google.com; last visited on 08.04.2013.

[6] Google. Google Scholar alerts. http://googlesystem.blogspot.de/2010/05/email-alerts-for-google-scholar.html, last visited on 08.04.2013.

[7] Springer. Springer Link. http://link.springer.com, last visited on 08.04.2013.

[8] IEEE. IEEE Xplore. http://ieeexplore.ieee.org/, last visited on 08.04.2013.

[9] ScienceDirect. ScienceDirect. http://www.sciencedirect.com/, last visited on 08.04.2013.

[10] ACM. ACM Digital Library. http://dl.acm.org/, last visited on 08.04.2013.

[11] BibTeX.org. http://www.bibtex.org/, last visited on 08.04.2013.

[12] Wikipedia. BibTeX description. http://en.wikipedia.org/wiki/BibTeX, last visited on 08.04.2013.

[13] Google. Google Scholar. http://scholar.google.com/, last visited on 08.04.2013.

[14] JabRef development team. JabRef web site. http://jabref.sourceforge.net/; last visited on 08.04.2013

[15] Härri, J., Fiore, M., Filali, F., Bonnet, C., Chiasserini, C., Casetti, C.: *A realistic mobility simulator for vehicular ad hoc networks.* Research report RR-05-150, EURECOM (2005)

[16] Kotushevski, G., Hawick, K.: *A review of traffic simulation software.* In: Conf. Information and Mathematical Sciences, New Zealand (2009)

[17] Sommer, C., Dressler, F.: *Progressing toward realistic mobility models in VANET simulations.* In: Communications Magazine, IEEE, 46, pp. 132-137 (2008)

[18] Karnadi, F., Mo, Z., Lan, K.-c.: *Rapid generation of realistic mobility models for VANET.* In: Wireless Communications and Networking Conference, pp. 2506-2511 (2007)

[19] Queck, T., Schünemann, B., Radusch, I., Meinel, C.: *Realistic simulation of v2x communication scenarios.* In: Proceedings of Asia-Pacific Services Computing Conference, 2008. APSCC 08. IEEE, 1623-1627 (2008)

[20] Rondinone, M., Maneros, J., Krajzewicz, D., Bauza, R., Cataldi, P., Hrizi, F., Gozalvez, J., Kumar, V., Röckl, M., Lin, L., Lazaro, O., Leguay, J., Härri, J., Vaz, S., Lopez, Y., Sepulcre, M., Wetterwald, M., Blokpoel, R., Cartolano, F.: *ITETRIS: a modular simulation platform for the large scale evaluation of cooperative* ITS applications. In: Simulation Modelling Practice and Theory (2013)

[21] Python Software Foundation: Python Programming Language – Official Website. http://www.python.org/, last visited on 08.04.2013.

[22] Hunter, J., Dale, D., Firing, E., Droettboom, M. and the matplotlib development team: matplotlib web site. http://matplotlib.org/, last visited on 08.04.2013.

[23] Wikipedia: "Stemming". http://en.wikipedia.org/wiki/Porter_stemmer, last visited on 08.04.2013.

[24] Porter, M.F.: An algorithm for suffix stripping. In: Program, 14(3), S. 130-137 (1980)

[25] Feinberg, J.: WordleTM. http://www.wordle.net/, last visited on 08.04.2013.

[26] Gupta, V., Yoo, D.: Python Porter stemming implementation. https://hkn.eecs.berkeley.edu/~dyoo/python/py_lovins/, last visited on 08.04.2013.

[27] Cottingham, D. N., Davies, J. J., Beresford, A. R.: Congestion-Aware Vehicular Traffic Routing Using WiFi Hotspots, In: Proc. Communications Innovation Institute Workshop (2005)

[28] Karnadi, F.; Mo, Z., Lan, K.-c. MOVE: A MObility model generator for VEhicular network (2005)

[29] Shinyama, Y.: pdfminer web site. http://www.unixuser.org/~euske/python/pdfminer/, last visited on 08.04.2013.

2 Preparing Data for Urban Traffic Simulation using SUMO

José Capela Dias, Pedro Henriques Abreu, Daniel Castro Silva, Gustavo Fernades, Penousal Machado and António Leitão;
Department of Informatics Engineering, University of Coimbra / CISUC (Centre for Informatics and Systems, University of Coimbra), Portugal

2.1 Abstract

The ever increasing world population, added to the always present need of human transportation, makes road traffic in the big cities a huge and urgent issue, and, consequently, the target of substantial investigation. This paper introduces the first task of a larger project called COSMO. Using the city of Coimbra as the base, the goal of this project is to simulate the city traffic using a microtraffic simulator. For that, the SUMO platform was used and different experiments were conducted varying the number of cars being simulated, as well as other variables. At the end, the results show that the selected simulator was capable of simulating the Coimbra urban traffic in a realistic fashion, using an origin-destination matrix based on real data and routing algorithms. These results show that the simulator can be used in future works related to urban traffic, namely in road network management.

Keywords: Traffic Simulation, Origin-Destination Matrix, Intelligent Transport Systems

2.2 Introduction

The inhabitants of urban areas follow a lifestyle that stands on the ability to travel within the city for both work and leisure. Because of these needs, the city road planning has become a major issue in the last decades. Due to historical reasons mainly concerned to the unsupervised development of the cities, problems like narrow roads, lack of parking places and the inexistence of alternative routes for some critical places makes city road network planning a very complex problem. Associated with this reality, a number of problems arise, being traffic congestions the one with the most impact in the daily life of the citizens. This particular point can be measured by the time spent on the road to complete relatively short paths, which contributes not only to a decrease in the lifestyle quality but also to possibly aggravate existing health issues such as irritability, stress among others. Furthermore, these congestions also lead to environmental problems with the increased CO_2 emissions as well as other polluting chemicals. Also, economical issues arise, from the increased fuel consumption in traffic congestions, allied with the increasing crude prices.

In order to address the road network planning issue, the first step consists in collecting real data in order to identify the critical problems in a specific city context. The use of a simulator capable of providing visual feedback based on the collected data not only allows for an easier identification of those critical problems but can also be used as a testbed for future developments to the road network. Theses developments can be composed by changes to the structure of the road network itself or new routing mechanisms.

In this work, which is part of the COSMO project (described bellow), we focus on the use of a simulator to interpret and simulate the city traffic based on the collected data. For that, Coimbra city was used as the first case study mainly due to its topography.

The remainder of this paper is organized as follows: Section 2.3 presents the Cosmo Project, Section 2.4 illustrates the literature review regarding to traffic simulators, Section 2.5 details the selected traffic simulator. Section 2.6 pinpoints and discusses the obtained results. Section 2.7 presents final remarks about the approach and discusses future work.

2.3 COSMO Project

In the COSMO project (COllaborative System for Mobility Optimization) the authors propose to look at the city as a Complex System, where each citizen/driver is an agent with a local vision of the environment. The principle is to use heterogeneous information collected on city mobility (GSM, GPS, Road Sensors, Information Services, Historical data) to exercise influence on the individual agent behavior in order to optimize the city efficiency (e.g. energy consumption). This way, when reaching pre-congestion levels of network charge, individual drivers will be lead to collaborate in alternative, and minimally competing, routes. In this scenario, several research challenges are raised:

- How to predict distributed network load? In order to prevent the appearance of congestions, these situations have to be detected in early phase of their formation, and the individual behavior of the agents should be led to a synchronized collaborative behavior in a way that increases the whole system efficiency;
- How to fairly synchronize drivers? The system cannot favor one driver over the other more than within reasonable limits. Mutual dependencies will increase considerably the complexity of the choice;
- How to communicate with individual drivers? Will it be realistic to assume that high quality wireless access will be available in every vehicle? Or would the traditional Variable Message Signs become a proper option? If so, where should these signs be placed?

These challenges are tackled from the perspective of Complex Systems, Ubiquitous Computing and Intelligent Transport Systems. The main purpose of this project is to study these issues and develop a collaborative traffic routing/control system. The system should use the collected information to predict realistic traffic network load and present advice to users, by providing individually tailored network loads, in order for them to have a less selfish behavior. The developed system is to be applied in a controlled micro-simulation environment, for this point SUMO was chosen, which allow the testing and validation of the various aspects of research, as well as the study the behavior of the system according to a number of dimensions, including driver adherence rate, link capacity, driver profiles and synchronism model.

2.4 Related Work

Simulators are part of the category of analytical tools that are used to analyze various types of data, such as traffic flows, financial transactions or pathogens spreading disease through a population. They make it possible to assess the effects of various changes to the environment without altering the real world making a cheap and safe way to predict the effectiveness of the proposed modifications.

The simulation type that is the most adequate for this project is the traffic microscopic simulation that represents the process of creating a model of a certain network, observing and executing interactions between the agents of the system, which as previously referred, are an autonomous entity with individual characteristics, goals and decision making mechanisms that can greatly vary within a population. Chen and Cheng [4] analyzed a wide range of agent-based traffic simulations systems and divided them into five different categories:

- agent-based traffic control and management system architecture and platforms;
- agent-based systems for roadway transportation;
- agent-based systems for air-traffic control and management;
- agent-based systems for railway transportation;
- multi-agent traffic modeling and simulation.

Given the characteristics of this project, the adequate category is the multi-agent traffic modeling and simulation. In the mentioned paper the authors refer two open source agent-based traffic simulators:

- Multi-Agent Transport Simulation Toolkit (MATSIM) [7], a toolbox for the implementation of large-scale agent-based transport simulations and is composed of several individual modules that can be combined or used stand-alone. It allows for demand-modeling, traffic flow simulation and to iteratively run simulations.
- Simulation of Urban Mobility (SUMO) [2], a portable microscopic road traffic-simulation package that offers the possibility to simulate how a given traffic demand moves through large road networks.

In addition to these two simulators, other solutions exist, such as Vissim [5] or Transims [8], among others. Some comparison studies have been performed between two or more of these simulation platforms [9][3][6]. Based on some of these studies, and the more recent developments made to each of the simulation platforms, SUMO was the adopted one. This decision was made partly due to the commercial nature of Vissim (which presented a very good performance), as well as the recent developments that eliminated some of the drawbacks presented by SUMO in earlier analyzed versions.

2.5 Simulation Environment

As mentioned before, several simulation platforms were analysed; of these, SUMO (Simulation of Urban MObility) was chosen as the traffic simulator. SUMO is an open source tool and a microscopic road traffic simulation package that supports different types of transportation vehicles. Every vehicle has its own route and moves individually through the network. This tool supports traffic lights and is space continuous and time discrete (the default duration of each time step is one second). There are three main modules in the SUMO package:

- SUMO, which reads the input information, processes the simulation, gathers results and produces output files. It also has an optional graphical interface called SUMO-GUI;
- NETCONVERT, a tool to simplify the creation of SUMO networks from a list of edges. It reads the input data, computes the input for SUMO and writes the results into various

output formats, such as XML, CSV or VISUM-networks. It is also responsible for creating traffic light phases;

- DUAROUTER, a command line application that, given the departure time, origin and destination, computes the routes through the network itself using the Dijkstra routing algorithm.

As input data, SUMO needs three main files, representing routes, nodes and edges. The nodes and edges files represent the vertexes and edges in the road graph, respectively. The routes file represents the traffic demand and includes information about all the agents involved in this simulation and their characteristics (departing time, maximum acceleration, maximum deceleration, driving skill, vehicle length and color) and route (list of edges).

In terms of outputs, there are different types available, such as:

- a raw output that contains all the edges and all the lanes along with the vehicles driving on them for every time step, which results in a considerable large amount of data;
- log-files created by simulated detectors (a simulation of induct loops with the ability to compute the flow, average velocity on the lane, among other values) are written using the CSV format. This data can be aggregated for specified time intervals which may be configured by the user.

Finally, this simulator offers a way to measure some metrics such as fuel consumption or pollutant emission, based on the Handbook of Emission Factors for Road Transport (HBEFA) database. According to HBEFA's website[5], it was "originally developed on behalf of the Environmental Protection Agencies of Germany, Switzerland and Austria" and is now also supported by Sweden, Norway, France and the JRC (Joint Research Centre, of the European Commission). It provides emission factors per traffic activity, i.e., it offers a way of measuring CO_2 emissions and fuel consumption, among other pollutant factors, for various vehicle categories (such as passenger cars, light duty vehicles, heavy duty vehicles, buses, coaches and motorcycles), being suitable for a wide variety of traffic situations.

2.6 Experimental Results

In order to be able to run tests in Coimbra, we firstly needed to obtain a map of the city, compatible with SUMO. The SUMO module Netconvert offers a way to import digital road networks from various sources (such as VISUM or OpenStreetMap) and creates networks usable by the other SUMO modules.

Regarding the inexistence of Coimbra network in VISUM or OpenDrive, we opted the OpenStreetMap format (Figure 2-1). By accessing the OpenStreetMap website we can search for the desired city and easily export that data into an XML file.

[5] More information available online at www.hbefa.net

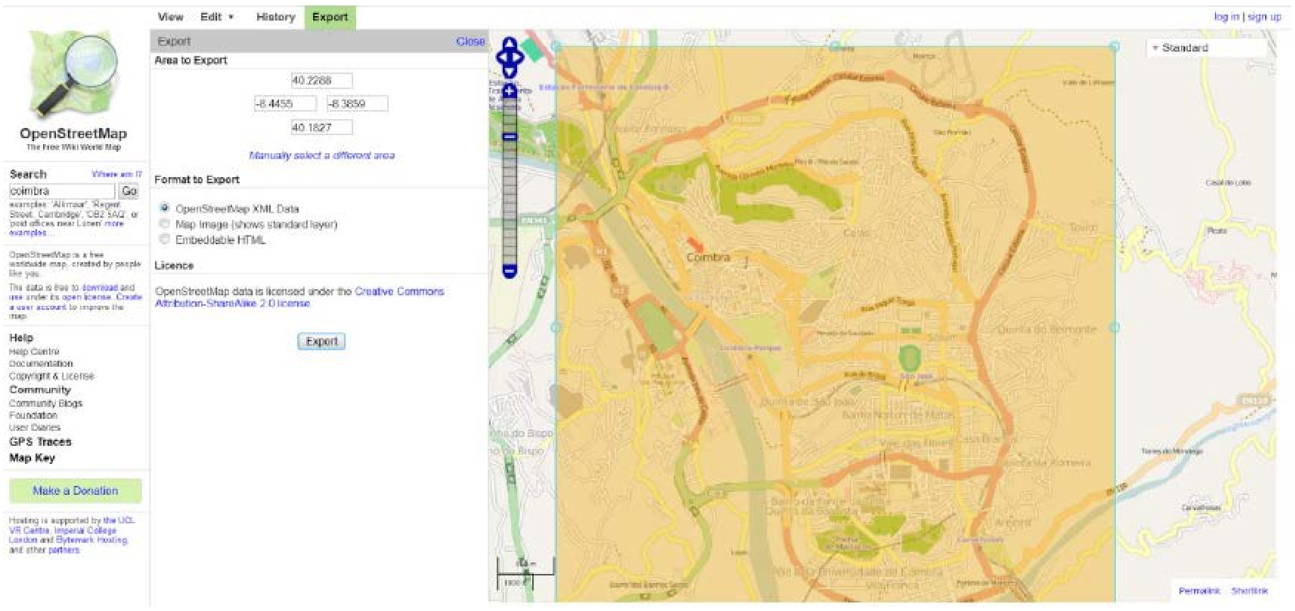


Figure 2-1: Front End of the OpenStreetMap website for Coimbra city

After this process, we experimented with the generated network in order to evaluate the outcome of this conversion. We found out that there were some errors in the generated SUMO network such as the conversion of pedestrian roads into vehicle roads. In order to filter out the pedestrian roads, and also to correct some other minor errors, the JOSM tool was used, which is an OpenStreetMap editor written in Java. The current version of SUMO offers the option of filtering out certain types of roads while converting a map, but at the time this option was unavailable. Another problem that we found, and corrected, was that the Netconvert module, by default, adds a turnaround possibility for almost every edge, making the network map unrealistic and somewhat confusing (Figure 2-2).

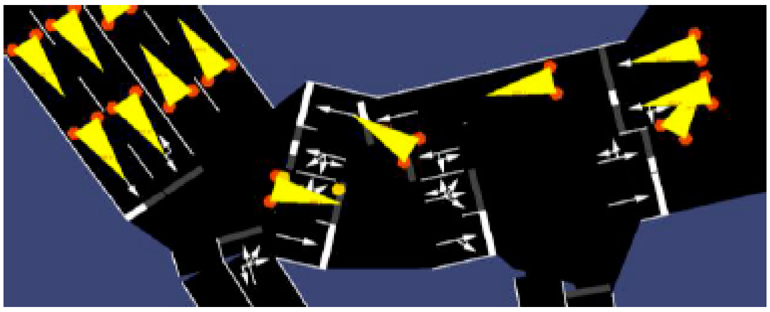


Figure 2-2: Edges with unrealistic turnarounds

Then we created a route set and using the SUMO GUI we identified the zones responsible for creating the more relevant bottlenecks. These areas contained several problems that range from an incorrect number of lanes to faulty connections between edges. Having corrected the previously referred faults, we believed the map to be ready for the testing process.

2.6.1 Coimbra Routes

With a corrected version of the map, it was time to generate the correspondent traffic demand. To complement the real city map we needed realistic traffic information. The work by Dr. Álvaro Seco and Nuno Norte Pinto [10] provided this information in the shape of an Origin-Destination (OD) matrix regarding the city of Coimbra. An OD matrix consists of a table that matches trips origins and destinations and also displays the number of trips going from each origin to each destination. In order to create the traffic demand based on this OD matrix, we first needed to analyze its data. This matrix was divided into several zones as can be seen in Figure 2-3 and these zones referred to inner and outer city traffic.

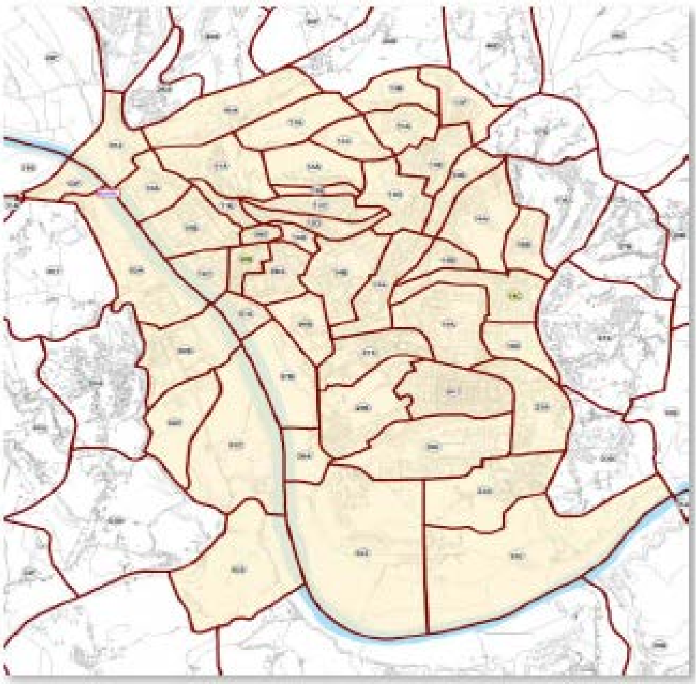


Figure 2-3: Coimbra OD matrix zones

There were numerous zones referred to outer city traffic so, given that in this project we were mostly interested in studying traffic within a city, we decided to group these several outer city zones into 7 zones, which match the existing city entrances (Figure 2-4): North (IC2), Northeast, East, Southeast, Southwest (Europa Bridge), West (Santa Clara Bridge) and Northwest (Açude Bridge).

The data in this matrix comprises 60.000 trips and corresponds to the morning period (7h30 to 10h30) with the duration of three hours. To make the testing process feasible, given the considerable map size, we decided to use 10.000 trips with an insertion period of one hour, i.e., vehicles are inserted at uniform time intervals during one hour. To make simulation environment more realistic, we inserted 2.000 vehicles at the beginning of the simulation (in the first time step) while the rest of the vehicles were inserted during the simulation. In SUMO, route definitions are as follows:

```
<vehicle id = "0" type = " type1 " depart = "0" color = "1,0,0">
<route edges = " id_beginning id_middle id_end"/>
</vehicle>
```

This means that each trip is defined as a collection of the id of the edges that comprise that trip. The main difficulty was establishing the match between the OD matrix's zones and the ids present in the SUMO network. Firstly we experimented with two id values per zones but as a result the entrance of cars in the network was very slow. This is due to the fact that SUMO inserts vehicles in a pre-determined position in each edge and does not insert another vehicle unless there is space for that insertion. This means that the simulator would have to wait until there was enough room to insert each new vehicle, greatly increasing the simulation time. It also means that it would not be possible to achieve the desired effect given that this fact significantly delayed the insertion of vehicles and making it impossible for some vehicles to be inserted within the previously defined time slots.

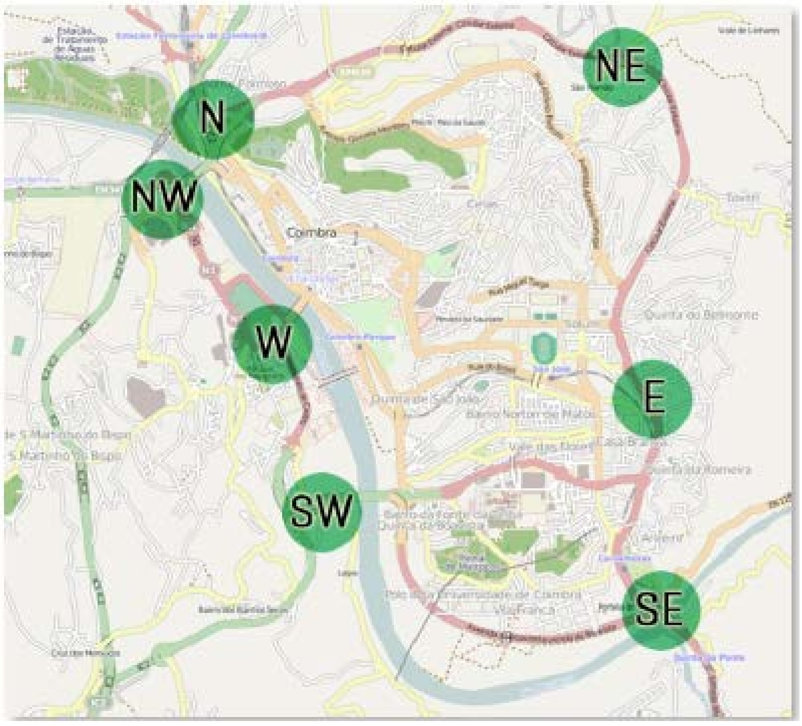


Figure 2-4: Coimbra city entrances

In order to overcome this limitation, more id values were added to the trip generation process. It was not possible to define a constant number of ids per zone given that the number of roads in each zone was not uniform. However an average amount of approximately 7 id values was defined per zone.

This id search process lead to significant time consumption given that most zones defined in the OD matrix do not exactly match street names and so these id values were manually defined. However, it allowed for a much smoother insertion of vehicles.

2.6.2 Coimbra Map Problems

After experimenting with the OD routes we concluded that the results were still not satisfactory. An experiment with 10.000 vehicles lasted 27.000 seconds, i.e., approximately 7 hours, which is a very large and unrealistic amount of time, when compared to travel times observed by the authors on a daily basis.

We came to the conclusion that this was due to problems with the simulator and also with the used network. So, we started to study how to further improve the quality of the network in order to obtain more realistic results. The following figures show some of the problems we encountered.

In the leftmost image of Figure 2-5 we show an example of halted vehicles, each queue apparently blocking each other. In the rightmost image we see the red vehicles giving total priority to the blue vehicles, waiting until there was no blue vehicle left in order to advance. It appears that SUMO has some issues when it comes to detecting whether or not the leader is occupying the junction and therefore blocking the passage. In Figure 2-6 we can see that, although the vehicle on the right lane (highlighted in green) wants to go forward and there does not seem to be any obstacle in front of him, it does not advance. However it considers that the other vehicle is in its way and therefore will only move once the other vehicle has also moved, causing an unrealistic and unnecessary queue.

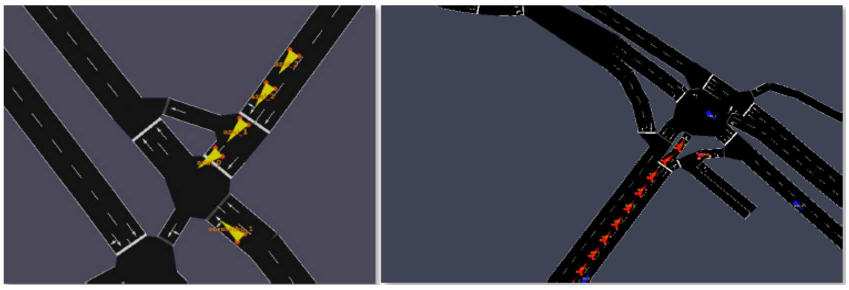


Figure 2-5: Hesitation situations

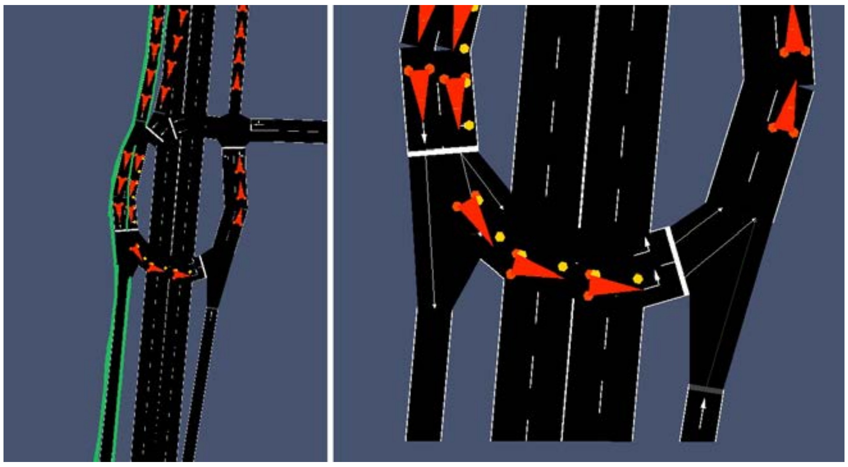


Figure 2-6: Junction problems

We noticed that some of these zones lacked traffic lights - that were somehow lost in the conversion. Therefore, with the intent of reducing the hesitation that appears to occur in Figure 2-5, we added traffic lights in the areas that were most affected by congestion: Portagem, Avenida Fernão de Magalhães and Casa do Sal. The addition of traffic lights appears to improve the ordering of traffic, reducing the hesitation phenomena. However, while solving one problem it creates another - due to the addition of traffic lights in some junctions, vehicles are often stopped at intersections (Figure 2-7).

SUMO is a collision-free simulator and therefore safety is paramount. So its vehicles require a large gap to competing vehicles in order to decide to move forward as a few experimentations demonstrated. This highly defensive attitude can partially justify this problem.

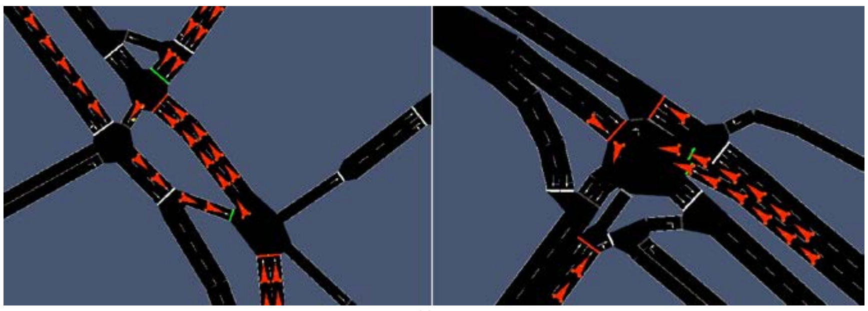


Figure 2-7: Traffic light problems

2.7 Conclusion and Future Work

We described the representation of Coimbra's road network within the SUMO simulation environment. We start by analyzing the construction of the road network of Coimbra by importing data from Open Street Maps and the issues that this operation raises. Although some issues have been detected, these have been satisfactorily overcome. To test the simulation environment we employ real traffic data, in the form of an origin-destination matrix, gathered from the city of Coimbra. The use of this data in the simulation is also discussed and analyzed, allowing us to identify the situations where the behavior of SUMO's vehicles deviates from the expected behavior of Coimbra's drivers. We indicate and discuss the mitigation actions taken to minimize these issues and the consequences of such measures. Overall, the results indicate that SUMO was capable of simulating the Coimbra urban traffic in a realistic fashion, validating its adequacy for testing routing algorithms in real-world scenarios. The areas identified in the previous section as critical in terms of congestions correspond to the actual critical areas where congestions occur in real life. Furthermore, it has sometimes been observed that a congestion in one of those critical spots (Portagem) propagates to other spots (Casa do Sal, ...), which, given the city road network configuration, can cause a city-wide congestion, which limits traffic in and out of the city. These problematic areas and their tendency towards congestion propagation were correctly identified in the simulator, which together with the above mentioned problems in network representation leads to the large observed congestion times.

Future research will focus on the development and test of distributed route planning algorithms, which should minimize travel time while promoting fuel efficiency, and reducing CO_2 emissions. The preliminary results obtained using a nature-inspired approach - namely, on ant colonies - are promising indicating significant improvements on all considered aspects.

Acknowledgments. This work was partially funded by the Portuguese Foundation for Science and Technology(FCT) under project COSMO with reference PTDC/EIA-EIA/108785/2008.

2.8 References

[1] Jeffrey L. Adler, Goutam Satapathy, Vikram Manikonda, Betty Bowles, and Victor J. Blue. A multi-agent approach to cooperative tra_c management and route guidance. Transportation Research Part B: Methodological, 39(4):297 - 318, 2005.

[2] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. SUMO - Simulation of Urban MObility: An Overview. In Proceedings of the Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona,Spain, pages 63-68, October 2011.

[3] Loren Bloomberg and Jim Dale. A Comparison of the VISSIM and CORSIM Traffic Simulation Models. In Institute of Transportation Engineers Annual Meeting, August 6-9, 2000, Nashville, Tennessee, USA, August 2000.

[4] Bo Chen and Harry H. Cheng. A Review of the Applications of Agent Technology in Traffic and Transportation Systems. IEEE Transactions on Intelligent Transportation Systems, 11(2):485-497, June 2010.

[5] Martin Fellendorf and Peter Vortisch. Fundamentals of Traffic Simulation, volume145 of International Series in Operations Research & Management Science, chapter Microscopic Traffic Flow Simulator VISSIM, pages 63-93. Springer, 2010.

[6] Wenli Gao, Michael Balmer, and Eric J. Miller. Comparisons Between MATSim and EMME/2 on the Greater Toronto and Hamilton Area Network. Transportation Research Record: Journal of the Transportation Research Board, 2197:118-128,2010.

[7] Andreas Horni. A Brief Overview of the Multi-Agent Transport Simulation MATSim. In MATSim Tutorial, May 17-20, 2011, Shanghai, China. Tongji University, May 2011.

[8] John D. Leonard II and Stephanie Lynne Shealey. Transims and Developments of Regional Impact. Final report, School of Civil and Environmental Engineering, Georgia Institute of Technology, September 2010.

[9] Michal Maciejewski. A Comparison of Microscopic Traffic Flow Simulation Systems for an Urban Area. Transport Problems, 5(4):27-38, 2010.

[10] Álvaro Seco and Nuno Norte Pinto(2002) Matriz OD Coimbra 2002 - Justificação da Metodologia e Apresentação

[11] Tomohisa Yamashita, Kiyoshi Izumi, Koichi Kurumatani, and Hideyuki Nakashima. Smooth traffic flow with a cooperative car navigation system. In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '05, pages 478-485, New York, NY, USA, 2005. ACM.

3 A Use Case for SUMO: Simulating Traffic around the Port of Duisburg

Gerhard Hermanns, Thomas Zaksek, and Michael Schreckenberg;
Physics of Transport and Traffic, University of Duisburg-Essen, Germany

3.1 Abstract

In this article we present our method and results from the project CLUSTER, where we analyzed the traffic to and from the area of the domestic port of Duisburg. The traffic data was obtained by a detailed traffic census that distinguished between passenger cars and trucks and recorded their directions across junctions. The simulated road network consists of main roads in Duisburg and was imported from OpenStreetMap. Vehicles and routes for the simulation with SUMO were generated with the JTRROUTER module. Main results are the identification of bottlenecks, the deduction of main freight traffic routes to and from the port, and the prognosis of the impact of increased (freight) traffic flow.

Keywords: Traffic Census, Freight Traffic; Bottleneck Analysis; Traffic Guidance

3.2 Introduction

The CLUSTER (Conzepte für Logistik Und SchiffsTransport Entlang der Ruhr) project was a cooperation between the Development Centre for Ship Technology and Transport Systems (DST), the Port of Duisburg (duisport) and the University of Duisburg-Essen (UDE) from 2009 to 2012. The main subject of the project was the evaluation of possible new logistical areas in the domestic port of Duisburg. While the DST analyzed the nautical aspects of the problem in regard to possible trajectories of ships in different geometries, the UDE analyzed the roadside traffic in the surroundings of the port with emphasis on the outgoing and incoming road traffic of the port area.

The main goal was to analyze the influence of changes in the volume of traffic, especially with respect to several possible traffic bottlenecks such as roundabouts or a construction site on a bridge. Statistical traffic data was obtained via a detailed traffic census. We were able to identify bottlenecks on the main routes to and from the port, and we could provide a prognosis on the impact of increased freight traffic flows.

This article is organized as follows: In chapter 3.2 we describe in detail our approach of the traffic census, as the overall quality of this data is of fundamental importance for our analysis. In the third chapter we describe how we processed those data for the use in a SUMO simulation. We also explain difficulties we came upon during those preparation steps. In the final chapter we present the simulation scenarios and our main results.

3.3 Traffic Census

As a first step, traffic data had to be obtained for the considered parts of the city of Duisburg. As there was no available data from automated traffic detection within the city, data had to be obtained via a traffic census. Because of the limited possibilities of obtaining reliable data via a census only, the simulated city area was restricted to the northern parts of the city center of Duisburg and an area around the port in Duisburg-Ruhrort, north of the city center (see Figure 3-1, left). Due to limited resources, a careful selection of times and places of the census had to be made in consultation with our partners. The main routes of the considered traffic were identified by duisport and the resulting road network had an almost linear appearance (see Figure 3-1, right) since it consists mainly of the routes from the port to the autobahn and vice versa. There are routing alternatives outside the scope of the considered network though, but duisport and we considered analyzing them less important than a good coverage of the main roads. To limit the effect of open boundaries within the network (vehicles may simply vanish to or suddenly appear from parking lots, driveways, etc.), almost all streets connected to those routes had to be covered by the census as well. The resulting extracted network has an extent of about 3.8 km in both north-south and east-west directions.

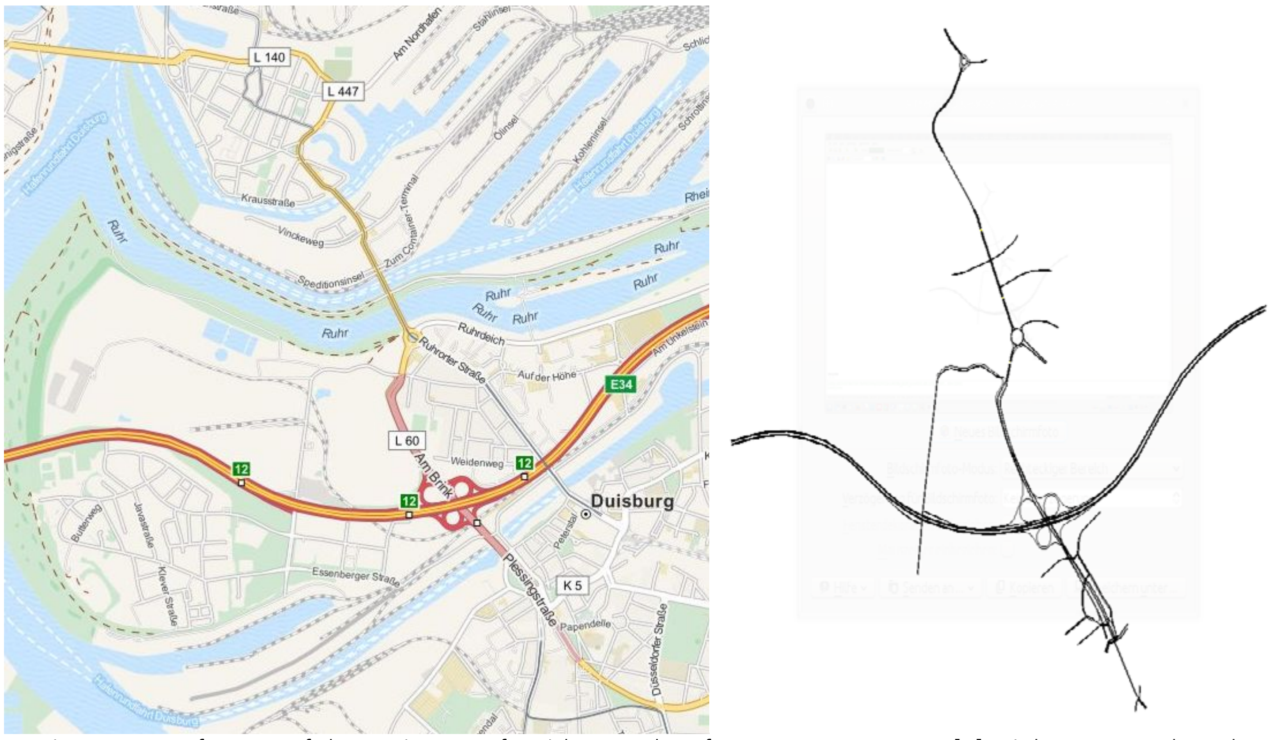


Figure 3-1: Left: Map of domestic port of Duisburg; taken from OpenStreetMap [4]; Right: Extracted road network of main streets in SUMO [5] (about 3.8 km in both north-south and east-west directions)

Within the analysis we did not only want to count the amount of vehicles on the main roads but also to determine their origin and destination, since there were no data on this matter available to our partners. Therefore, we decided not to count at cross sections of the streets but on the junctions, tracking the way of each vehicle directly and thus getting turning ratios at the junctions.

The counting positions were chosen such that we were able to cover the described network with 27 students counting simultaneously. By this method, we obtained traffic data of 66 "relations of traffic directions" (RTD), where an RTD is defined as a driving direction of a vehicle from one street across a junction to another street (see Figure 3-2). An additional of five RTDs was obtained from autobahn detector data. From these 71 RTDs we were able to derive 31 more that are linear dependent, resulting in 102 RTDs altogether.

Because the focus of this project was on freight traffic we also had to distinguish between passenger cars and trucks. As a rule of thumb, the vehicles were divided at a weight of 3.5t into passenger cars and trucks, respectively, giving way to a certain level of uncertainty due to subjective classification of vehicles by the students.

The vehicles were counted in 10-minute time intervals. The census took place on two consecutive Fridays (March 16th and 23rd, 2012) since Friday is the day of the week with the highest amount of traffic. As we wanted to cover the time from 7:30 a.m. to 1:20 p.m., the census had to be split into two days to avoid exhaustion of the students, with an overlapping interval due to calibration issues.

So on the whole we obtained data in 35 10-minute time intervals for passenger cars and trucks at 102 RTDs resulting in 7140 individual traffic flow values at junctions. This allows for a very detailed variation of the data for different scenarios.

The additional autobahn traffic data was obtained from the project OLSIM [1]-[3], a simulation of the freeways of North Rhine-Westphalia, based on an advanced version of the well-known Nagel-Schreckenberg model [6]. These traffic data are available in a higher, 1-minute resolution, so we aggregated them to the 10-minute intervals of the traffic census.

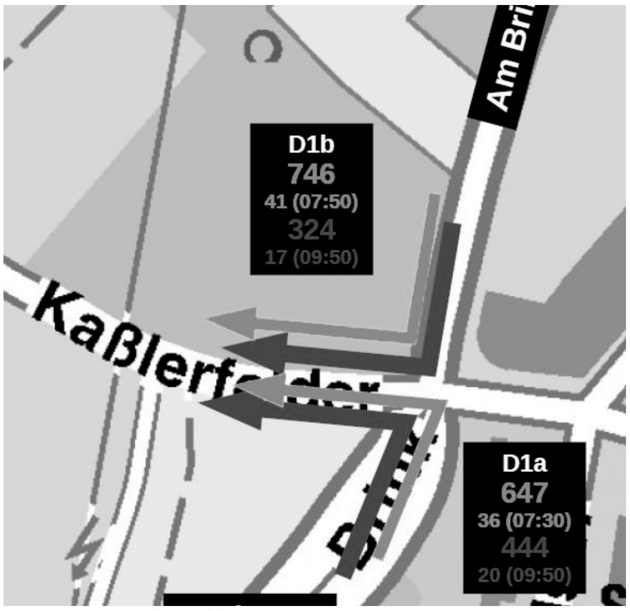


Figure 3-2: Example of RTDs (Relations of Traffic Directions) with census data. Light gray: passenger cars; dark gray: trucks. D1a and D1b are the abbreviations of two RTBs; the numbers describe the total sum of vehicles in the counting period (big numerals), and maximum values (small numerals).

3.4 Preparing the Simulation

The network was imported from OpenStreetMap (OSM) [4] via NETCONVERT after some editing (e.a. setting lane numbers and speed limits). The city of Duisburg provided us with information about the traffic light cycles. Some post-import edits had to be done for complicated parts of the network, especially for the roundabouts, the connections of the autobahn A40 to the city roads, and for turning restrictions. In some cases the right-of-way rules and intra-junction connections had to be corrected manually.

From the census data, turning ratios for all junctions were calculated individually for each lane connection per 10-minute time interval, both for passenger cars and trucks. The traffic flows were calculated for all 20 vehicle sources at the edges of the network. So for each 10-minute interval a pair of source flow rates and turning ratios for the whole network was given as input to the JTRROUTER module. We would like to point out that the use of an O/D-matrix approach for generating the vehicles was not possible because we did not possess the necessary information about destinations of the vehicles. This had to be derived from the census data via the JTRROUTER module first (see section 3.4.1).

Unfortunately, the vehicle input for SUMO [5] produced by our approach contains a non-negligible number of nonsensical routes: Vehicles entering and leaving the network at the same edge (only making a 180° turn at a roundabout), vehicles driving in circles, or leaving and re-entering the autobahn in the same direction as before. We could not use the NETCONVERT option "--no-turnarounds" or the JTRROUTER option "--remove-loops" because those would have prevented some routes that actually needed 180° turns due to turning restrictions. Therefore, we developed a list of "allowed" routes and replaced all nonsensical ones by them. This was done very carefully for each time-interval individually, conserving the correct turning ratios and source flows for this interval.

3.5 Census Analysis, Simulations and Results

3.5.1 Traffic Census Analysis

First, we analyzed the data obtained by the traffic census. Figure 3-3 shows the ratios of passenger cars and trucks, respectively, at the overall traffic for some RTDs for the whole time period of the census. Here, the highest ratio of trucks is at the RTD named "B1b" with about 47 %, the lowest ratio is at "G3a" and "G4a" with only 9 % each. Those are plausible results, since the former RTD describes the traffic into the port area and the two latter RTDs describe traffic near the city center. For a more detailed analysis we also took the total numbers of vehicles into account. This way we were able to identify main (freight) traffic routes in the network and confirm – and sometimes counterproof – assumptions made by our partners.

One important goal for our project was to provide information to duisport about the routes of freight traffic to and from the port. As mentioned in section 3.2 duisport already knew about the main freight traffic routes – but they needed more details about them. For example we wanted to know the percentage of trucks leaving the port via a certain route or driving in a certain direction, respectively.

With our method of recording the vehicles according to RTDs during the census we were able to derive routes via the JTRROUTER module. Figure 3-4 shows the distribution of directions for trucks leaving the port area at the street "Speditionsinsel", according to those derived routes: From 172 trucks leaving during the counting period of 6 hours, almost the same number drove towards the inner city of Duisburg ("DU-Kaßlerfeld", 33 trucks) as on a bypass road ("Ruhrdeich", 32 trucks). Interestingly the number of trucks for each of the two directions matches the overall number of trucks going onto the nearest autobahn A40, both directions together.

However, it is outside the scope of this article to go into the complete analysis that was done. Also, further analysis is done outside the project by our partner duisport directly.

At this point we would like to stress out that we are aware of the limitations of our analysis resulting from the census method, like open boundaries within our network or the uncertainty due to subjective classification of vehicles by the students. Also the census provided only data for a very limited period of time, no velocity data, and with the 10-minute intervals only a rough temporal resolution. Therefore it should be repeated for a more reliable data basis.

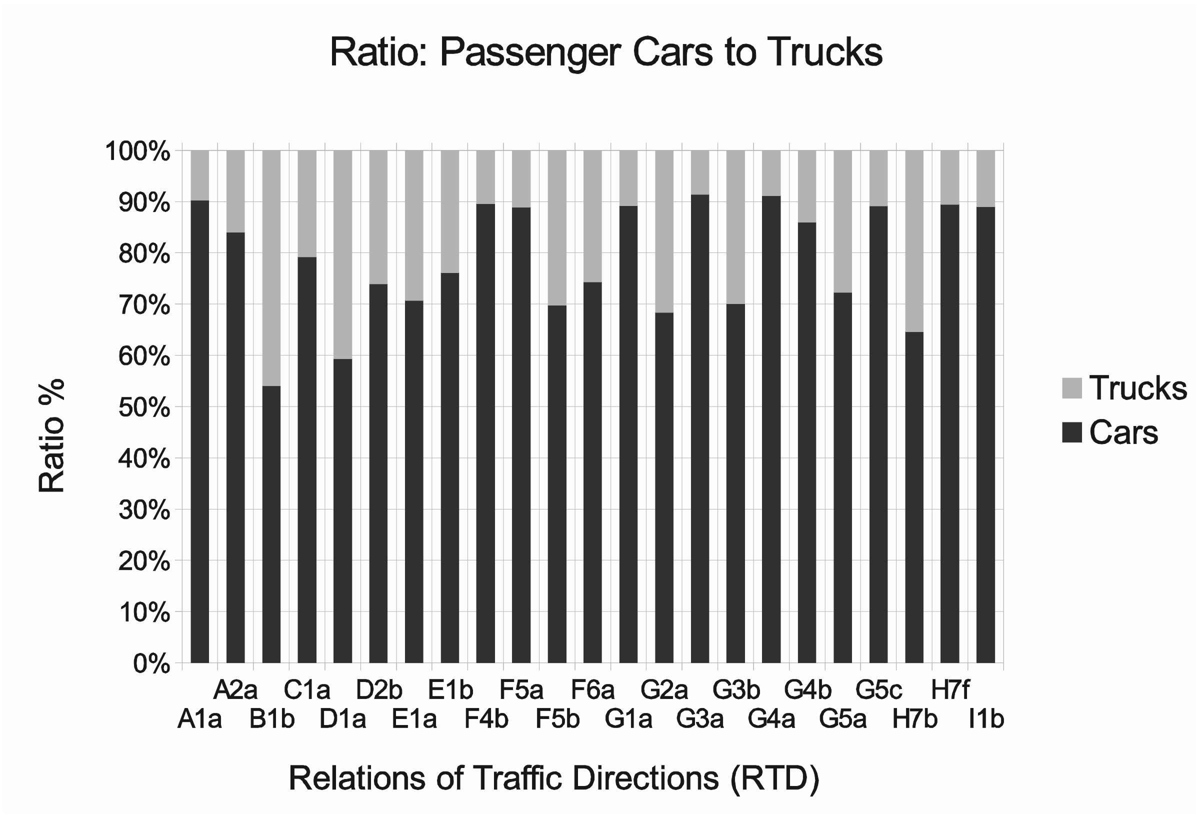


Figure 3-3: Ratios of passenger cars (dark grey) to trucks (light grey) for some selected RTDs. The abbreviations "A1a" to "I1b" describe the counting positions of the census and therefore the RTDs. Data according to traffic census.

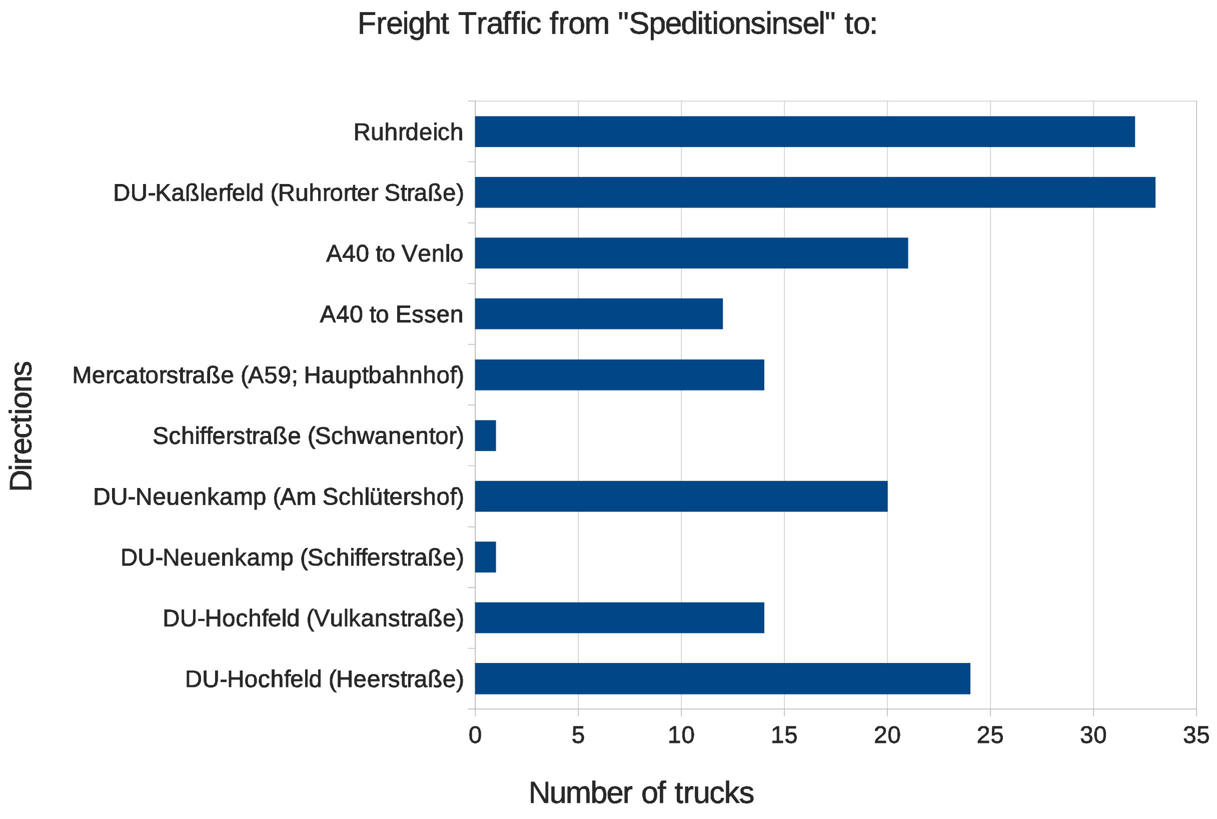


Figure 3-4: Distribution of directions for trucks that leave the port area from "Speditionsinsel". The directions are derived from the census data according to turning ratios at junctions, using the JTRROUTER module.

3.5.2 Freight Traffic Prognosis

To analyze the impact of an anticipated increase a) in overall traffic, and b) in freight traffic, we simulated different scenarios as decided in consultation with duisport. We performed three different kinds of simulations: At first, we manipulated the JTRROUTER input data by increasing the source flows to 110%, 120%, and 150% of the basic values from the census. Then, with the same percentages, only the freight traffic flow was increased while passenger traffic flow remained at the base value. Finally, we increased traffic from and to those edges only that directly connect to the port area, thus simulating a local change in traffic flow due to the opening of new logistical areas within the port. Since we did not have information about the actual planned areas, we made an analysis by example. Additionally, we investigated situations with artificial bottlenecks like a construction site at a major road as well as some other areas of interest to our partners. A more detailed analysis is done by duisport directly, as it is part of their business secrets.

In the following we will present exemplary results of our simulations. First, we were able to identify bottlenecks of the traffic flow in the network, both spatial and temporal ones by simulation of the real traffic situation based on the census. Figures 3-5 and 3-6 show a comparison of velocities within one of those bottlenecks from the second kind of simulations with increased freight traffic flow only. We compared speed and traffic flow on a single lane, which is part of a bottleneck at “Ruhrorter Straße”, between a simulation with data from the traffic census and scenarios with increased transport traffic flow rates as described above. It is important to stress out that the transport flow was only manipulated at the network sources with only an indirect influence on the flow at the bottleneck. The flow rates of passenger cars remained unchanged.

While the simulation based on the census dataset shows a stable velocity near the absolute velocity limit (50 km/h) on this track, some of the scenarios exhibit a different behavior with a breakdown of the velocity down to less than 10% of the allowed velocity. The scenario with a 10% extra transport traffic flow shows an unchanged behavior in the velocities. The scenarios with plus 20% and plus 50% extra traffic flow exhibit heavy breakdowns in the velocity with an increase of this behavior with increasing flow. This leads to the conclusion that small increases of the transport traffic up to about 10% have a non-disturbing influence on the simulated area around the port while higher flows can lead to a traffic breakdown and jams.

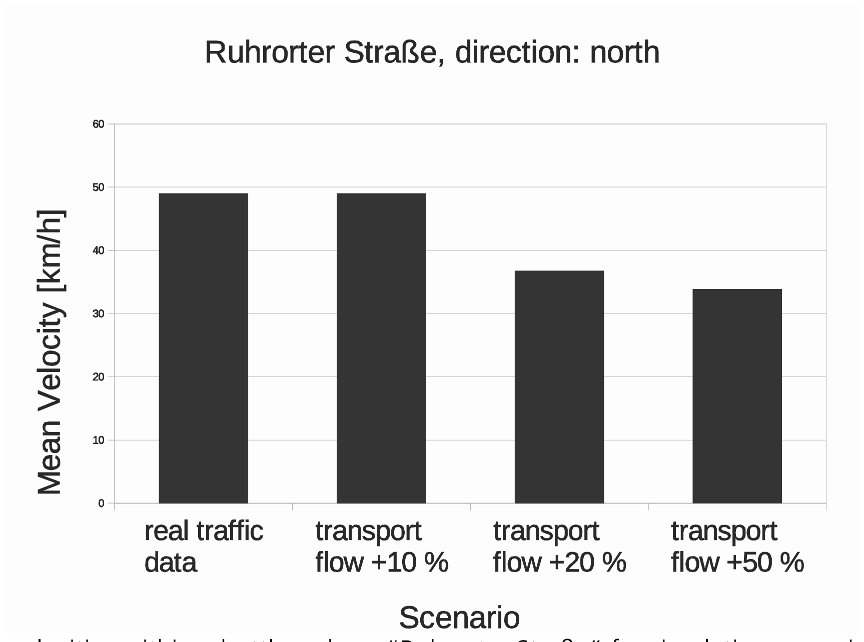


Figure 3-5: Mean velocities within a bottleneck on “Ruhrorter Straße” for simulation scenarios with an increase of freight traffic flows at the sources of the network in comparison with simulation data based on traffic census.

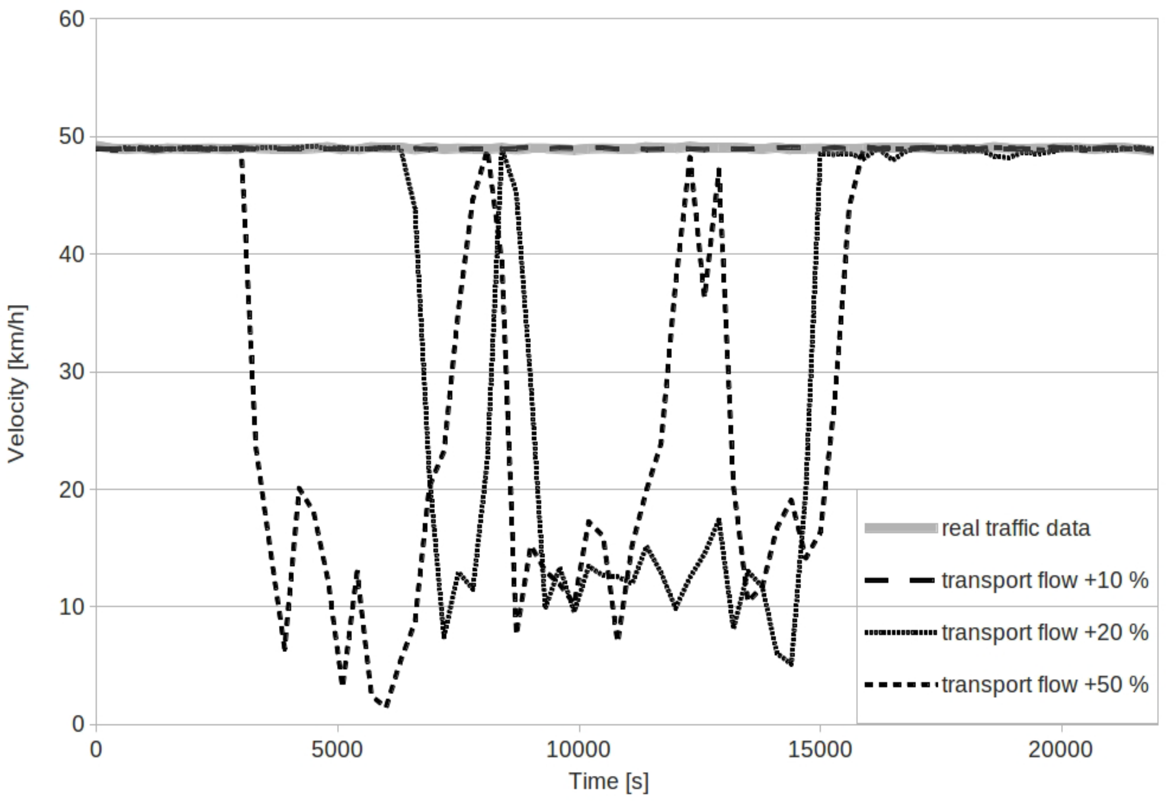


Figure 3-6: Simulation results for velocities of one lane within bottleneck for overall traffic in 4 different scenarios: i) basic traffic data from census, ii) – iv) freight traffic increased by 10%, 20%, and 50%, respectively, with unchanged amount of passenger cars

It should be noted that at very high increases of the traffic flows congestion up to the source edges and backlog prevent the simulation to insert all vehicles on time and the traffic state in the network is dominated by saturation effects.

3.6 Conclusion

In this article we described a method for obtaining detailed traffic data via a traffic census. With only 27 counting positions we were able to get 7140 individual traffic flow values at junctions over a period of 6 hours, distinguishing between passenger cars and trucks, in a network of main streets that spans about 3.8 km in each direction. This network was imported from OpenStreetMap into SUMO with some additional manual editing.

From the census data we generated the vehicles for the simulations with the JTRROUTER module. Again, we had to edit the results as this method produced a non-negligible number of nonsensical routes that we had to replace carefully.

We made three kinds of simulations based on this data, where i) the overall traffic flow was increased, ii) only the freight traffic flow was increased, and iii) only traffic from and to those edges that directly connect to the port area was increased.

The main results are the following: From the census data we were able to identify main (freight) traffic routes in the road network and to confirm – and sometimes to counterproof – assumptions made by our partners. From the simulations with SUMO we were able to derive the distributions of directions for trucks leaving the port area. We determined local bottlenecks of the traffic flow in the network, both spatial and temporal ones, and we were able to give some prognosis about the impact of an expected increase in (freight) traffic.

The results presented in this paper are only exemplarily, as a further detailed analysis is done by our partner directly. However, we found that SUMO is a suitable simulation framework that allows for a detailed analysis of traffic within a city.

3.7 Acknowledgement

Funding for this work has been provided by the state of North Rhine-Westphalia and the European Union within the NRW-EU Ziel 2 program 2007-2013 (EFRE).

3.8 References

[1] Pottmeier, A., Chrobok, R., Hafstein, S. F., Mazur, F., Schreckenberg, M.: OLSIM: Up-To-Date Traffic Information on the Web. In: Proceedings of the Third IASTED International Conference on Communications, Internet, and Information Technology. St. Thomas, US Virgin Islands, November 2004, pp. 571-576

[2] Pottmeier, A.: Realistic Cellular Automaton Model for Synchronized Two-Lane Traffic - Simulation, Validation, and Applications, Dissertation, Universität Duisburg-Essen, 2007

[3] Verkehrsinformationssystem autobahn.NRW, http://www.autobahn.nrw.de

[4] OpenStreetMap, http://www.openstreetmap.org/index.html

[5] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO - Simulation of Urban MObility: An Overview In: SIMUL 2011, The Third International Conference on Advances in System Simulation, 2011

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

4 Real-Time Traffic Conditions with SUMO for ITS Austria West

Karl-Heinz Kastner, Robert Keber, Petru Pau, and Martin Samal;
RISC Software GmbH, Austria

4.1 Abstract

ITS Austria West is a long-term project, funded by the Austrian government, to create a platform for providing real-time traffic data and prognoses. The calculation of the traffic situation is based on the open source package SUMO (Simulation of Urban MObility). One innovation in this project is the scenario builder, a module which maintains the origin/destination matrix and aggregates data from various sources, like traffic counters and floating car data. The simulation model is continuously calibrated in order to provide an accurate view of the traffic situation in the whole street network. The calculated level-of-service information is visualized and exported to other intelligent transport systems. Due to the complexity of the model (streets, vehicles and real-time data), the simulation's performance is unsatisfactory. We describe our attempt to a possible improvement: the parallelization of SUMO.

Keywords: traffic simulation, traffic condition, parallel computing.

4.2 Introduction

In the future, Intelligent Transport Systems (ITS) will make a significant contribution to secure our long-term mobility. Key challenges are intelligent handling of the increasing traffic by use of efficient multimodal transport and the most balanced use of transport infrastructure. In this respect, the availability of reliable real-time traffic information to all stakeholders (institutions, companies and individuals) is necessary.

The project ITS Austria West aims at providing a traffic data platform which generates accurate traffic situation information based on a (limited) number of traffic data sources and on historical information regarding vehicle trips on Upper Austrian roads. For the computation of current status of each road, as well as for short- and mid-term prognoses, the traffic simulation package SUMO is used.

In this paper we describe the concepts of ITS Austria West and focus on the integration of SUMO, and its results, into the real-time traffic data platform.

Section 4.3 contains an overview of the ITS Austria West project, its architecture and short descriptions of the main components. In section 4.4 we describe the integration of SUMO package within ITS Austria West and give some details about our approach to a parallel SUMO. Section 4.5 contains some ideas about consuming the results – using the traffic information provided by SUMO to define level-of-service values that can be used to describe synthetically the current traffic status on Upper Austrian roads. We end with some conclusions and future extensions.

4.3 ITS Austria West

The goal of the project ITS Austria West is a real-time snapshot of the traffic situation in Upper Austria and Salzburg. Project partners are the state Upper Austria, the state Salzburg, RISC Software GmbH, Salzburg Research and the institute Logistikum Steyr. The project is funded in large part by the Austrian KLIEN fond ("Klima- und Energiefond"). It started in June 2011 and ends in September 2014.

The street network is provided by GIP [2] ("Graphenintegrationsplattform"), a graph which comprises all Austrian highways, interstate and municipal roads. The GIP graph was created in the frame of a previous project conducted by all nine states of Austria. This graph is intended as a basis for many current and future applications and will eventually be open-data.

In order to create a real-time snapshot for the state of Upper Austria, up-to-date traffic data is vital. Various data sources are utilized: Firstly, about 50 *traffic counters*, owned by the Upper Austrian government, are installed on the main streets. The second data source is *road maintenance data*, also owned by the Upper Austrian government. Thirdly, the Austrian highway agency ASFINAG streams up-to-date information for the 2.178 km Austrian highways. This data is generated by traffic counters and traffic cameras covering the biggest part of Austrian highways.

Apart from that, *floating car data* (FCD) is integrated as an additional data source. In case of Upper Austria, "Samariterbund Linz", an emergency service, provides FCD for the city of Linz. Other FCD suppliers, e.g. taxi cabs, will be integrated at a later point of time.

4.3.1 ITS Austria West: Architecture

The ITS Austria West Management System consists of a management console, a simulation environment, a scenario builder, all incoming data and a visualization and export unit. Figure 4-1 gives an overview of the system architecture.

The controlling part of the ITS West Austria Management System is the management console, where all processes are created, parameterized and controlled.

The scenario builder, triggered by the console, is responsible for the periodic creation of new scenarios, which serve as basis for the simulation of the next step. This complex system part uses traffic data generated by previous simulations and combines them with real-time sensor data, according to parameters set by the console.

The demand model, i.e. the origin / destination matrix, is based on historical data regarding the traffic on Upper Austrian roads. From the raw data, trends are computed and, based on these trends, routes between every possible pairs of locations are generated, distributed during one day.

The traffic light system (VLSA configuration) contains locations of the traffic lights and the logic behind their functioning.

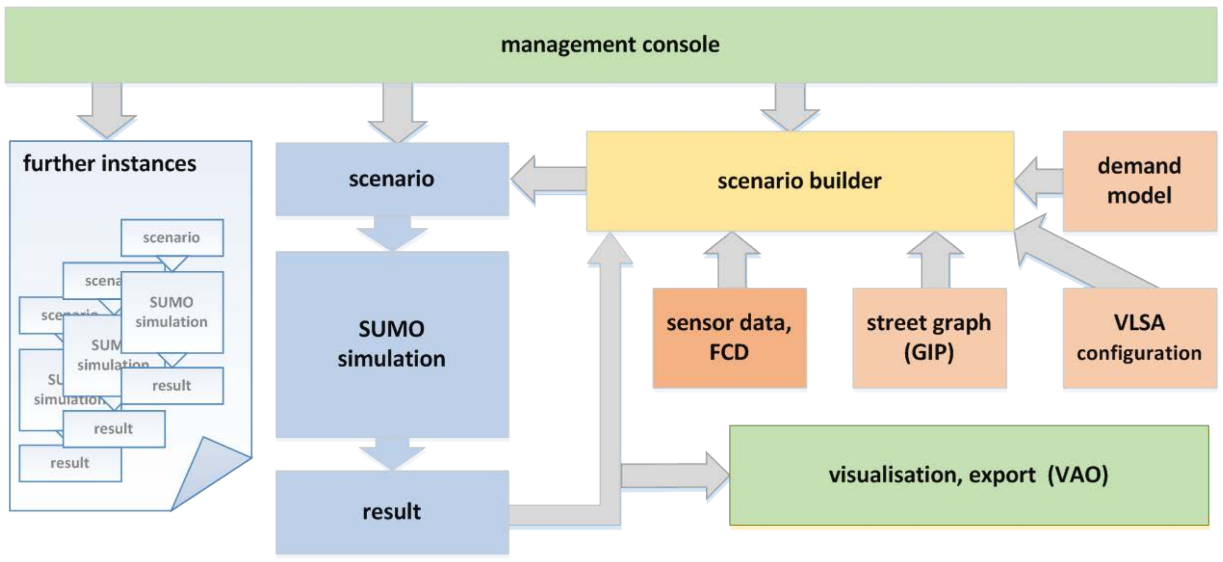


Figure 4-1: ITS Austria West Management System: System architecture.

The sensor data module receives information from different types of sensors:

1. Vehicle detection loops are stationary detectors which count the number of vehicles and their velocity. The suppliers are ASFINAG and the Upper Austrian government.
2. Floating car data is provided by an Austrian emergency rescue service "Samariterbund" (only when not in action) and a worldwide tracking company. "Samariterbund" tracks its own vehicles in order to optimize the rescue service. The anonymized data is received via a dedicated interface. The worldwide tracking company is a provider for GPS tracking hardware and software systems, to analyze, monitor and secure vehicle armadas. Negotiations with additional providers are currently in progress.
3. VLSA configuration is provided by the Upper Austrian government. It contains the traffic light phases which are taken into account during the simulation process.
4. Traffic Message Channel (TMC) is a technology for broadcasting real-time traffic and travel information to drivers via FM radio stations.
5. GeoRSS is an emerging standard for encoding locations as part of a web feed and will be used in Austria to deliver road work information. "Doris", an Upper Austrian government agency, provides this information in a special format.

All generated scenarios are transferred into a scenario pool. Periodically, the console triggers simulations of selected scenarios. For specific purposes, like calculating traffic forecasts or simulations with different parameters and data, parallel execution of different simulations is possible. Traffic data generated during these simulations are made available to various client applications over a TCP/IP Interface.

The visualization and export module is such a client application. It takes the calculated traffic data and provides a graphical real-time traffic view.

4.4 Traffic simulation

Traffic simulation applications are developed since the 1950's. The tools can be classified in microscopic, mesoscopic and macroscopic models.

In a microscopic model every vehicle is simulated. Such a model contains a detailed street graph, with traffic lights and turn lanes, and a demand model – an origin / destination matrix.

Mesoscopic models describe the traffic entities at an equal level of detail, but their behavior and interactions are described at a lower level of detail.

In macroscopic models all parts are simulated in a lower detail. In the ideal case, a macroscopic model is the generalized form of a microscopic one.

Main providers of traffic simulation applications are PTV (with VISUM), Aimsun, Caliper and UAF, but for development purposes, the free, open-source simulation tool SUMO is the most interesting.

4.4.1 SUMO

SUMO is an open source, microscopic simulation package developed by the German Aerospace Center (DLR) in 2001. It is not only a traffic simulation, but rather a suite of applications which help to prepare and to perform the simulation of traffic [1]. Sumo comprises utility tools, to import and generate graphs, to build trips & routes and to communicate.

The first step for the simulation is the generation of a street network. In Upper Austria two street graphs are currently available: An old VISUM graph, used in an earlier project as basis for a macroscopic simulation, and a modern, state-of-the art GIP graph [2] which is continuously maintained and extended. The current simulation is based on the VISUM graph, currently converted into a SUMO network with the tool NetConvert. Because of the different structure of the GIP graph, a separate tool has to be implemented in order to import it. Additionally, for testing purposes, different synthetic graphs are created with the tool NetGenerate.

In the second step, based on a demand model (origin / destination matrix), routes are built for the simulation on this graph. Within ITS Austria West, the scenario builder generates a trip file with start and end parameters based on a time variation curve. This provides the input for the DuaRouter tool, which generates the routes.

After this, the simulation can be started. In order to get good results, the simulation requires a well-tuned traffic lights signal system (VLSA), which is also generated by the scenario builder.

4.4.2 Simulation Run

For computing an accurate snapshot of the traffic, a simulation run starts every 5 minutes. SUMO loads a scenario from the scenario pool, which contains the graph with available vehicles from the last run, the new routes for the next 5 minutes (inserted at the beginning) and the traffic light system. After 5 minutes have been simulated, the results – i.e. the graph with the vehicles and the average travel times – are saved.

This will be the new input for the scenario builder, which will use:

- the real-time information from the sensor network to calibrate the demand model. Consequently, the recalculation of at least a part of the routes is necessary.
- the velocities provided by the floating car data to adjust the average calculated velocities on the edges of the output graph.

After these steps, the scenario builder generates a new scenario – the new input for simulating the next 5 minutes.

To generate every 5 minutes a new prognosis, the simulation must run considerably faster than real time. Several tests have shown that the rush hours, in Upper Austria, cannot be in fact simulated by SUMO faster than real time.

4.4.3 Parallelism

For real-life scenarios (containing, in our case, the whole street network of Upper Austria and the corresponding daily traffic), the number of processed elements is simply too big. The most immediate way to speed things up is parallelization of the software. To achieve this goal, the existing SUMO single-threaded calculation model should be reengineered and converted into a multi-threaded model, where the new positions of the cars are calculated in parallel.

We describe in the following our experiments for parallel SUMO and conclude with a few results. It will become apparent that a profound reengineering is necessary in order to obtain an efficient version of parallel SUMO.

For the simulation of the traffic in a working day, routes were generated according to statistical data received from official, governmental institutions. In Upper Austria only, the total number of vehicles driving daily exceeds one million. As expected, the highest density of vehicles occurs during the rush hours – 5 to 9, and 15 to 19 – when more than 100 thousand vehicles are simultaneously on the roads.

When inputted into Sumo, these routes lead to an unsatisfactory simulation speed of the traffic, starting from 6:00 am simulated time. The efficiency of Sumo decreases steadily, reaching levels where the simulated time runs slower than the real time.

This fact challenged us to look for ways to parallelize the heavy work load. A quick profiling showed the functions in which Sumo spends most of the time, during an internal simulation step (see Figure 4-2). Further investigations revealed that the function `MSTL-LogicControl::setTrafficLightSignals` can be easily parallelized, but for the other two relevant functions, `MSEdgeControl::moveFirst` and `MSEdgeControl::moveCritical`, the parallelization is not trivial. Also, when the number of routes starting in a small time interval is extremely high, the program spends a significant amount of time in the function `MSInsertionControl::emitVehicles` and therefore it also needs to be considered for parallelization.

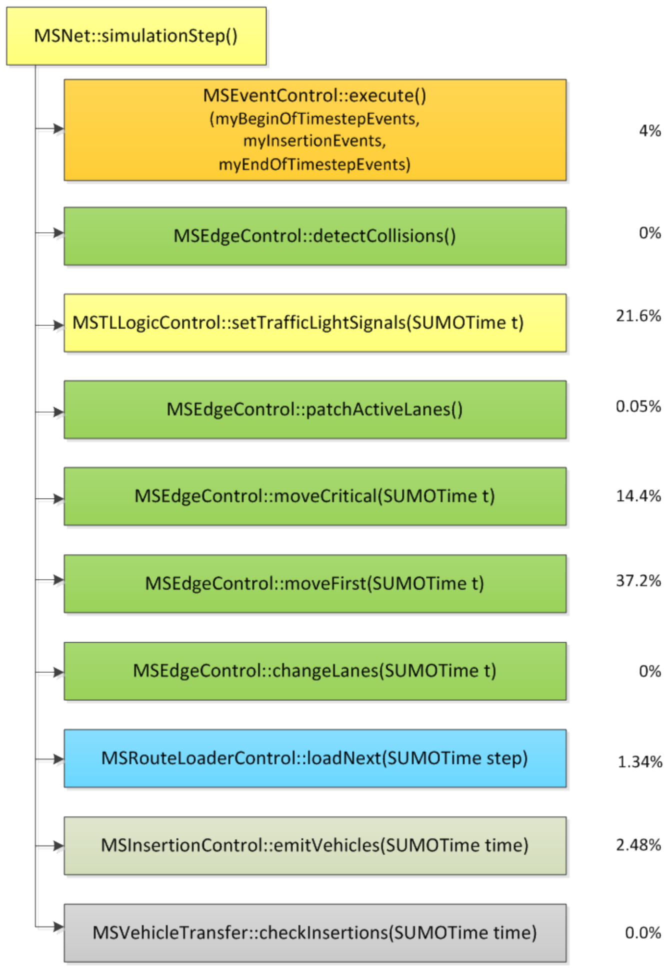

MSNet::simulationStep()
MSEventControl::execute() (myBeginOfTimestepEvents, myInsertionEvents, myEndOfTimestepEvents)
4%
MSEdgeControl::detectCollisions()
0%
MSTLLogicControl::setTrafficLightSignals(SUMOTime t)
21.6%
MSEdgeControl::patchActiveLanes()
0.05%
MSEdgeControl::moveCritical(SUMOTime t)
14.4%
MSEdgeControl::moveFirst(SUMOTime t)
37.2%
MSEdgeControl::changeLanes(SUMOTime t)
0%
MSRouteLoaderControl::loadNext(SUMOTime step)
1.34%
MSInsertionControl::emitVehicles(SUMOTime time)
2.48%
MSVehicleTransfer::checkInsertions(SUMOTime time)
0.0%

Figure 4-2: The most important functions invoked in a simulation step.

Our first attempt at parallelization involved OpenMP and was a brute-force approach. Steps in highest-level "`for`"-loops were executed in parallel; critical sections were defined to forbid concurrent access to data structures, if these data structures were being modified from at least one thread. This approach, its shortcoming and some results will be described in the next section.

All subsequent discussions refer to adapting and compiling SUMO within Microsoft Visual Studio 10, on Windows platforms.

4.4.4 Using OpenMP in a Minimally-Invasive Parallelization

In our first approach, we used OpenMP directives to define code blocks executed in parallel. Where concurrent access, especially in writing, was not desired, we defined critical sections.

The OpenMP standard provided by the C++ compiler in Visual Studio 10 does not allow parallelization of "`for`"-loops where the loop variable is an STL iterator. Since the main "`for`"-loops in the three functions mentioned above are driven by iterators over STL containers, some artificial constructs are needed in order to use the OpenMP "`for`" directive.

In fact, starting a parallel thread for each step of those for loops is not computationally efficient, due to the considerable overhead involved. We decided to split the affected STL containers beforehand; each part is then processed in its own thread.

This simple strategy is clearly effective when the STL containers do not change after initialization – like the elements processed in function `MSTLLogicControl::set-TrafficLightSignals`. Here, a number of iterator domains are defined on the container with traffic light logics, as soon as the initialization of this container ends. Using OpenMP "`for`" directive, in `MSTLLogicControl::setTrafficLightSignals`, each iterator domain is processed in a separate thread, in which a local iterator runs over the current domain.

The objects processed in the other two functions, `MSEdgeControl::moveFirst` and `MSEdgeControl::moveCritical`, are *active lanes*, i.e., lanes which contain vehicles. Active lanes can be removed from this container in both functions; lanes becoming active are added to the container in `MSEdgeControl::moveFirst`. A splitting strategy similar to that described above would need a re-computation of the iterator domains at the end of each of these two functions, and would involve essentially a sequential traversal of the container.

We use an alternative strategy, namely an array of STL containers. Each element of this array is processed in a separate thread, started by an OpenMP "`for`" directive, in both functions `MSEdgeControl::moveFirst` and `MSEdgeControl::moveCritical`. Lanes which were not active, but in the current step contain vehicles, are added after the parallel section of `MSEdgeControl::moveCritical` has finished; these new lanes are evenly distributed among the containers in the array.

Some internal data structures in classes `MSVehicle`, `MSLane` and `MSLink` are accessed and modified from functions that may be running, at the same time, in different threads. Since STL containers are not multithreaded-safe, mechanisms must be provided to avoid multiple access. The first idea was to use again OpenMP directives, namely to define *critical sections*.

With an OpenMP "`critical`" directive, blocks of code are marked with a name that identifies a critical section. If a thread gets hold of a marked block, any other thread attempting to run a block marked with the same name must wait until the running block is freed.

We defined a set of critical sections to restrict the access to internal data structures for vehicles, lanes and links.

This approach has two important shortcomings:

1. The critical sections restrict access to portions of code, rather than to shared data. As a consequence, processing the vehicles on a street in Berlin, for example, must wait until the same kind of processing finishes the vehicles on a street in Leipzig, although they can – and should – be processed in parallel without restrictions. In other words, the threads must wait for one another although there may be no reason to do that.
2. In every processing step (once every second simulation time) a number of threads are generated – once in `MSEdgeControl::moveFirst` and once in `MSEdgeControl::moveCritical` – and closed. While small, the involved overhead is not negligible.

The alternative to critical sections is to define *locks* (by means of, e.g., mutexes) as members of each instance of `MSVehicle`, `MSLane` and `MSLink`. Whenever a critical structure is being accessed, the lock shall be activated; at the end of the processing, the lock shall be released.

The conceivable overhead for this alternative is not negligible, both in memory consumption and computation effort. Our first experiments revealed that, indeed, parallelization with locking objects defined for each element (lane, link, and vehicle) is even slower than the sequential SUMO, and the memory consumption increases drastically.

4.4.5 OpenMP Parallelization: Results

In our tests we used a version of the modified SUMO console application (version 0.16) adapted for 2, 4 and 8 threads: Two, four and respectively eight iterator domains were defined for the traffic lights, arrays of containers with similar lengths for the active lanes. We ran this version on a computer with four cores with hyperthreading at 3.4 GHz, and 8 Gb RAM.

For a moderately large network and set of vehicles, the best results were obtained with the 4-thread parallel version. It finished the simulation in about 65% of the time needed by the sequential version. This network was randomly generated, with 150000 edges. The 20 000 routes were also randomly generated; the maximum number of vehicles simultaneously on the roads was around 3500.

For a large network (whole Upper Austria), after the first 25 000 seconds, the 4-thread parallel version took only 57% from the time needed by the sequential version. The maximum number of vehicles simultaneously on the roads was around 80 000; more than 263 000 vehicles have been generated.

Parallel versions with two and eight threads proved to be less efficient. In the 2-thread version, the overhead did not pay off: the gain in efficiency, for the parallelized part, was too small. In the 8-thread version, the number of threads competing to execute critical sections led to prohibitively long waiting times.

Exact results, with the time spent by SUMO in each relevant function, are shown in Table 4-1.

Table 4-1: Time spent by SUMO in the most relevant function, in a processing step.

	Random network *(150 000 edges 20 000 route)*	**Upper Austria** *(180 000 edges, 1 200 000 routes)*
Sequential	After 13600 steps and 182.661 seconds:	After 25000 steps and 1128.699 seconds:
	Timestep Events: 9.012 seconds	Timestep Events: 13.493 seconds
	setTrafficLightSignals: 35.213 seconds	setTrafficLightSignals: 62.747 seconds
	patchActiveLanes: 0.041 seconds	patchActiveLanes: 0.128 seconds
	moveCritical: 33.23 seconds	moveCritical: 348.401 seconds
	moveFirst: 39.986 seconds	moveFirst: 426.741 seconds
	changeLanes: 0.003 seconds	changeLanes: 95.919 seconds
	loadNext: 2.005 seconds	loadNext: 17.874 seconds
	emitVehicles: 2.148 seconds	emitVehicles: 34.950 seconds
	checkInsertion: 0.002 seconds	checkInsertion: 20.176 seconds
Parallel (2 threads)	After 13500 steps and 152 seconds:	After 25000 steps and 820.784 seconds:
	Timestep Events: 10.169 seconds	Timestep Events: 14.775 seconds
	setTrafficLightSignals: 29.004 seconds	setTrafficLightSignals: 40.176 seconds
	patchActiveLanes: 0.04 seconds	patchActiveLanes: 0.126 seconds
	moveCritical: 24.438 seconds	moveCritical: 231.840 seconds
	moveFirst: 30.979 seconds	moveFirst: 257.587 seconds
	changeLanes: 0.009 seconds	changeLanes: 102.652 seconds
	loadNext: 2.103 seconds	loadNext: 18.569 seconds
	emitVehicles: 2.322 seconds	emitVehicles: 34.468 seconds
	checkInsertion: 0.002 seconds	checkInsertion: 22.728 seconds

Parallel (4 threads)	After 13500 steps and 119.1 seconds:		After 25000 steps and 649.733 seconds:	
	Timestep Events:	9.085 seconds	Timestep Events:	13.687 seconds
	setTrafficLightSignals:	21.673 seconds	setTrafficLightSignals:	25.024 seconds
	patchActiveLanes:	0.036 seconds	patchActiveLanes:	0.113 seconds
	moveCritical:	15.3 seconds	moveCritical:	174.939 seconds
	moveFirst:	18.19 seconds	moveFirst:	173.349 seconds
	changeLanes:	0.009 seconds	changeLanes:	103.212 seconds
	loadNext:	2.01 seconds	loadNext:	18.653 seconds
	emitVehicles:	2.249 seconds	emitVehicles:	34.901 seconds
	checkInsertion:	0.001 seconds	checkInsertion:	23.766 seconds
Parallel (8 threads)	After 13600 steps and 133.35 seconds:		After 25000 steps and 798.479 seconds:	
	Timestep Events:	9.87 seconds	Timestep Events:	15.934 seconds
	setTrafficLightSignals:	26.465 seconds	setTrafficLightSignals:	33.021 seconds
	patchActiveLanes:	0.034 seconds	patchActiveLanes:	0.129 seconds
	moveCritical:	19.78 seconds	moveCritical:	209.453 seconds
	moveFirst:	26.96 seconds	moveFirst:	236.049 seconds
	changeLanes:	0.011 seconds	changeLanes:	144.329 seconds
	loadNext:	2.391 seconds	loadNext:	23.945 seconds
	emitVehicles:	2.951 seconds	emitVehicles:	43.803 seconds
	checkInsertion:	0.006 seconds	checkInsertion:	30.203 seconds

4.4.6 Alternative Approaches to Parallelization

The brute-force parallelization is inherently limited, if no additional information about the network and/or vehicle routes is employed. Generating a number of threads in each simulation steps, threads that afterwards compete to gain access to shared data structures, cannot lead to efficient, scalable parallel versions of SUMO.

A more elaborate approach at parallelization should start by splitting the underlying data structures using some topological criteria. From the original sets of roads, junctions, traffic lights, etc., *sub-networks* should be defined. The number of connections between different sub-networks should be as small as possible.

The simulation should proceed on each sub-network in parallel, so that each thread runs its own simulation loop. Synchronization points should be defined, in order to keep fast threads from running ahead. Also, vehicles that jump over sub-network boundaries should be processed in a sequential manner: It must be ensured that no multiple accesses to the data structures affected in this processing step occur.

It is clear to us that this approach requires a significant re-engineering of SUMO, starting with a design oriented towards parallelism. While the foreseeable effort is considerable, we are confident that a fully parallel, scalable version would provide the efficiency needed to simulate large-scale, real-life traffic situations.

4.5 Project Outcomes

Based on the information available during a simulation run, the level-of-service (LoS) should be calculated. The term “level-of-service” has been defined differently over the time. The origin of this definition comes from the transportation planning in the U.S., where this notion was defined for the service quality of road infrastructure. The traditional level-of-service is available in 6 categories, from A (for free ride) to F (for gridlock). The details are set out in the Highway Capacity Manual [3]. In Europe usually only the traffic-light colours are used (green:

free travel, orange: slow-moving traffic and red: traffic jam). Additionally, the colour Black can represent gridlocks.

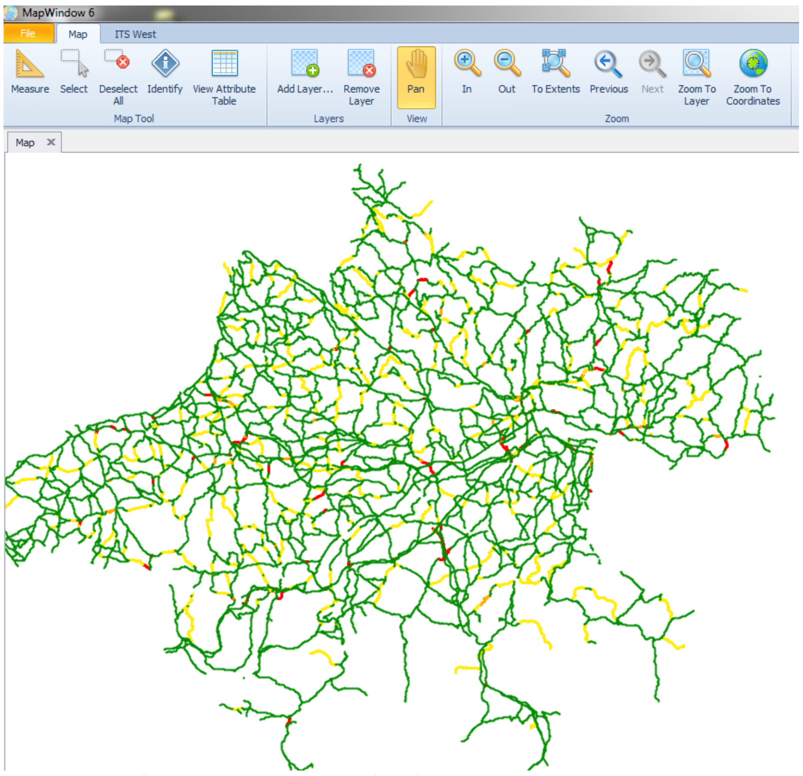


Figure 4-3: LoS mapping in an open-source GIS.

In our case, the level-of-service is computed from information associated to relevant edges during a SUMO run. Specific attributes, like the vehicle density or the average speed of the vehicles, are used to determine a LoS degree. As a result, every edge of the GIP graph with a corresponding density of traffic is assigned a level-of-service. The SUMO architecture allows for the necessary data to be exported and processed by other intelligent transport systems.

For online access during the simulation run of SUMO, a socket interface has been set up. At configurable time intervals during a SUMO session, a dump of the network status (e.g., vehicle density on edges) is streamed to this socket. Other applications can connect to this socket and read and process this data. The LoS values can thus be calculated from the information associated to each street.

In our prototype, the LoS information was generated from a SUMO instance with streaming enabled. In MapWindow, an open-source GIS, an add-on was implemented that reads the information from the socket and maps the relevant data to an appropriate attribute of a Shapefile layer. An attribute-dependent visualization definition shows now an actual LoS image. Figure 4-3 gives a snapshot of the Upper Austrian road network where the streets were coloured according to traffic data obtained from a running SUMO session.

In the future, the information shall be provided by means of *web map service* (WMS) layers to diverse information systems.

4.6 Conclusions and future work

In this paper we described a system that employs SUMO as a core simulation system for computing short-term traffic data starting from a given traffic situations and a dynamically adjusted demand model. The traffic status at the end of the simulation is adjusted with real-time data coming from networks of sensors and stands as basis for the next short-term simulation.

For the street network of Upper Austria, and for real-life scenarios, the efficiency of SUMO decreased below real-time. We described in section 4.4 our attempt at parallelizing SUMO. While imperfect and, in fact, not efficient enough, our parallel-SUMO prototype showed that the multithreaded approach yields significant speed-improvements.

Besides the short-term simulation, we intend to use SUMO for a *mid*- to *long-term prognosis*, in which the next 30 minutes, or more, are simulated. This prognosis will be used to study the effect of various changes in the street network, coming from e.g. accidents, road works, or different speed limits.

To improve the quality of the prognosis, the existing sensor network should be further expanded, which should lead to a faster recognition of traffic issues. The generated real-time traffic condition should be the base for a dynamic traffic information system and adaptive traffic control systems.

4.7 References

[1] M. Behrisch and L. Bieker and J. Erdmann and D. Krajzewicz, SUMO - Simulation of Urban MObility: An Overview, Barcelona, Spain, SIMUL 2011, 2011, 63-68

[2] 2. http://www.gip.gv.at (10. Jan. 2013)

[3] Highway Capacity Manual, Transportation Research Board, National Research Council 3. ed. 1985, 5. [updated], 1994.

5 Benchmarking SUMO Generated Traffic Simulation Results Based on GEH Statistic

Carsten Dalaff, Rüdiger Ebendt, Jakob Erdmann, Gaby Gurczik, Louis Calvin Touko Tcheumadjeu;
German Aerospace Center (DLR), Germany

5.1 Abstract

Within the project VABENE a decision support system for public transport management authorities to estimate the impact of different traffic management strategies is developed. To identify critical traffic states on an early stage, the traffic situation is monitored continuously, and a forecast of the development of the traffic situation in the upcoming 30 minutes for a big city area is integrated. Both are based on the traffic simulation SUMO - Simulation of Urban Mobility, a microscopic, inter- and multi-modal, space-continuous and time-discrete traffic flow simulation platform. This paper aims to present the quality assessment of the traffic simulation results by using the GEH statistic.

Keywords: SUMO, Traffic Simulation, Benchmarking, Error Measures, GEH, Quality Assessment

5.2 Introduction and Motivation

Managing the traffic caused by big events, natural disasters and large traffic disasters is one of the transportation research topics of the German Aerospace Center (DLR) since several years. In this scope the main focus is on the reduction of traffic jams, problems inside the transport systems and maintaining of daily life. Within the project VABENE [1] the DLR Institute of Transportation Systems is developing a decision support system for public transport management authorities to estimate the impact of different traffic management strategies. One topic of this project is the simulation of traffic in big cities. As one demonstration area, Munich was chosen because in this area the availability for the needed infrastructure and traffic data is very good. The VABENE system visualises the current traffic state of the whole traffic network. To help traffic managers to identify a critical traffic state on an early stage, a prediction of the development of the traffic situation in the upcoming 30 minutes is integrated. This traffic forecast is based on the traffic simulation SUMO - Simulation of Urban MObility [2]. SUMO is a microscopic, inter- and multi-modal, space-continuous and time-discrete traffic flow simulation platform.

To simulate the traffic of a large region such as Munich at multiple real-time speeds, a mesoscopic traffic model was implemented in SUMO. The simulations described in this work were undertaken with SUMO versions 0.15.0 and 0.16.0, in combination with a DLR-internal extension called mesosim. This extension replaces the time discrete micro-simulation with a queue-based event-driven simulation. The dynamics are based on Eissfeldt [11].

The simulation is still agent-based in the sense that individual vehicles are tracked along road segments. However, the dynamics of car movements are modelled more coarsely and thus allow for faster computation of large scenarios.

But what about the accuracy of the simulated traffic? How good does the simulation represent the reality? How reliable is the traffic simulation? In this paper we will give an answer to these questions.

In the first step we focused on the research of available indicators, methods and tools for the benchmarking of simulation results. Based on the research results, we added a simulation quality monitoring tool to SUMO. This tool was practically tested with the VABENE simulation for the area of Munich during the time of the Oktoberfest 2011 with traffic data sets from 26.09.2011 until 10.10.2011. Analysing and discussing these first outcomes led us to the conclusion that there must be some inconsistencies within the input data, the used digital network, or the simulation. After the adaptation of the simulation a second test run was done. This paper will give an overview of the evaluation of the simulation by using the GEH statistic.

5.3 Basics on Benchmarking Simulation Results

The calibration and validation of a simulation tool is the process of comparing the simulation results for (a) chosen variable(s) with a set of observations of this (these) variable(s). The appropriate comparison scale is the goodness-of-fit measure (GoF). To determine the goodness-of-fit, various measures can be used. A detailed list of well-established goodness-of-fit-measures which are used for both calibration and validation of traffic flow models including formulations and indications to their suitability is given in [3]. As described and analysed in several experiments by [3], amongst those measures which are considered to be suitable for model calibration, there is no sufficient evidence to prefer one of them.

The practical application of the goodness-of-fit measures differs from theoretical considerations to some extent. Global standards of required accuracy ranges are missing since different purposes have miscellaneous requirements and therefore different desirable accuracies. In addition, the accessible accuracy might be higher in the case of strongly spatially and temporally aggregated values as it would be on a fine-grained level. In practice, however, experience based specifications are often used for benchmarking the performance accuracy. Experience-based accuracy definitions do exist in Germany e.g. at multilevel highway crossings where capacity deviations between simulated and observed measurements should have an inaccuracy less than 5 percent for single lanes. For large networks with several mashes and alternative routes a deviation of less than 15 percent should be obtained in at least 85 percent of the simulated traffic flow values (also adaptable to travel times). When simulating a carriageway with more than one lane, the traffic distribution on each lane can be adjusted by using RMS with an inaccuracy of less than 5 percent [4].

According to [3] more attention should be paid on how to use GoF measures to verify the calibration results and on how to give comparable quantities for acceptable deviations. A GoF measure which has proven useful in comparing simulation results against the true data is the Geoffrey E. Havers (GEH) indicator which will be described more detailed in section 5.4.

5.4 Performance Measurement Using GEH Statistic

An indicator with proven usability for benchmarking simulation results is the so called Geoffrey E. Havers (GEH) indicator [5]. With the help of the GEH indicator an overall simulation performance can be reflected. The major benefit of the GEH is his self-scaling character which makes him a perfect base for the comparison of a pair of values with different scale. In contrast to relative and absolute error measures where for instance diverse traffic volumes cause varying ratings, the GEH gives similar values in different domains [12].

The GEH statistic is usually not evaluated over a series of data as all the other goodness-of-fit measures. Instead it is applied to a single pair of observed and simulated measurements. Its mathematical formulation appears to be similar to the chi-squared test but in fact it is not a true statistical test. Rather, its application has been proven useful due to empirical tests. For individual traffic flows the GEH can be determined by using the following formula where x and y represent, respectively, the simulated hourly traffic volume of the traffic model and the true hourly traffic count:

$$GEH_i(x, y) = \sqrt{\frac{2(x_i - y_i)^2}{x_i + y_i}}$$

x_i = simulated hourly traffic volume

y_i = true hourly traffic count

The magnitude of the absolute error may be misleading, if the values of x (and y) are large. While the relative error accounts for this, giving a relative error can be problematic for small values of x. The GEH statistic can be seen as a good compromise, often avoiding the aforementioned disadvantages of giving absolute or relative errors. For a visual comparison of the GEH statistic to the absolute and the relative error, see Figure 5-1.

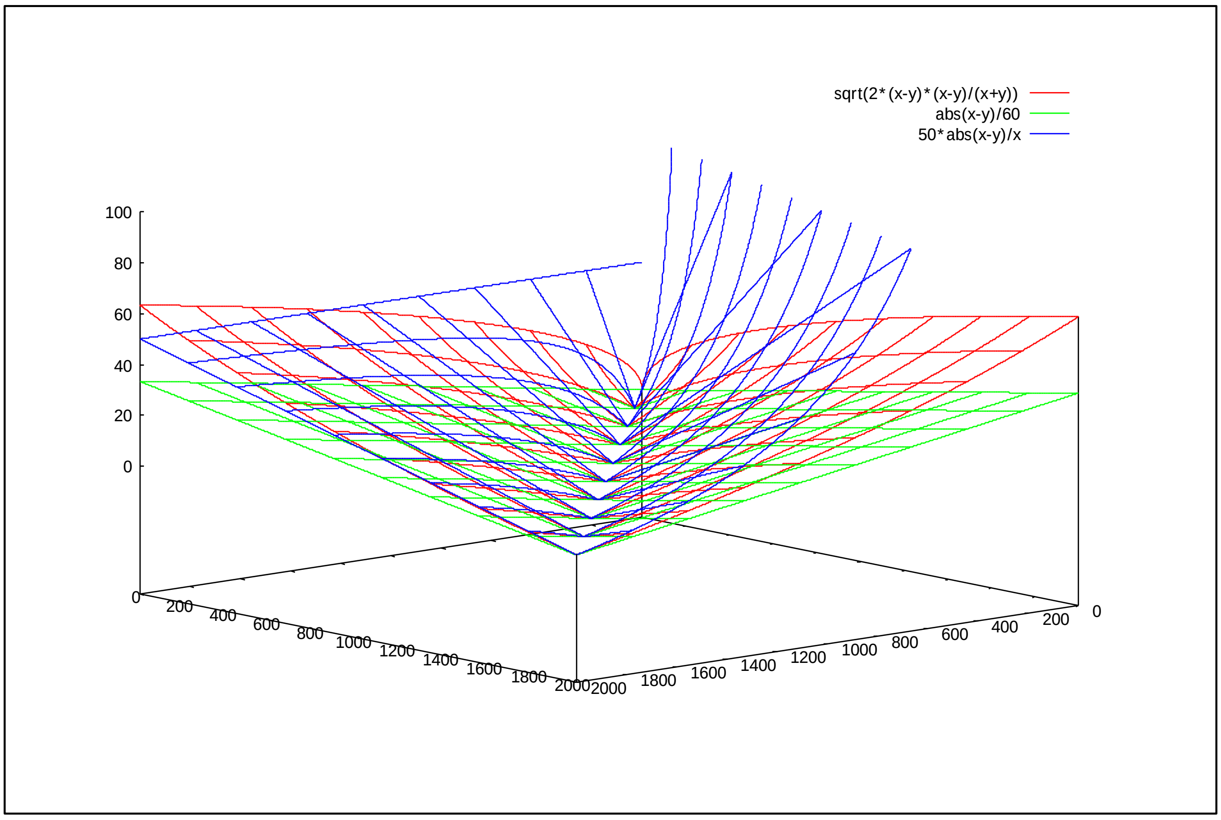


Figure 5-1: Comparison of GEH statistic to absolute and relative error

For example in [10], the following guidelines are given, based on the large corpus of respective empirical results: for any location that is relevant to the study goals, the GEH needs to be less than 5 to indicate that flows can be considered to be consistent with each other. Applied to a time series, the GEH should be less than 5 for 85% of the observed-simulated measurements. A GEH between 5 and 10 indicates that simulated flows may require further investigations (i.e., there might be possible model errors or bad data). For a GEH greater than 10, flows cannot be considered to be consistent. If the GEH is greater than 10, there is a high probability that there is a significant error either referred to the traffic demand model or the input data (see Figure 5-2).

Domain	Meaning
GEH < 5	Flows are consistent with each other
5 < GEH < 10	Descrepancies; Flows may require further investigations
10 < GEH	Flows are not consistent; High probability of significant errors in traffic demand model or input data

Figure 5-2: Meanings of the different GEH domains

The GEH can also be used counting the number of times its value is under a certain threshold [3]. This would avoid taking very small errors into account. In order to remove the dependence on the number of available observations, the number of times the GEH is under the threshold can be divided by the total number of observations. In this way the indicator would be bounded between 1 (perfect fit) and 0 (worst fit). In the following considerations only the normal GEH formula was taken into account.

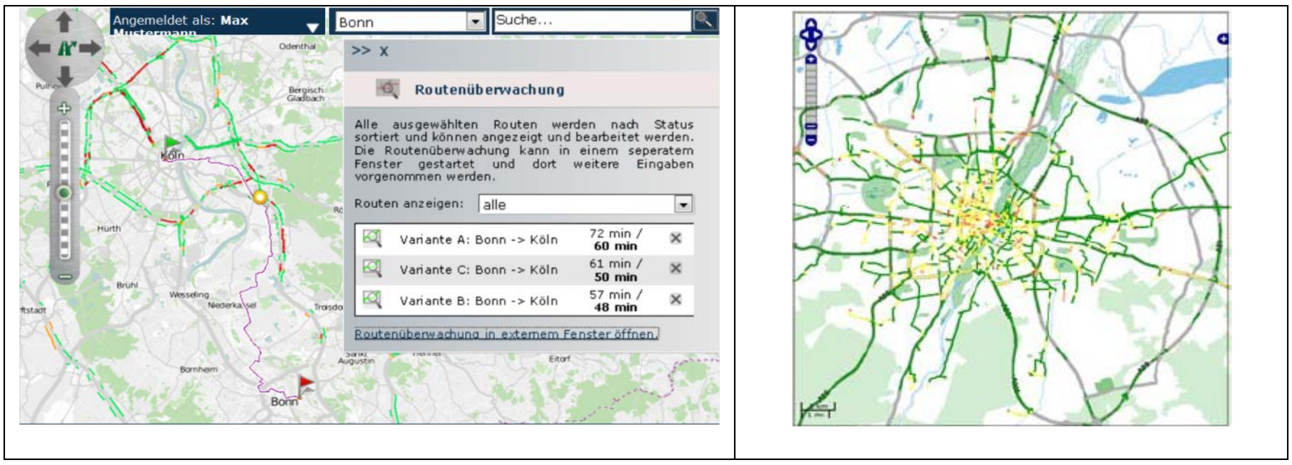
As mentioned above the GEH indicator is a self-scaling character which makes it possible to use him as mutual basis for differently scaled pairs of values. Hence, the GEH indicator is a good compromise between the usage of relative and absolute goodness-of-fit measures. An unfavorable effect in the application of the GEH is, however, that no detailed information regarding concrete sources and reasons of upcoming errors can be identified. Consequently, further analyses are required. Nevertheless, the applicability of the GEH indicator for benchmarking the overall quality of a simulation result stands out due to his ability for pointing out serious errors at a very early stage of the process. This makes the GEH an appropriate benchmark factor for validating simulation results against the true data even in the case of parallel online result processing.

Therefore, the GEH indicator was chosen for the analyses carried out in this paper. As important instruction for use in the following analyses it is specified that:

Results with a GEH less than 5 in at least 85% of all cases show a good performance. That means flows can be considered to be consistent with each other.

5.5 The VABENE Quality Tool and its Extension for Assessment of Simulation Results

The EmerT portal (“emergency mobility of rescue forces and regular traffic”) [1] is an integral part of the system for monitoring of natural disasters, mass events, and large traffic disasters, and was developed during the project VABENE. It includes a route monitoring system which continuously supervises the most important urban routes (see Figure 5-3), and is based on simulation results. For the simulation, SUMO is in continuous operation, and connected data streams (FCD and other data sources such as loop detectors) are used for a continuous online calibration of SUMO.



Alle ausgewählten Routen werden nach Status sortiert und können angezeigt und bearbeitet werden. Die Routenüberwachung kann in einem seperatem Fenster gestartet und dort weitere Eingaben vorgenommen werden.

Routen anzeigen: alle

Variante A: Bonn -> Köln	72 min / **60 min**	✕
Variante C: Bonn -> Köln	61 min / **50 min**	✕
Variante B: Bonn -> Köln	57 min / **48 min**	✕

Routenüberwachung in externem Fenster öffnen.

Figure 5-3: Monitoring of route travel times and traffic situation in VABENE [1]

To ensure persistence of quality for the connected data streams (loop detectors and FCD), a software tool has been developed which continuously assesses data quality. The framework for quality measures as outlined in [6], [7] has been used in this tool: as an example, for FCD, the tool calculates the accuracy (as the systematic error, comparing actual and calculated mean travel times) for FCD upon every hour [8]. Another example for assessment of FCD quality in this tool is a simplified "FDOT reliability" [6], which is suitable for hourly reassessments of the reliability of the travel times as computed by the route monitoring system [9]. Both approaches make use of the actual travel times as observed from the taxis of the FCD fleet. Inasmuch as both approaches can make use of larger amounts of data as the ground truth, they do not rely on the rather scarce data of e.g. field tests, and moreover, this data is also continuously available. Consequently, the chosen approach can yield meaningful statistics and operates continuously, i.e. it is not restricted to a mere offline calculation as it was the case for most of the previous approaches.

This software tool has now been extended to also assess the quality of the simulation results of SUMO. The extension is capable of assessing larger sets of simulation results offline, as well as of a continuous quality assessment. For the reasons outlined in Section 5.4, the GEH has been used as a quality benchmark.

5.6 Assessment of Simulation Results and Evaluation

In this section, the assessment steps and the evaluation of the simulation as well as the results obtained during the test campaign are described in detail.

5.6.1 The City of Munich as Test Region for the SUMO Simulation

To demonstrate the SUMO simulation and to evaluate its quality, a test campaign has been conducted in the city of Munich in Germany (see Figure 5-4) in the period from September to October during the "Oktoberfest" in year 2011 and 2012, a well-known mass event organized annually in this city, with the participation of about more than one million persons from different countries. The city of Munich as well as the time period has been chosen carefully. This was possible since DLR has exclusive access to the large amount of real time and historical traffic data from different sources like loop detectors, FCD and video data, provided by the authorities of the city of Munich. Moreover, the event was a good opportunity to test the VABENE Quality Tool, an integral part of the EmerT-portal as traffic management tool supporting decisions of first responders in case of natural disasters, mass events, and large traffic disasters.

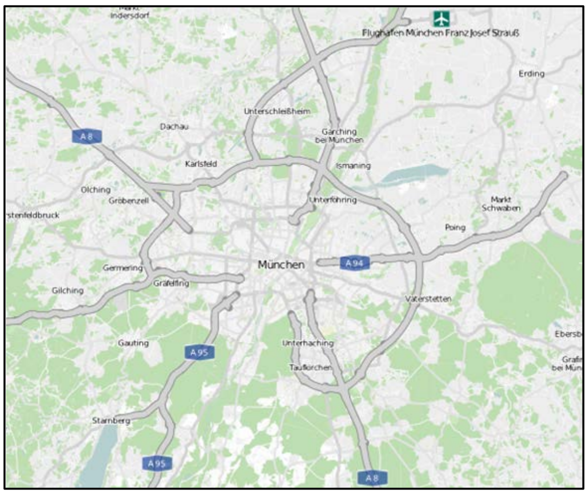


Figure 5-4: The city of Munich as test region to demonstrate the SUMO simulation [1]

5.6.2 Data Used to Evaluate the SUMO Simulation

The measured traffic data obtained from about 600 loop detectors has been used to assess the simulation results. The GEH approach described in section 5.4 has been used as quality benchmark to validate the SUMO simulation results. In this context, the measured and the hourly simulated local speeds and traffic flows have been compared.

The observation period has been chosen such that two basic scenarios have been covered: while the Munich Oktoberfest 2011 took place in the first half of the observed period, the second half comprises a period of the normal big-city traffic in Munich. More precisely, from 26.09.2011 to 03.10.2011, the traffic during a major event has been observed in the second week of the Oktoberfest, whereas in the period from 04.10.2011 to 10.10.2011, the more usual traffic in a big city has been observed.

5.6.3 Recalibration of SUMO

The discrepancy between real-time traffic measurements and simulated traffic measurements in the online-simulation are resolved by dynamically modifying the simulation state. Whenever simulated vehicle counts are in excess of real world measurements, vehicles in the vicinity of the measurement location are removed from the simulation. If the number of simulated vehicles falls below real-world measurements, additional vehicles are inserted near the measurement points. These newly inserted vehicles are assigned to a route which is representative of the route distribution at this measurement point. This is done to avoid a systematic skewing of route choices. Additionally, vehicle speeds near a measurement point are directly set to reflect real world speed measurements. Using this type of calibration a strong agreement between real-world measurements and simulated measurements is achieved at all measurement locations. This calibration increases the accuracy of predicted traffic levels when the near future is simulated. It must be noted that calibration is not possible if the simulation experiences a traffic jam which has no basis in reality. In this case, the root cause of the jam usually lies somewhere downstream of the measurement point and cannot be affected with local calibration. However, if such a "virtual" jam is detected, calibration can be used to avoid spillback.

5.6.4 Calculated GEH Values for the SUMO Simulation

The first results obtained after the GEH calculation for the 2011 version of SUMO simulation are depicted in Figure 5-5.

For 27.09.2011 and 28.09.2011, the GEH statistic has not been calculated. Most probably this is due to the fact that the used SUMO version was not entirely stable, which is the reason that the respective simulation results could not be computed. Since the used version is now outdated, it would now be difficult or at least very tedious to identify the then-reasons exactly.

As a first impression, most of the obtained GEH values indicate that the results delivered by the 2011 version of the simulation were not reliable and that the simulation needed to be improved. The possible reasons for and sources of discrepancy between real-world measurements and simulation measurements have been investigated and the respective problems have been resolved in the course of this work:

- handling of inconsistent/missing real world measurements
- The loop detector correction is now more robust
- implementation bugs (i.e. counting errors related to departed, arrived, entering and leaving vehicles) calibration can create false jams if vehicles are inserted at inopportune moments
- jams which occur locally in the simulation but not in reality may invalidate large parts of the simulation due to spillback (This situation is detected and spillback prevented through calibration) data fusion has been improved
- side effects: the overall resource consumption has been improved

Some problems still remain, in the order of importance:

- network inaccuracies (lane numbers, connectivity, traffic lights,...)
- base-demand inaccuracies
- Inaccurate vehicle dynamics

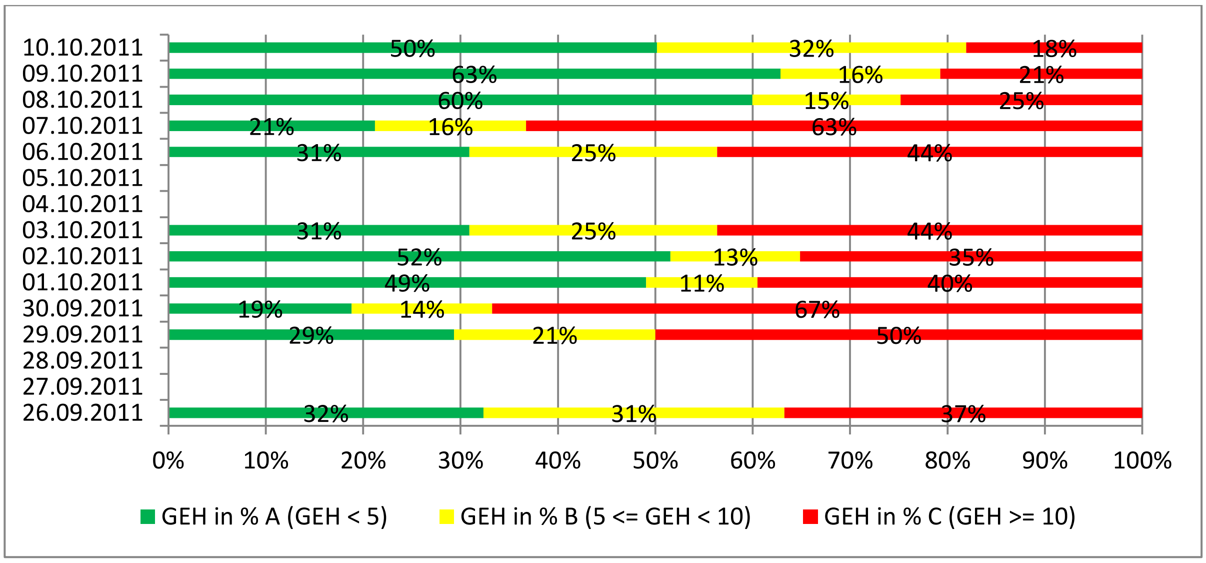


Figure 5-5: GEH results 2011 for Munich Oktoberfest

The simulation has been enhanced and a new release SUMO version 0.16.0 has been provided in 2012 with new functionalities. The GEH results obtained with the new version of

SUMO are shown in Figure 5-6. As we see, the calculated GEH values obtained for the corresponding days show much better results compared to those obtained previously in 2011.

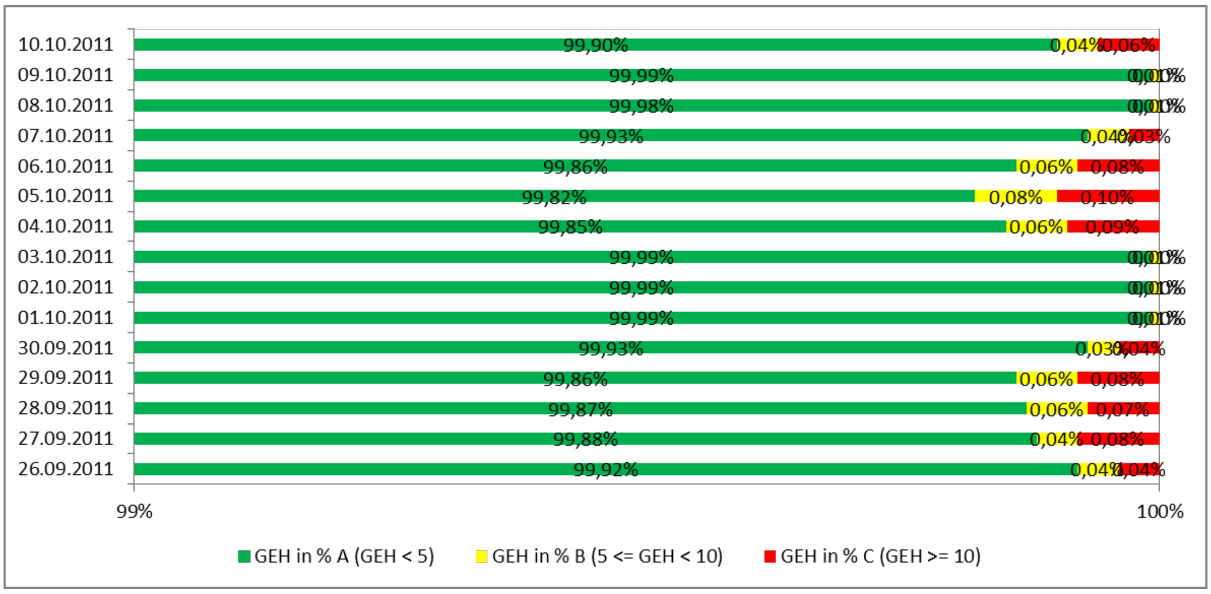


Figure 5-6: GEH results 2011 for Munich Oktoberfest, with the enhanced version of SUMO 2012

More precisely, the calculated GEH-statistic now complies with the requirements that the percentage of GEH-statistics less than 5 should be at least 85%. In fact they have improved significantly, and with an average percentage of more than 99% of GEH statistics less than 5 we now have a very good compliance of the simulated with the real traffic data.

5.7 Conclusion and Future Prospects

We have presented a coarse overview of the quality assessment of the simulated traffic situation by using the GEH statistic. The first GEH results have shown that the output delivered by the 2011 version 0.15.0 of the simulation was not reliable. After some investigations and after fixing some bugs, the GEH results obtained with the new version of SUMO 0.16.0 are very good. This supports the view that the GEH statistic is appropriate to evaluate traffic simulation.

Moreover, it has been shown that the GEH statistic is fit for the purpose of a continuous online evaluation of the simulation. This is of particular interest if the simulation is used to constantly monitor the traffic situation of a big city. By the parallel online usage of GEH statistic during the simulation process, the overall quality of the simulation results can be assessed. Therefore, while specific error sources and causes can not be directly provided by the GEH-statistic, serious mistakes of the simulation, and of the data feeds used for a continuous recalibration of the simulation can be identified at an early stage. The presented experiences will be of help for practitioners who are using SUMO, or who are using simulation in general, especially if used for purposes which are similar to the ones which have been outlined in the present work.

Furthermore, the traffic simulation results can be considered to be in excellent fit against the measured data, which shows a good step of progress in the development of SUMO.

In a next step we will determine the GEH for Munich motorways and inner city streets separately for one hour time intervals. The edge-based visualization of the GEH will follow as well as further analysis of the simulation and the traffic prediction.

5.8 Acknowledgment

The authors would like to thank the Munich City Authorities to provide the traffic data for the test region Munich in Germany used to assess the simulation results.

5.9 References

[1] DLR VABENE Project - http://vabene.dlr.de (Last access 31.01.2013 11.27)

[2] SUMO - Simulation of Urban Mobility - http://sumo.sourceforge.net/ (Last access 31.01.2013 10:02)

[3] Ciuffo, B.; Punzo, V.: Verification of traffic micro-simulation model calibration procedures: analysis of Goodness-of-Fit measures. Proceedings of the 89th Annual Meeting of the Transportation Research Board, Washington DC, USA, 2010.

[4] Forschungsgesellschaft für Strassen- und Verkehrswesen Arbeitsgruppe Verkehrsführung und Verkehrssicherheit: Hinweise zur mikroskopischen Verkehrsflusssimulation – Grundlagen und Anwendungen. Ausgabe 2006, FGSV-Verlag, Cologne, Germany, 2006.

[5] Highway Agency: Design Manual for Roads and Bridges, Volume 12, Traffic Appraisal of Road Schemes, Section 2, Part I, Traffic Appraisal in Urban Areas. The Stationery Office, London, 1996.

[6] Federal Highway Administration, U.S. Department of Transportation: Traffic Data Quality Measurement, Final Report, USA, 2004.

[7] Turner, S.: Defining and Measuring Traffic Data Quality: White Paper on Recommended Approaches, Traffic Data Quality Workshop, 2002

[8] Kuhns, G.; Ebendt, R.; Wagner, P.; Sohr, A.; Brockfeld, E.: Self Evaluation of Floating Car Data based on Travel Times from actual Vehicle Trajectories. IEEE Forum on Integrated and Sustainable Transportation Systems, Vienna, Austria, 2011.

[9] Ebendt, R.: Measuring Travel Time Reliability for an FCD-based Route Information System, Proceedings 6th International Symposium "Networks for Mobility", Stuttgart, Germany, 2012.

[10] Federal Highway Administration, U.S. Department of Transportation: Analysis Toolbox Volume III: Guidelines for Applying Traffic Micro-simulation Modeling Software.", July 2004.

[11] Eissfeldt, N. G.: Vehicle-based modelling of traffic. Theory and application to environmental impact modelling. PhD thesis, Universität zu Köln, 2004.

[12] Compendium of the Summer School Verkehrsbeeinflussung in Netzen 31. August – 03. September 2010, Mühltal bei Darmstadt, Organisator: Prof. Dr.-Ing. Manfred Boltze, TU Darmstadt, Prof. Dr.-Ing. Fritz Busch, TU München, Prof. Dr.-Ing. Bernhard Friedrich, TU Braunschweig, Prof. Dr.-Ing. Markus Friedrich, Universität Stuttgart, Chapter 4, Pages 30 – 43

6 Real-time simulations based on live detector data – Experiences of using SUMO in a Traffic Management System

Mario Krumnow, Andreas Kretschmer;
Dresden University of Technology, Chair of Traffic Control Systems and Process Automation, Germany

6.1 Abstract

An accurate real-time simulation of traffic behavior requires a large amount of very specific data. It seems obvious that the use of currently measured data from the field is a great opportunity to lead a simulation as close to reality as possible. Once a realistic simulation of the traffic behavior is available, many applications are imaginable. It is possible to simulate different scenarios to support the decision making process. Forecasts of the impacts of different management strategies can be analyzed fast and easily to help improving the quality of urban traffic. Furthermore a prediction of future traffic states can be made using currently measured data of the traffic situation.

Keywords: Microscopic Traffic Simulation, Traffic Management, VAMOS, Dresden

6.2 Motivation

In the city of Dresden the regularly operated Traffic Management System VAMOS combines data from various traffic data sources, which provide information about the past and current traffic situation. [2] Data from sensors and actuators can be used to feed and calibrate the micro simulation. Most of the data is collected by induction loops, which measure the traffic volume as well as the composition of traffic. Further sources for collecting traffic data are floating cars, which provide information about their own position and velocity. Recently gathered online data from actuators like traffic lights, variable message signs and variable traffic control units may also be used to feed the simulation. In consideration of the mentioned online traffic-data simulation tools like SUMO can be used to determine traffic situations for several parts of a traffic system, dedicated for different modes of transport (such as private traffic or public transport) as well as for the whole overall transportation system. Altogether simulation assisted calculation might ease or even enable the estimation of traffic states for complex transport networks and at least help to improve forecast quality.

6.3 Data sources

Data from various sources like detectors or actors are collected and processed by the basic services of the management system VAMOS and finally stored in several relational databases. For some detectors even a continuous data stream is available, e. g. cameras or automatic traffic counters. The next section will give a small overview about the traffic data sources and how to use them in a micro simulation.

6.3.1 Traffic volumes

For the whole city of Dresden a up-to-date traffic volume map is available which represents information about the daily traffic volume and the percentage of heavy traffic (Figure 6-1). The traffic volume map provides no further information about the classification of the traffic volume into specific classes like cars, motorbikes or trucks. Though, that information is important for the calibration of the simulation and for reliable statements concerning the emission and the average speed in the SUMO network model.

Moreover, the traffic volume map contains no information about the distribution of traffic at intersections. For that reason the traffic surveys are carried out periodically to determine these distribution values. The information about the current traffic load is gathered by automatic traffic counters which can also be used to get knowledge about the traffic distribution.

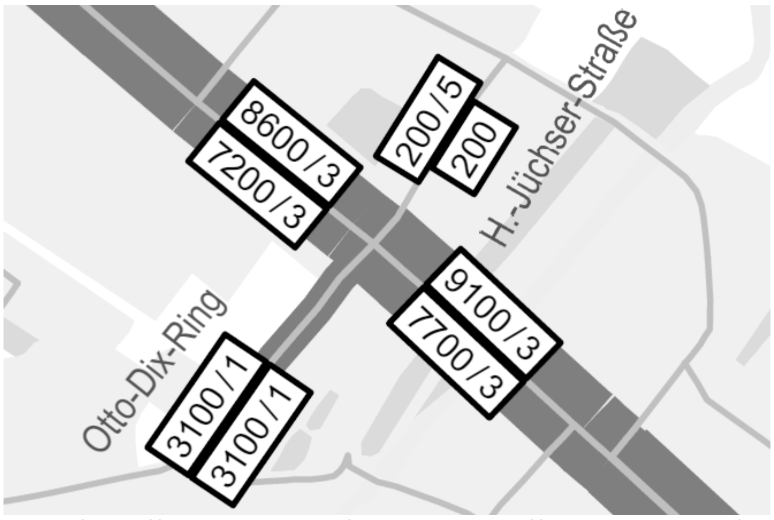


Figure 6-1: Part of a traffic volume map (format: daily traffic / percentage of heavy traffic)

6.3.2 Traffic lights (TLS)

To model the behavior of the traffic light control it is necessary to correctly reproduce the TLS program. Due to the fact that traffic light signalling differs in kind of control – especially regarding the degree of dependency on traffic situation and data – several approaches covering different cases have to be considered.

The simplest case is a fixed time control. Here the main task is to define the switching times for each signal group. This is typically done in TLS cycles where the initial timestamp of the first cycle has to be valid so that there is no offset to the TLS outside in the field. The more sophisticated case is the traffic actuated TLS control whereas dynamic cycle times and traffic influenced green phases exist. Besides the dependency on data about private traffic, controlling is particularly affected by input from and about public transport and has therefore to be considered for reproduction. For example, in Dresden more than 80 % of the TLS have the opportunity to interact with the public transport. At the moment the implementation of traffic actuated signal controls in SUMO is a complex and very time consuming process. The whole TLS logic has to be implemented in SUMO because tools to import native TLS programs, e. g. VS-PLUS , are not yet available.

One approach to integrate the input values for the TLS control is the usage of induction loops within the simulation. The data is then pushed to the TLS software control influencing the simulated TLS. Another approach is to build up a hardware-in-the-loop model. In this case an existing TLS hardware controller would be needed to put the data from the simulation to the controller input unit. The hardware controller generates signals for the TLS controlling which are pushed back from the controller output unit to the simulation.

Both variants are not useful for huge networks, because all current TLS algorithms have to be implemented either as part of the simulation or as specific model. Even the detailed maps including detector locations have to be reviewed and to be integrated into the simulation. Whilst execution of a simulation each single detector measuring value has to be transferred in real time to be considered within this simulation cycle. That means that various kinds of detectors like induction loops, cameras, traffic eye universals etc. have to be imported and calibrated correctly. A decision has to be generated whether the detector works properly and no systematic failures (like interchanged channel numbers for single induction loops) exist. Altogether, both approaches might lead to highly accurate simulation, but are nevertheless not or hardly affordable.

Highest priority was given to the simulation of a traffic network rather than one more exact copy of one single TLS controller. Thus, and because of the expected affords for the both approaches described above, another opportunity was taken into account. The third approach is to capture only the exact signal states of the TLS controller in the field and push them to the TLS in the simulation to override the state of the default traffic signal control. In comparison to the previously described approaches the third one demands a trustworthy simulation. This means the traffic volume and all the physical parameters have to match precisely with the reality which is hardly to achieve. In order to forecast the future signal states some algorithms for traffic actuated controllers have been developed [9]. This information will also be pushed to the TLS control in the simulation to get some information for some decision making processes.

6.3.3 Floating car data (FCD)

In Dresden a fleet of 500 taxis collects floating car data. Position, velocity and an occupied flag are recorded every 5 seconds. These data are used to determine the level of service (LOS) for all parts of the entire traffic network. Several algorithms integrated in VAMOS verify the generated LOS messages by comparing the FCD message with other traffic detectors like automatic traffic counters [2].

Moreover, it is possible to generate highly accurate maps of the entire traffic network. This depends on the accuracy of the GPS location and the frequency of the GPS tracker signal. In figure 6-2 the GPS location points are connected so that a map becomes visible.

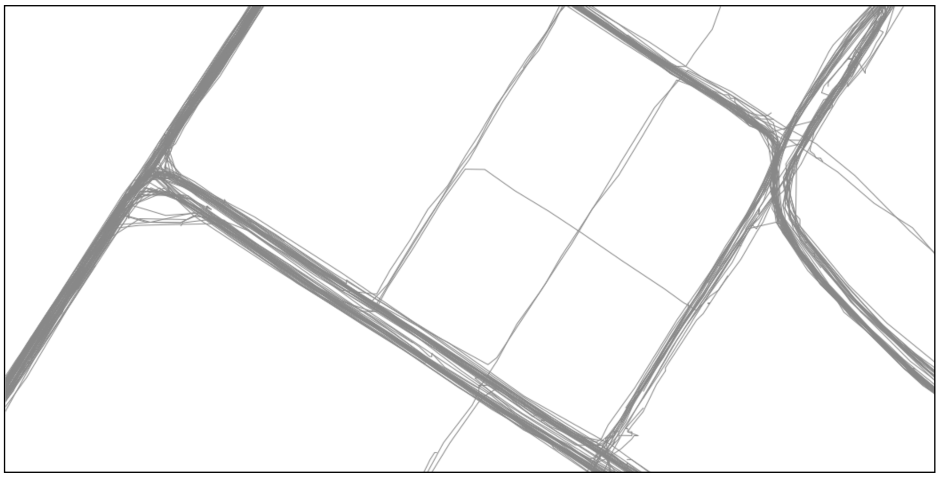


Figure 6-2: Generated map based on floating car data)

These taxi FCD data can be used to validate the simulation findings. In cases of inaccuracy within the simulated scenario the velocity can be adjusted to a lower or even a higher value. Consequently vehicles on a specific link will be removed to dissolve traffic jams. Additionally the FCD data are suitable to decide whether the simulated traffic works well or not.

6.3.4 Public transport data

In Dresden most of the TLS are influenced by public transport. In order to reproduce a realistic behaviour of the public transport vehicles routes for busses and trams are specified. Subways or light rails were not modeled because they do not interact with the individual traffic on the roads.

The simplest approach is to use a fixed time schedule (accurate to a minute) for public transport representing realistic traffic behaviour in a rough manner only. For higher accuracy more information is needed, initially static information as the planned arrival time (accurate to a second), detailed information about the speed profile of the vehicles or the speed restrictions in front of curves or at tram switches.

Also information about the current location in association to the positions according to the time schedule and the dwell time at the stops are necessary. For that purpose the data of a special measuring tram which is part of the cooperation between public transport operators and the Dresden University of Technology can be accessed. Additionally information results from corresponding probes gathered for bus and again tram lines with a mobile GPS logger. In summary, the first approach seems to be very rough, but allows gaining information for a whole network with less afford. The comparison to data of the additional information sources mentioned above results in good average quality and accuracy for public transport in Dresden. However, users have to consider that quality gathered according to the method of this first approach may depend on the operational and quality management of the public transport aiming to keep close to their time schedule. In case of special situations and therefore as requirement for high quality traffic and transport management this approach might not be suitable without restrictions.

According to the tasks in traffic management (e. g. management in case of large-scale events and in case of incidents relevant to the transport network) and as a base for high quality management the second approach deals with more situation depending data. As additional input real time data from the local transport company is available. This data is generated by an intermodal transport control system (ITCS) which is based on an analog radio system. By using a pull mechanism the position of each bus and tram is captured every 15 seconds.

With this data more accurate information about the arrival and leaving times at the stops is available. Furthermore incidents or the location of transfer synchronization points can be identified. In a first implementation properties of both approaches have been considered and integrated: Trams as well as busses are put into the simulation at a specific time but already based on the information on the telegrams of the ITCS.

In future work the position of the public transport will be also controlled for the whole route by moving or teleporting forward or backward on the route. Also the routes of the public transport should be able to be adapted because of special events like returning to the depot or because of incidents in the network.

6.3.5 Parking data

The information about the current load of parking lots in the city is available. The management system also predicts a trend of the load for the next couple of minutes. At the moment this data is not used within the simulation. Nevertheless this kind of data might be used in the future to simulate a more realistic driver behaviour which is based on dynamic data.

6.3.6 Camera data

A lot of cameras distributed all over the city can be accessed by the management system. Most of the cameras are installed at the traffic lights to work as an optical detector for the TLS control. The video data will be used to validate the simulation results. Due to the lack of tools for automated integration and as the process for calibration of the image recognition is very time consuming, this validation is done in a manual way as a first step.

In the future important image recognition algorithms for cameras exclusively driven for traffic management might be calibrated for example to compare the current queue length with the queue length in the simulation.

6.4 Real time simulation setup

6.4.1 Building the network

SUMO has already a tool to import OpenStreetMap (OSM) data, which is not suitable for the road network of Dresden. Reasons are the low level of detail within the generated SUMO network, incorrect numbers of lanes per road section and incomplete information about primary and secondary roads. Finally, the existing net model of the traffic management system VAMOS was used as the base for a SUMO network. That model is mapped in a MySQL database where the geometries of all elements are stored in different tables. The VAMOS network model includes information about the geometry of each lane, the edges and the intersections. Also the connections between the lanes are available.

The task was to build SUMO shapes from the VAMOS geometries and import the information about the connections between the edges. Because of the XML exchange format of the SUMO net file, this task can be reduced to a simple conversion between both data sources.

6.4.2 Access to the simulation

To handle and simplify the integration of several and different data sources a universal interface is needed which can be accessed by several applications like the modules for handling the detector data. Another requirement is that the simulation should act as a service which means that the simulation will not terminate to a specific time. A suitable solution is a web service that interacts with SUMO and handles all connections to the several clients. This service (TraCI as a service – TraaS) was implemented in Java and will become OpenSource very soon . The Traas web service uses the well known Simple object access protocol (SOAP) instead of the TraCI byte code and is based on a TCP/IP connection.

In Figure 6-3 the concept of the Web Service is shown. On the left side all input values are pushed to the service and will then be forwarded to different instances of SUMO.

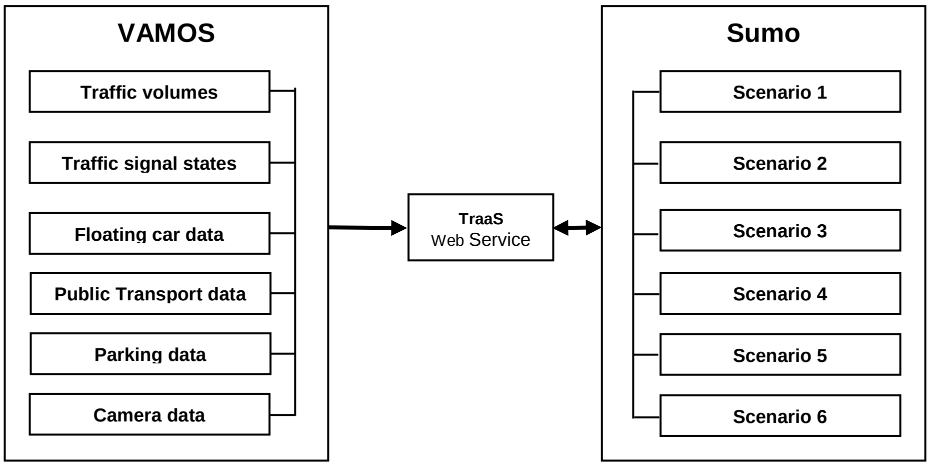


Figure 6-3: System setup to interact with different instances of SUMO

6.5 First Results

In a first realistic simulation an arterial of Dresden including 15 traffic signals with individual traffic as well as public transport vehicles was modelled. By the help of this model the advantages of the real-time I/O data interface are demonstrated [6]. Other works [7] dealt with the simulation of a single intersection, concentrating on the emission output of vehicles depending on the traffic signal program. As measuring emissions in practice is a real challenge, it is very helpful to have a simulation-based assessment in order to choose the best suiting signal program to reduce these emissions. An analysis of congestions in front of intersections, depending on the traffic signal program, has also been carried out. Figure 6-4 shows, that video images and snapshots from the simulation can be compared to verify congestions at an intersection.

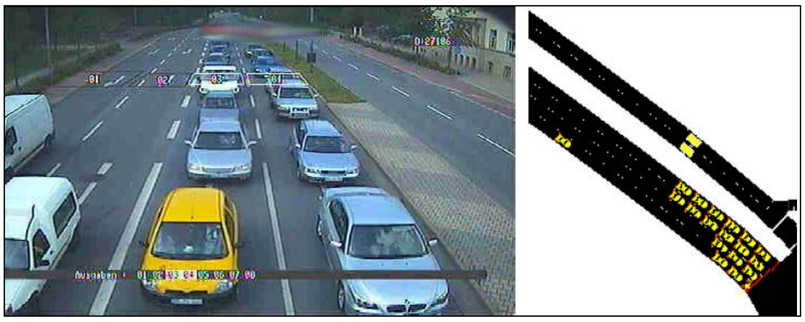


Figure 6-4: Comparison between video image and simulation snapshot

Most of the gathered test series seem to have sufficient quality in representing the real traffic situations and passed cursory plausibility tests. In Figure 6-5 a comparison of the simulated velocity and the real velocity of a taxi is shown. The route has a length of approximately 9 km and various traffic adapted signal lights. At two discrete locations the traffic volume is measured with automatic traffic counters. The behaviour of the taxi driver in adapting the velocity is smoother. The opportunity to choose the maximum acceleration or deceleration of the vehicle is not used so often in the real taxi because of the forward-thinking behaviour of the driver. This driving style is also more comfortable for the passengers.

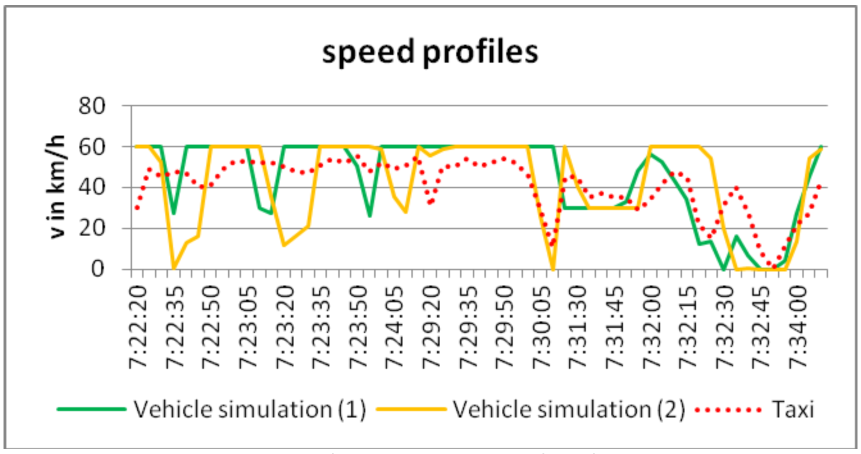


Figure 6-5: Comparison between simulated and real journey times

6.6 Conclusions

With the approaches described a first implementation of a real-time simulation based on live detector data has been developed and tested. The developed real-time simulations of specific traffic situations can be processed with SUMO very fast and offer the possibility for many output options. This various output data can be used as a base for extensive subsequent analytical processes. In order to predict the reaction of traffic participants on the applied strategy, many simulations can be run in a parallel mode.

According to the positive first results and the benefits seen in additional simulations one of the next steps should be the thoroughly analysis of data quality which can be achieved by using SUMO real time simulation. For further use in daily operation users and developers should be aware that data quality can be assumed as sufficient only while measuring and passing comparison to detailed data of the real traffic network. For that reason additional future tasks would be the definition of an initial calibration process and for the ongoing quality management based on supporting points.

Using live detector data in SUMO at least offers numerous possibilities for traffic analysis and, by the help of comparison, for the further development of the simulation software itself. A further main aspect is the supported replication of the current traffic situations by the simulation. Starting from that view it is particularly interesting to compare the progresses within the real traffic network with the one in the simulation environment. With the implementation and testing of real time simulation an essential pre-condition for the application SUMO for a variety of new tasks and especially for its integration into short term decision making processes has been carried out.

In particular, the potential for testing traffic management measures before applying them to the field actuators and mainly the opportunity to analyze more than one single traffic measure or more than one single strategy simultaneously or almost in real time let expect new and important findings.

Currently Research engineers and students of the Dresden University of Technology are working constantly on the improvement of that kind of real-time simulations to ensure the quality and represent the reality as accurate as necessary.

6.7 References

[1] SUMO – Simulation of Urban Mobility (2012).Institute of Transportation Systems, German Aerospace Center, Germany. http://sumo.sf.net. Accessed March. 27, 2013.

[2] VAMOS - Verkehrs-Analyse-, Management- und Optimierungs-System (2012).Chair of Traffic Control Systems and Process Automation, Dresden University of Technology, Germany. http://www.vamosportal.de. Accessed March. 27, 2013.

[3] Microscopic real-time simulation of Dresden using data from the traffic management system VAMOS, 19th ITS World Congress Vienna, Austria, Oct.25, 2012

[4] http://www.dresden.de/media/pdf/mobilitaet/VMK.pdf

[5] Arlt, A. (2012). Realitätsnahe Simulation des Verkehrsflusses auf der Süd-West-Umfahrung in Dresden mit SUMO unter Berücksichtigung des MIV und des ÖPNV (in German). Report student research project, Dresden University of Technology, Germany.

[6] Reiche, M. (2012). Vorher-Nachher-Analyse der Emissionsbelastung am Knotenpunkt Nürnberger Platz im Rahmen der ÖPNV-Bevorrechtigung des NSV-Projektes (in German). Report student research project, Dresden University of Technology, Germany.

[7] Wießner, E (2013). Analyse der Datenmengen der Car-2-Infrastructure Kommunikation durch realitätsnahe Simulation unterschiedlicher Testszenarien (in German). Report student research project, Dresden University of Technology, Germany.

[8] Krumnow, M (2012). Schaltzeitprognose verkehrsadaptiver Lichtsignalanlagen im Rahmen des Projektes EFA 2014/2, 8. VIMOS Tagung, Dresden

7 Predicting Travel Time of a Vehicle in Occluded Area with SUMO

Wei Li, Claudia Sánta and Matthew Fullerton;
Technical University of Munich, Germany
Dominik Rosenbaum;
German Aerospace Center (DLR), Germany

7.1 Abstract

The work aims to develop robust continuously vehicle-tracking algorithms with aerial images. It concentrates on predicting travel time of a vehicle travelling in occluded areas, such as tunnels and roads under bridges. Microscopic traffic simulation is used to estimate the movement of vehicles on roads, in the case that available information is limited or a large amount of factors are involved, such as intersections with traffic lights and accidents. According to preprocessed data from aerial images, roads, flows and traffic signals are defined and calibrated. Ultimately, the travel time exported from the simulation is regarded as the time to guide a glider to track the vehicle at the end of the occluded area. With outputs of the simulation, the queue length and speed entering a tunnel is compared and assessed by detecting and tracking data from aerial images, which are taken above Munich by 3K cameras on a glider of DLR.

Keywords: vehicle-tracking algorithms, microscopic simulation, SUMO, travel time in tunnels, Altstadtringtunnel in Munich, 3K cameras, aerial images, detecting and tracking data

7.2 Introduction

The work is carried out by the Department of Photogrammetry and Image Analysis in German Aerospace Center (DLR) under the project Chicago, which focuses on tracking vehicles with aerial images [7]. The work concentrates mainly on predicting travel time of a vehicle passing through occluded areas, such as tunnels and roads under bridges, since images taken by cameras in a glider cannot monitor positions of vehicles there. The predicted travel time provides guidance-information for a glider to track a vehicle continuously.

As one of the predictive approaches, microscopic traffic simulation is used to estimate the movement of vehicles on roads, in the case that available information is limited or a large amount of factors are involved, such as intersections with traffic light and accidents. The other approaches are statistical analysis and the cell transmission model [2][3]. For the microscopic traffic simulation, flows and traffic control methods are defined and calibrated according to preprocessed data from aerial images with SUMO [1]. The availability of databases for networks and traffic-control methods determines the feasibility of the microsimulation for tracking vehicles continuously.

The elementary traffic information coming from the images are taken above Munich by 3K cameras on a glider of the DLR [6]. Preprocessed data for detection and tracking from the aerial images are the basic information. Based on the preprocessed data, aggregated traffic data, such as traffic volume, speed and queue length, are derived and used to calibrate the simulation. The roads, the traffic signals and the traffic flows are defined as the basic inputs, while the travel time, the velocities entering the tunnel and the queue length before intersections are the desired outputs from the simulation. The travel time exported from the simulation is then regarded as the time to guide a glider to reach the exit of the tunnel in the research.

7.3 Inputs of Simulation

In the simulation the basic inputs of the simulation are networks and routes. In networks nodes, edges, types and connections of the edges are defined. A file of routes includes all possible paths of vehicles and traffic flow in these paths. Traffic management equipment, such as induction loops and signals are included in the simulation by way of additional files.

7.3.1 Model of Roads

In SUMO networks can be imported from various sources of maps, such as NAVTEQ and Open Street Map (OSM) [1]. However, in this work only a few roads were studied and the roads are with different numbers of lanes. Thus the roads are built in SUMO according to the UTM coordinates of junctions near the Altstadtringtunnel in the orthogonal images, which are processed from photos taken by 3K cameras in gliders [5].

The Altstadtringtunnel, which is studied for the simulation in the paper, is located in the centre of Munich, to the north of the metro-station Odeonsplatz. The nodes in the simulation contain intersections, starting points of roads, points of forks, position of the entrances and exits of the tunnel, points within curves and so on.

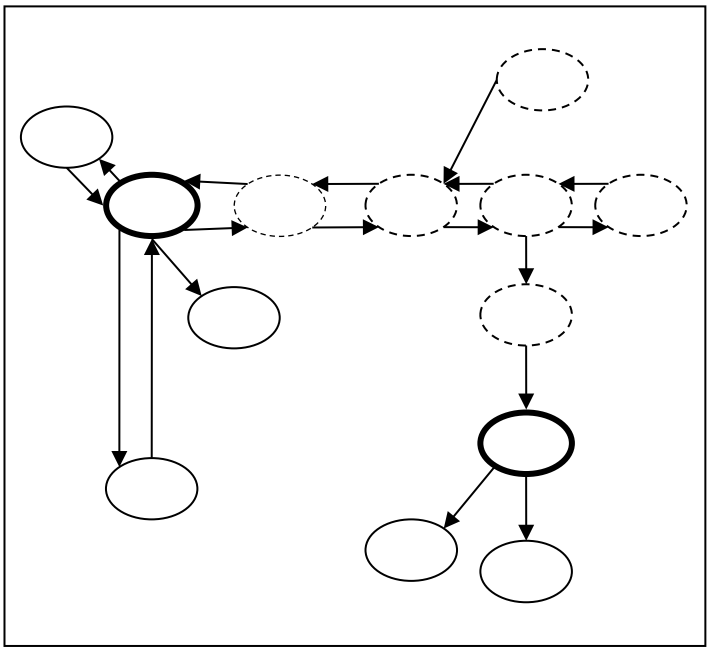


Figure 7-1: Network model of Altstadtringtunnel in Munich

The nodes and edges of the Altstadtringtunnel are illustrated in Figure 7-1, the dotted ellipses are the nodes in the tunnel and the bold ellipses are the signalized intersections. T1, T2, T3 and T4 are the ends of tunnels. T2 is an entrance only and T3 is an exit only. T0 and T33 are the signal-controlled intersections near the tunnel. M5 is the fork point, while M4 is the merging point. It should be highlighted that there is no traffic flow within nodes and U-turns are not allowed. The speed in roads with more than one lane in one direction is 75km/h, since velocity of most running vehicles in these road parts detected from aerial images is beyond

60km/h but below 70km/h. The speed limit of edges of one-lane, such as edges t0-s4, m5-t3 and t33-t34, is 50km/h or 60km/h. Except for the entrance of T2 with one lane and T4 with two lanes, the exits of T3 with one lane and T4 with two lanes, the road consists of three lanes per direction in the tunnel.

7.3.2 Routes and Traffic Flows

First of all, the vehicles are categorized into two types with the length of 5m and 10m. The short vehicle can run with a max speed of 250km/h, while the long with 200km/h. The minimum gaps of the short and long vehicle are 0.5m and 1.2m for safety in the traffic. The ratio of the two types in flows shall be 60:1 based on the observation from the aerial image sequences ON0204-14 at the Altstadtringtunnel.

One flow and a few routes are created based on a departing edge. A route is defined with a sequence of edges and an assigned weight. Then vehicles of one flow will choose a route from the departing edge randomly based on the weights of the route. Figure 7-2 shows the starting and ending points of roads around the Altstadtringtunnel. Zones A, B, C, D are the entrances and A, B, D, E, F, G are the exits of the network.

It should be noted that there was a bus near junction T33, which moved slowly for a few seconds and led to a long queue before the stop line. The bus shall be specially defined with a length of 12 meter. It moved slowly, and nearly stopped while approaching the intersection T33. The bus was inserted at the simulation time 1300s on the edge before T33. Then it stopped at the position of 15m before T33 with the duration of 135s. Some vehicles would overtake it but few might stop behind.

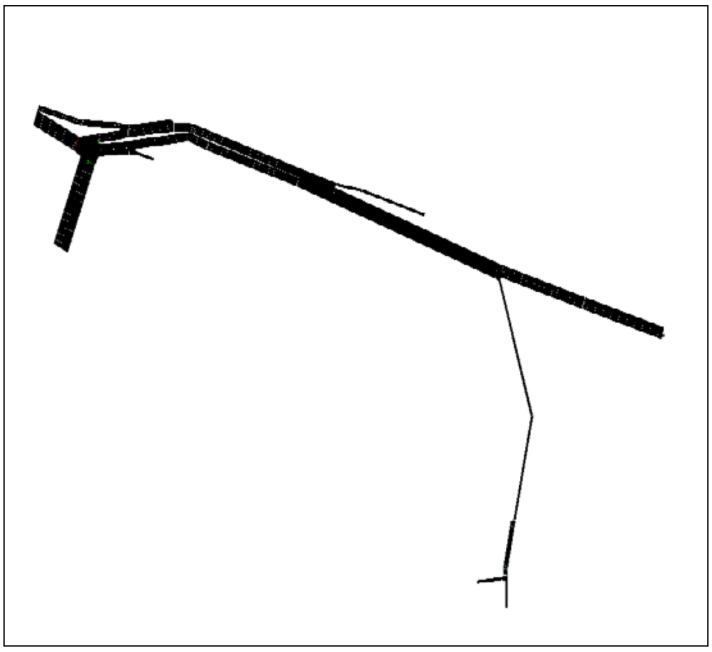


Figure 7-2: Road model of the Altstadtringtunnel in Munich

Table 7-1 shows the routes, flows and assigned weights in the simulation. The flows are calibrated according to the maximum queue length before the intersection. The lanes, in which vehicles depart to enter the roads, are defined as 'best', since there would be chaos, if one vehicle turns right but departs from the leftmost lane and another would go straight but depart from the rightmost lane.

Table 7-1: Routes and traffic flows in the study area.

From	To	Flow (veh/h)	Assigned Weights
A	B	900	0.03
	D		1.1
	E		0.05
	F		0.45
	G		0.03
B	D	1100	1.1
	E		0.05
	F		0.45
	G		0.3
C	A	800	0.4
	B		0.75
D	A	1400	0.4
	B		0.75

7.3.3 Signal Groups and Programs

The signal groups are the means of traffic control in the study. Here signal programs are fixed-time. Thus the duration and the phases shall be predefined to run the simulation.

In the junction T0 there are three phases of the signal programs. The phase I is the flow from east to south, the phase II serves the flow from south to east and the phase III serves the flow between east and west. In the simulation non-motorized traffic is not considered, since there is little effect on the traffic flow. The cycle time in the intersection is 100 seconds. The duration of each phase was measured at the intersection at 11:00 on 28th July. The Figure 7-3 illustrates the signal program in the junction T0. The cycle time is 98s in the intersection.

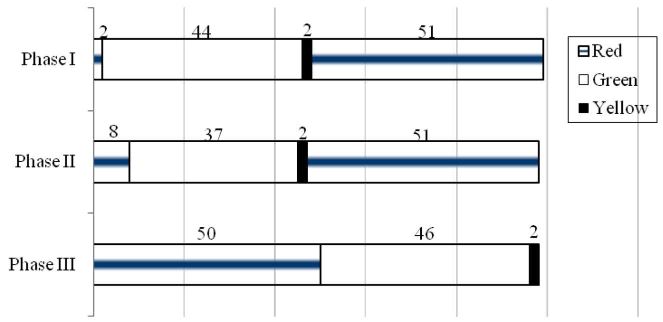


Figure 7-3: Signal Program at the intersection T0 (time unit: s)

The second intersection in the research is T33. It is the intersection after the exit of the tunnel T3 with one lane and extends to two lanes before the stop line. In the junction T33, the cycle time is 90 seconds. The Figure 7-4 shows signal program of T33.

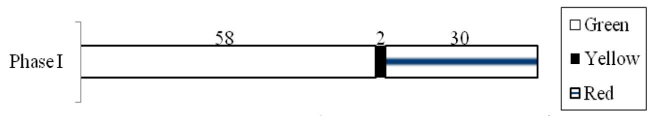


Figure 7-4: Signal Program at the intersection T33 (time unit: s)

7.3.4 Detectors

In order to export the traffic information to files, detectors are set in SUMO. Here three kinds of detectors are installed in the simulation. One is induction loops that record information of traffic flow, such as arriving time, number of vehicles detected. Another is instantaneous induction loops, which are used to get individual ID, speed, arriving, leaving and dwelling time of the vehicles. The instantaneous induction loops are set at the ends of the tunnel to calculate the travel time of vehicles in the tunnel. The last kind of detectors is lane area detection namely E2 detection. They receive information of jams, such as halting time, length of the jam with vehicles and meters and the total number of vehicles influenced in the jam. The lane area detectors cover 40m distance before the stop lines to get the information on the queue.

7.4 Error Checking and Calibration

Before processing the simulation outputs, the simulation clock should match the time series of the aerial images along the Altstadtringtunnel in the center of Munich. The matching base is the offset between the starting time of the first phase in two intersections T0 and T33. The collection and aggregation of traffic data from simulation are done from simulation clock 1431s to 1488s, accordingly aerial images from OF0175 to OF0232 above Munich on 26th April 2012.

Error checking is needed to guarantee that the calibrating process will not be affected by major errors in network and demand. Error checking focuses on three basic stages: software error-checking, input coding error checking and animation review to point out slight input errors [4]. In the research about Altstadtringtunnel the network generated in the simulation matches to the UTM-coordinates [5]. The demand is based on the observation from the aerial images, and then the flows assigned on each path and components of normal and long vehicles are written in the route-file. The inductive loops are set to confirm that the number of the generated vehicles fits the written flows in the route file.

Then the simulation needs to be calibrated to produce a reliable result. Calibration data are composed of measures of capacity, traffic counts and measures of system performance, such as speed and queue [4].

Calibrating factors in the simulation of the Altstadtringtunnel are the max queue length at the exits of the intersections T0 and T33, the velocity of vehicles entering and exiting the tunnel and the right-turning ratio in the intersection. Variables to be revised in the simulation are path choice percentages, the traffic flow and the standing time of the slow-moving bus at the exit T3 of tunnel. The comparison is according to preprocessed data from aerial images. The purpose of simulation is to get travel time in tunnels.

The first targeted calibration data of traffic performance is the queue length at a signalized intersection. The considered period of simulation is from 1430s to 1490s. The queue length of a multi-lane edge is the sum of the number of jammed vehicles on all lanes, since the number of jammed vehicles in the images is counted based on edges and the result will not be influenced by the lane-changing behavior. Table 7-2 shows the result of the calibration considering the measure of the queue length.

Table7-2: Maximum simulating and actual queue length with vehicles.

Queue length with vehicles	Max queue vehicles (simulation)	Max queue vehicles (images)

South of T0	20	21
West of T0	11	-
East of T0	17	17
Straight at T33	12	12
Right turning at T33	4	3

Besides, the speed of vehicles entering and exiting the tunnel needs to be calibrated, since the velocity is decisive for prediction of travel time. The distribution of the velocity during the period of 1430s to 1490s is analyzed statistically. Table 7-3 compares the distribution of the velocities observed in images and through simulation at the entrance T0 and the exit T4 of the tunnel. The variances of velocity entering the tunnel in images are much bigger than in simulation, while the means differ slightly. The speed distribution of the simulation at the exit T4 are displayed almost the same as in reality. However, since we are primarily interested in a general estimate of travel time through the tunnel, a close match with mean is more important than an accurate variance.

Table 7-3: Distribution of velocity entering and exiting the tunnel.

	Velocity (km/h)	Images	Simulation
Entrance T0	Max (km/h)	57.320	52.776
	Min (km/h)	42.264	48.096
	Mean (km/h)	50.028	50.583
	Standard Deviation (km/h)	6.019	1.235
Exit T4	Max (km/h)	69.732	69.948
	Min (km/h)	67.932	68.688
	Mean (km/h)	69.021	69.232
	Standard Deviation (km/h)	0.850	0.432

7.5 Estimation for Performance of Simulation

The travel time is the important result from the simulation. Instantaneous induction loops will be used to calculate the travel time within the tunnel. The time, when one vehicle passes the instantaneous detectors at the entrances and the exits of the tunnel, is recorded in the file ‘travleTime.xml’.

The distributions of the travel time of both paths are shown in Figure 7-5 and Figure 7-6. The result of travel time from T0 to T3 varies greatly, since it is influenced by the congestion at the intersection T33, when the slow-moving bus increases the queue length and the vehicles slow down while approaching the jam. The estimation of travel time of the tracked vehicle is regarded as the same as the vehicle in the simulation with the most similar velocity entering the tunnel. As the result, the glider shall be guided to reach one exit of the tunnel to track the vehicle continuously.

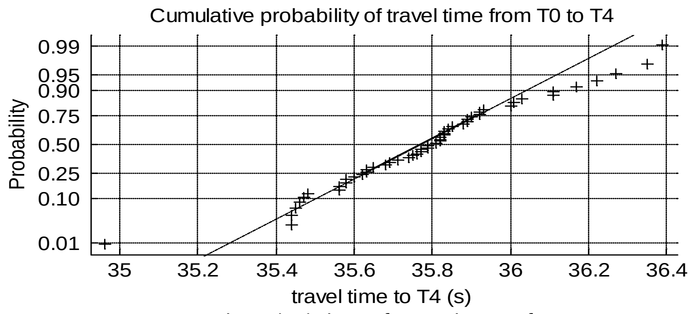


Figure 7-5: Normal probability of travel time from T1 to T4

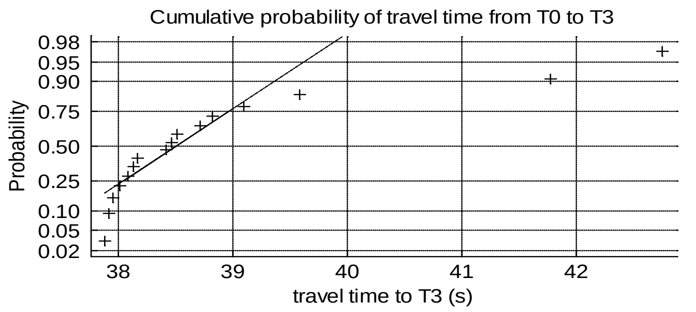


Figure 7-6: Normal probability of travel time from T1 to T4

7.6 Conclusions and Outlooks

The study has shown that microsimulation in general and SUMO in particular can be used to estimate travel times when no data from the aerial images is available. The traffic simulation solves the problem under complex situations, such as an accident in the signalized intersections. However, the database of network and the traffic-control mode, such as signal program of traffic light, are required for applying microsimulation to navigate an airplane to track the vehicle continuously.

Future work could include incorporating SUMO as an online simulation to be used directly with tracking algorithms with aerial images, so that continuous, live results can be obtained.

7.7 References

[1] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz D.: SUMO-Simulation of Urban MObility: An Overview, In: The Third International Conference on Advances in System Simulation, Berlin, pp. 63--68(2011)

[2] Daganzo, C.F.: The cell transmission model: a dynamic representation of highway traffic consistent with the hydrodynamic theory. In: Transportation Research: part B, vol. 28B, no. 4, August, pp. 269--287(1994)

[3] Daganzo, C.F.: The lagged cell-transmission model. In: Proceedings of the fourteenth International Symposium on Transportation and Traffic Theory, Jerusalem, Israel, pp147-171(1999)

[4] Dowling, R., Skabardonis, A., Alexiadis, V.: Traffic analysis toolbox volume III: guidelines for applying traffic microsimulation modeling software, FHWAHRT-04-040. Federal Highway Administration, Washington, D.C. (2004)

[5] German Aerospace Center (DLR) 2012, Remote Sensing Technology Institute, Photogrammetry and Image Analysis, viewed 1 Oct 2012, http://www.dlr.de/eoc/en/desktopdefault.aspx/tabid-5431/9230_read-17834/

[6] Kurz, F., Tuermer, S., Meynberg, O., Rosenbaum, D., Runge, H., Reinartz, P., Leitloff, J.: Low-cost optical Camera Systems for real-time Mapping Applications. In: Photogrametrie Fernerkundung Geoinformation, Stuttgart, vol. 2, April, pp. 159--176(2012)

[7] Rosenbaum, D., Leitloff, J., Kurz, F., Meynberg, O., Reize, T.: Real-time image processing for road traffic data extractiopn from aerial images. In: Commission VII Symposium (2010)

8 3D visualization for microscopic traffic data sources

Andreas Wenger;
Xenoage Software, Germany
Matthew Fullerton, Mathias Baur,Silja Hoffmann, Jonas Lüßmann;
Chair of Traffic Engineering and Control, Technische Universität München, Germany

8.1 Abstract

The typical approach to realistic 3D visualization of microscopic traffic data is usually location-specific and hence requires a lot of effort from the user to generate an eye-pleasing model. We present a simulation/location/data-neutral 3D visualization software module that allows the 3D display of an XML-coded road network, vehicles and communication involving vehicles. These objects are automatically modeled from source data with empty space filled in an intelligent way. The simulation can be freely explored in time and space. Vehicles can be color-coded, viewed from above or from the driver’s perspective, or alternatively in a traditional 2D form.

Keywords: Microscopic traffic simulation, data visualization, 3D visualization

8.2 Introduction

Microscopic traffic simulation has become an established tool for the examination of microscopic level “what if” questions in traffic engineering [1]. These can include proposed infrastructure changes, changes in traffic regulation, and devices in traffic providing communication between vehicles for purposes of improved traffic safety or efficiency. Although the critical outcomes of simulation are usually ‘hard’ numbers (such as average speeds, travel times, traffic flows), visualization of some kind is essential for purposes of visual validation and debugging. In both science and industry, 3D visualizations have become more popular: driver and traffic behavior is easier to judge, the viewer is more engaged in the presentation and the display invites exploration and questions. Hence such visualizations are useful both for demonstration and teaching.

For a traffic simulation, we usually only configure roads, junctions, traffic lights etc. There may also be objects in the simulation that correspond to real world objects, but are somewhat abstracted (for example, a network link with reduced speed might actually correspond to an overhead motorway control sign). Overall, the normal simulation objects are not enough to generate a realistic and attractive 3D visualization where viewers probably expect other real-world items such as safety barriers, vegetation and buildings. In addition, with the trend towards simulating communication amongst vehicles and infrastructure [2], we have data on communication that we also want to see in a simulation visualization. As tools for simulating communication are usually in the form of add-on modules that interface with the simulation tool (e.g. VSimRTI [3], iTETRIS [2]), the communication is not usually displayed in the simulation user interface. This abstraction also motivates the choice for the visualization to also be abstracted and separated from the simulation tool: modern questions of traffic engineering, in particular vehicular communication, may require a greater diversity of simulation tools, if only for verification in what is still a relatively young field [4]. We know of no popular microscopic traffic simulator that visualizes the communication between vehicles as part of the normal visualization. SUMO [5] in particular lacks any kind of 3D visualization, either online during the simulation or offline based on the simulation log file.

Furthermore, in modern traffic engineering, we also have access to other data with a similar level of detail as microscopic simulation results, such as Floating Car Data (FCD) or trajectories from video analysis. Given visualization software that only requires a network description and the trajectories of vehicles, there is no reason why these data can also not be viewed in the same animated, 3D format.

To resolve these issues, we have developed a module for the vtSim data and simulation management framework [4], vtSim.VIEW that visualizes any road network, set of vehicle trajectories upon it and communication between entities. The data source for the visualization is a simple XML: the main vtSim software is not necessarily required, acting only as a bridge between simulation-native data formats and vtSim.VIEW. Based on this road network, the module automatically generates much of the road environment and surroundings that are not defined in the simulation network: no 3D modeling is required on the part of the user. Although the visualization is offline in the sense that the simulation is performed independent of the visualization (only the data describing the network and vehicle trajectories are required), it is performed in real time based on this data, hence not requiring any advance video rendering before viewing. It should be stressed that the visualization is in no way a traffic simulator and is fully independent of whatever vehicle or communication models were used in the simulation. This also has the advantage that for slower-than-real-time simulations (as is often the case when communication between vehicles is included), the visualization is not delayed and can be viewed and reviewed from any viewpoint at the preferred speed.

The paper is structured as follows: first details of the implementation are covered, including a description of the data format and how the source data are interpreted and displayed. This is followed by a section on how the program can be used, including a number of examples that showcase the module features. We conclude with an outlook towards future work.

8.3 Implementation

8.3.1 Connection to other software modules

The data necessary for vtSim.VIEW are first a set of XML SUMO-compatible network source files (primarily edge and connection descriptions) and second a description (also in XML) of the vehicle data. These can either be created by the user (e.g. through conversion scripts) or by processing the network and results with the main vtSim tool. Here it is important to preserve the network, regardless of simulation tool used, as-simulated, which differs from the normal vtSim use case of importing networks and standardizing them into the SUMO format. Hence the network actually used for simulation is treated as a result of the simulation itself. Processing of the simulation output allows vehicle trajectories and communication data to be output for visualization. This workflow is summarized in Figure 8-1.

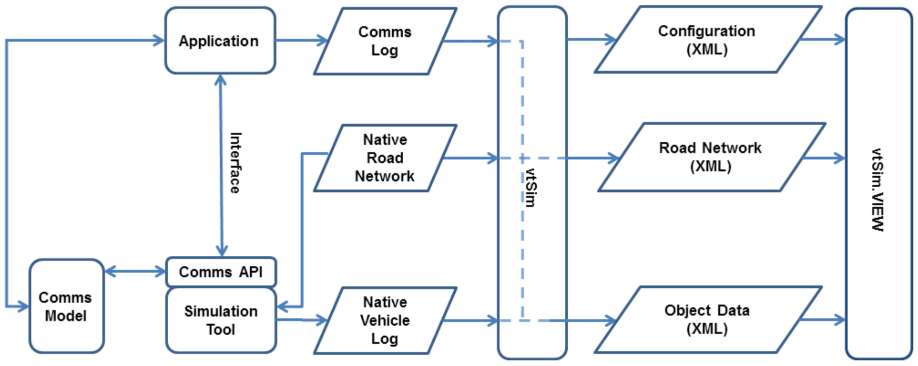


Figure 8-1: Visualization workflow. The network used for simulation along with the simulation vehicle trajectories log and communications log from the simulated application are prepared as XML files for the vtSim.VIEW visualization tool. This can either be done by the user as part of their results-handling process or achieved (currently for VISSIM) using vtSim.

8.3.2 Data format

The road network is coded in XML, broadly in accordance with the SUMO XML specification [6]. This specification is well documented and acts as a standard for the current vtSim software. Three XML files are required; .nod.xml representing road network nodes, .edg.xml representing road network edges (links) and .con.xml describing the connections between links. As we use both VISSIM [7] and SUMO [5] in our work, some parameters were added for preserving the geometry of edge connections (or in VISSIM terminology, connectors between links), which, unlike in SUMO, can have non-negligible lengths, geometries and must not necessarily begin or end at the start- or end-point of an edge. However, the data format used was intended to provide for SUMO compatibility. Vehicle data is structured as shown below.

The structure of an object data file for vtSim.VIEW. The minimum information to see moving vehicles are vehicle definitions with timestep entries describing their locations. In addition communication messages and vehicle statuses can be provided, referenced to relevant timestep entries. LKW represents truck traffic and PKW normal vehicles.

```
<vehicles>
  <vehicle id="29" type="LKW">
      <timestep LateralPosRel="0.5" LinkPos="927.10" LinkPosRel="0.97" X="25.62"
           Y="-89.61"      acceleration="1.42"      lane="2"      link="200733801"
           speed="24.67" time="300.0" timeGap="80.22">
           <warning distanceToEvent="9.7" type="congestion" />
           <status type="warned" />
      </timestep>
    ... further timesteps ...
  </vehicle>
  ... further vehicles ...
</vehicles>
```

Another scenario xml file states which files are required and which camera positions will be available. This file can be created manually or (as in vtSim) automatically written along with the data files to be visualized.

8.3.3 Language choice and libraries used

The module is written in Java, ensuring runtime compatibility with all desktop platforms. The fact that most popular traffic simulations are not written in Java is not a problem because of the modular structure of vtSim and vtSim.VIEW where data-transfer is achieved via files. SUMO in particular already takes an extensible approach to simulation whereby the core software is written in C++ but additional tools provided are written in Java or Python, and an online control interface for external applications is provided [5]. For 3D rendering, the open-source jMonkeyEngine [8] is used, which comes with some free models including the car object used. It also allows the display of all kinds of 3D models compatible with the free modeling software Blender [9], such as other objects that were acquired from the commercial model library TurboSquid [10].

8.3.4 Visualization

The conversion from the network representation to a (currently motorway-specific) 3D graphic is performed by first building a wireframe model based on the network description. Connections to other links and the number of lanes are taken into account. Figure 8-2 shows the construction of a road section schematically.

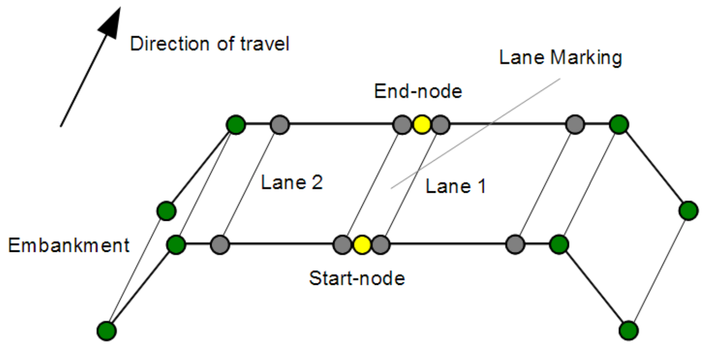


Figure 8-2: Schematic diagram of road link construction.

The edges of roads are augmented with a hard shoulder and curbing. Empty land is shown as grass featuring undulating hills and the sky is textured. Vehicles are shown with detailed vehicle models. The size of the network displayed by vtSim.VIEW is only limited by system memory. Efficient culling techniques in the jMonkeyEngine compute only the geometry which is actually visible on the screen.

In addition to the road geometry, vehicular communication and vehicle status can also be shown. A table of features currently available is shown below.

Table 8-1: Entities that can be visualized with the software.

Entity	Creation mechanism/relevant XML
Road	<edge> elements in edge definition XML
Road boundaries	Automated
Lane Markings	Automated based on road information
Vehicles	<vehicle> elements in vehicles definition XML; positions described by <timestep> elements per time step
Landscape	Automated grass & hills
Sky	Automated
Communication	<warning> element within a <timestep>
Vehicle Status	<status> element within a <timestep>

8.4 Usage

8.4.1 Control

Currently, navigation is accomplished with keyboard controls. A number of (starting) camera positions can be stored and switched between using numbered keys or an on-screen control (see Figure 8-3, below). This can include the 2D view. Other on-screen controls are also shown in Figure 8-3.

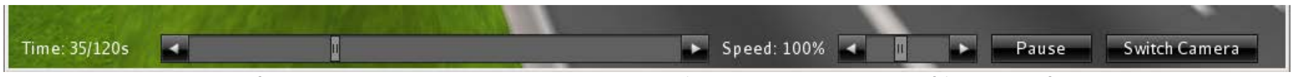


Figure 8-3: Controls for the visualization: the current time (shown on extreme left) can be freely selected using the slider on the left. Speed can be altered using the speed slider. The simulation can be paused at any time using the pause button, and the pre-defined camera positions can be used by clicking "Switch Camera".

8.4.2 Use cases

So far, the software has been used for simulations with and without vehicular communication in the research projects simTD [11] and SOCIONICAL [12], using source data from the VISSIM [7] simulator and data derived from video recordings. In simTD in particular the focus was communicating to the viewer what traffic situation required the communication application, and what the driver of the vehicle would see in such a situation. This can be seen in the driver viewpoint, in Figure 8-4 below. Visualization of communication data for a vehicular communication application can be seen in Figure 8-5. Visualization of data gathered from video is shown in Figure 8-6.

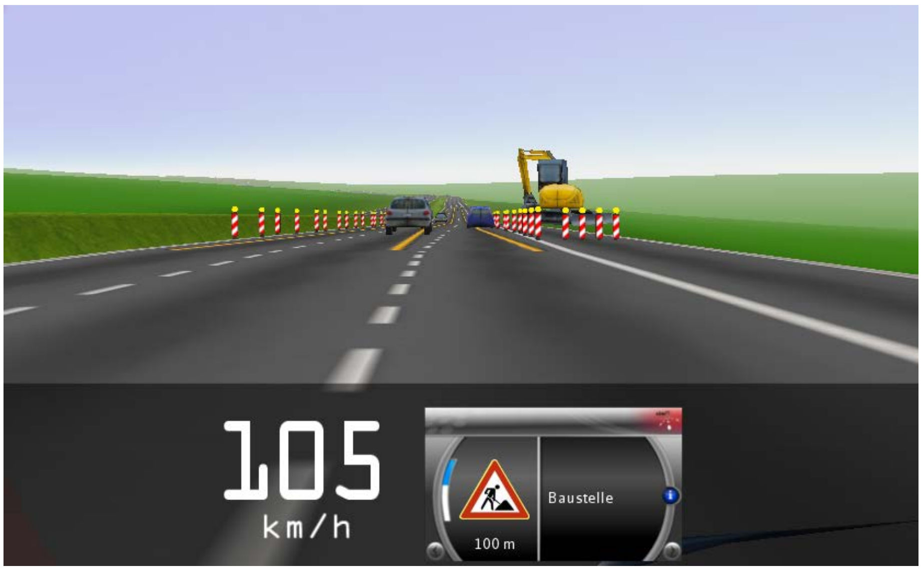


Figure 8-4: Driver view while approaching a work zone.

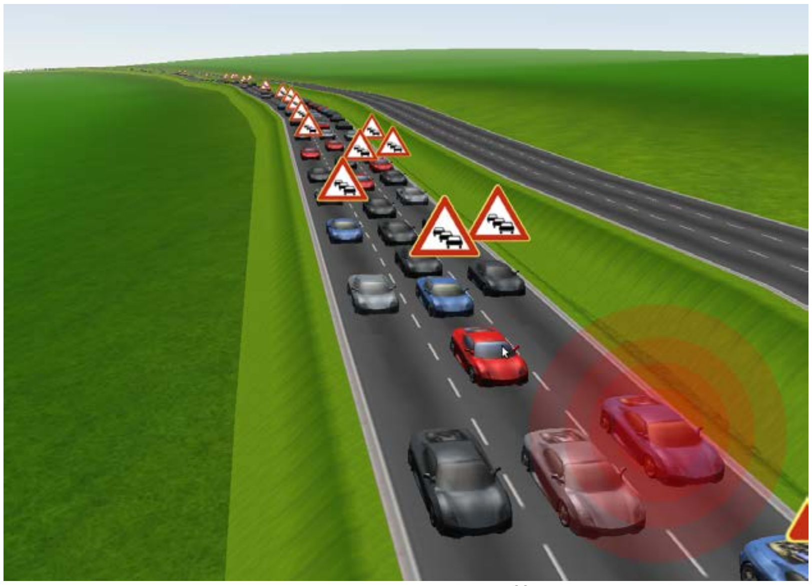


Figure 8-5: Vehicles receiving warnings about a traffic jam; communication shown.

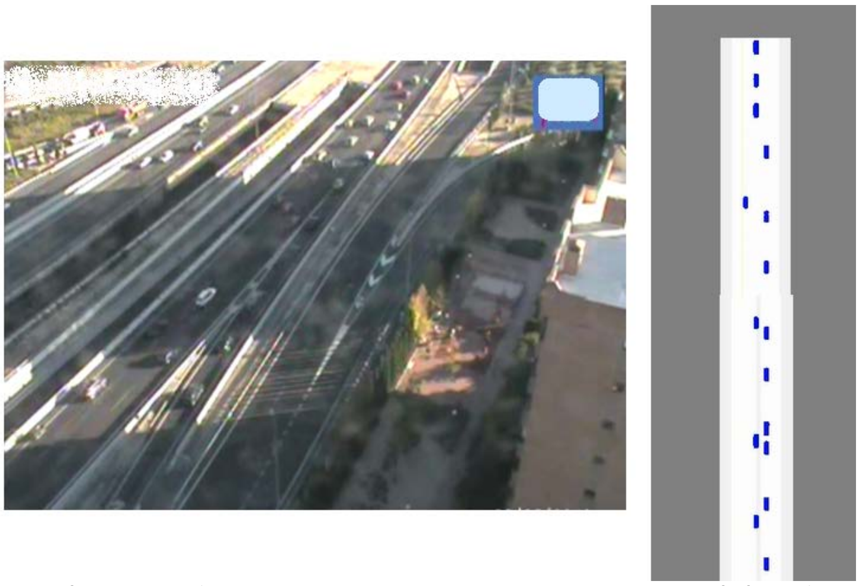


Figure 8-6: Data taken from video (merging and right-hand-lane vehicles only, left frame) animated in the 2D view (right frame).

8.5 Conclusions and Outlook

The creation of a tool for generic, realistic 3D visualization based only the simulation geometry, vehicle trajectories and (optionally) vehicular communication data has rapidly accelerated the process of creating life-like 3D material for exploration and display. The software is aimed at those in research and industry who require realistic visualizations that do not need to exactly reflect the exact geography of the modeled location. Likewise the software can add realism to generic simulations (e.g. vehicular communication system prototyping) that are not based on any real road network. As described in section 8.4, we have already used the software to great effect. By using a well-described, XML-based data format, the software is not specific to any one data source, simulated or otherwise.

We are currently extending the software to handle urban road networks and scenery, traffic lights, pedestrians, cyclists and roads that cross over one another (e.g. motorway junctions). To this end we will consider using information beyond edges and their shapes (which are normally enough for a motorway network) such as the shapes of junctions in the simulation. As not all necessary information can always be gleaned from the simulation data, an editor will be created where necessary information for the visualization can be annotated and stored.

8.6 References

[1] Barceló, J.: Fundamentals of Traffic Simulation. Springer, Heidelberg (2010)

[2] Härri, J., Cataldi, P., Krajzewicz, D., Blokpoel, R. J., Lopez, Y., Leguay, J., & Bieker, L.: Modeling and simulating ITS applications with iTETRIS. In: Proceedings of the 6th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, pp. 33--40. ACM, New York (2011)

[3] Rieck, D., Schünemann, B., Radusch, I., Meinel, C.: Efficient traffic simulator coupling in a distributed V2X simulation environment. In: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Article No. 72. ICST, Brussels (2010)

[4] Baur, M., Fullerton, M., Busch, F.: Realizing an Effective and Flexible ITS Evaluation Strategy Through Modular and Multi-Scaled Traffic Simulation.
Intelligent Transportation Systems Magazine. 2, 34--42 (2010)

[5] Krajzewicz, D.: Traffic Simulation with SUMO - Simulation of Urban Mobility. In: Barceló, J. (ed.) Fundamentals of Traffic Simulation. International Series in Operations Research and Management Science, pp. 269--294. Springer, Heidelberg (2010)

[6] SUMO User Documentation - Networks/Building Networks from own XML-descriptions, http://sumo.sourceforge.net/doc/current/docs/userdoc/Networks/Building_Networks_fro m_own_XML-descriptions.html

[7] Fellendorf, M., Vortisch, P: Microscopic Traffic Flow Simulator VISSIM. In: Barceló, J. (ed.) Fundamentals of Traffic Simulation. International Series in Operations Research and Management Science, pp. 63--93. Springer, Heidelberg (2010)

[8] jMonkeyEngine, http://www.jmonkeyengine.com

[9] Blender, http://www.blender.org

[10] TurboSquid, http://www.turbosquid.com

[11] simTD, - Safe and Intelligent Mobility Test Field Germany, http://www.simtd.de/index.dhtml//-/enEN

[12] Complex socio-technical system in ambient intelligence (SOCIONICAL), http://cordis.europa.eu/search/index.cfm?fuseaction=proj.document&PJ_RCN=1039312
8

9 Implementation of an Energy Model and a Charging Infrastructure in SUMO

Tamás Kurczveil, Eckehard Schnieder
Institute for traffic safety and automation engineering, Technische Universität Braunschweig, Braunschweig, Germany

9.1 Abstract

Future traffic that will be accompanied by higher alternative drive concepts will pose as a challenge when it comes to corresponding energy systems, coordination of operations, and communication interfaces, such as needed for data acquisition and billing. On one hand, the increasing attractiveness of electric vehicles will inevitably lead to the development and testing of compatible technologies; on the other, these will need to be conformed to existing systems, when integrating them into the prevailing infrastructure and traffic. Funded by the German Federal Ministry of Transport, Building and Urban Development, an inductive vehicle charging system and a compatible prototype bus fleet shall be integrated into Braunschweig's traffic infrastructure in the scope of the project emil (Elektromobilität mittels induktiver Ladung – electric mobility via inductive charging). This paper describes the functional implementations in SUMO that are required by the methodic approach for the evaluation of novel charging infrastructures by means of traffic simulation.

Keywords: Traffic simulation, urban traffic, inductive energy transfer, public transportation, vehicle model.

9.2 Introduction

The prospective post-oil era and rising fuel prices have lately resulted in several global trends towards alternative drive technologies. The main advantage of gasoline fuel over other energy carriers is its high specific energy of up to 44.0 MJ/kg. Current projections for the development of equally convenient alternative energy storages go far beyond 2030. Thus, the utilization of alternative energy sources, such as the electrochemical energy stored in Lithium-Ion batteries (0.5 MJ/kg), will result in high vehicle masses and/or low ranges for the coming decades [1][2].

Measures that aim to counter these deficits include the application of light-weight materials, energy/time-optimal routing, intelligent control of traffic light-signal systems, and government regulations that introduce (operational, financial and/or infrastructural) incentives for buyers of vehicles with alternative drive concepts.

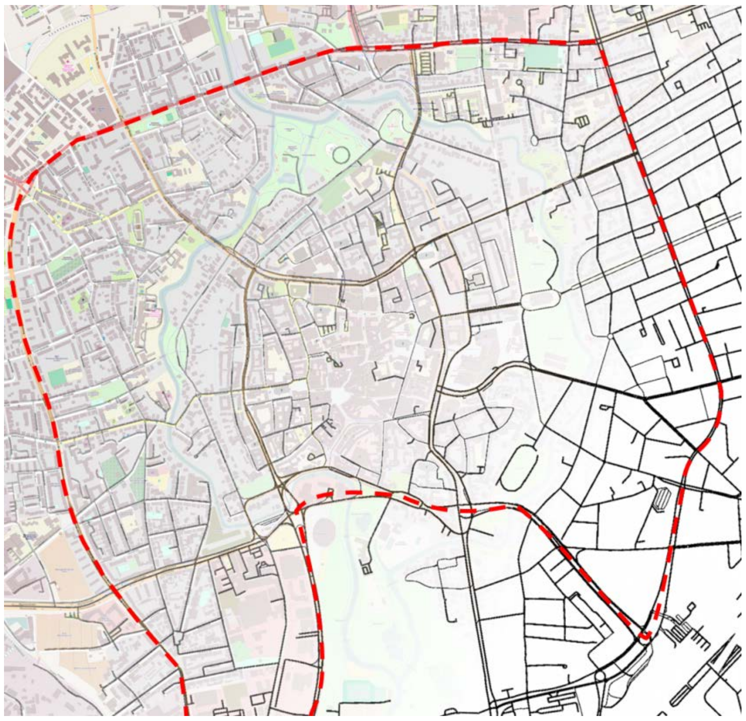


Figure 9-1: Braunschweig's urban road network with the route of bus lines M19 and M29 (red)
© OpenStreetMap-Contributors

The German Federal Ministry of Transport, Building and Urban Development (BMVBS) has therefore granted the funding of the project *emil* (Elektromobilität mittels induktiver Ladung – electric mobility via inductive charging). It includes the evaluative implementation of an inductive energy transfer system for public transportation and a compatible prototype bus fleet in the city of Braunschweig. The bus lines in question are the M29 and M19 that circle the city center counter-clockwise and clockwise, respectively. They carry the highest percentage of Braunschweig's publicly transported passengers with a high frequency from and to major traffic nodes and landmarks such as the central train station and the university. Figure 9-1 illustrates Braunschweig's urban road network.

The goal of the project *emil* is to analyze and optimize the operation and economic feasibility of an inductive electric charging system and to develop suitable operating strategies. One focus of research lies in the analysis of integrative aspects that allow for the common utilization of the charging and road transport infrastructure by public and private transport with minimum obstruction of public and total traffic. After outlining the methodic approach for these analyses, the implementations in SUMO shall be described that allow for its utilization in urban traffic optimization.

9.3 Methodic Approach

The goal of possible optimization measures (e.g. via genetic algorithms) lies in finding an optimum of the considered system in regard of the energy consumed. This evaluation requires a suitable simulation tool that allows the implementation of custom traffic scenarios, including traffic demand, a charging infrastructure, customized vehicles, prioritization, and different light-signal schedules. Additionally, it needs to generate an output of the required energy of individual road traffic participants and the entire system as the desired optimization criterion.

Since no simulation tools exist that allow all the mentioned requirement to be modeled, the best choice for this task is the traffic simulation tool SUMO (Simulation of Urban MObility), due to its open source character and high compatibility with numerous data sources, including many commercially available traffic simulation tools [3][4]. Its development was initiated by the Institute of Transportation Systems of the German Aerospace Center (DLR), in 2001. It has evolved into a simulation tool, high in features, functionality and interfaces. Even though instantiated vehicles follow a simplified behavior, traffic simulation tools like SUMO allow the realistic replication of prevailing traffic in arbitrary road networks.

The intended approach in the scope of this project is to model current representative traffic scenarios that take into account the existing infrastructure, a time-varying traffic demand model (differentiating between representative peak and nonbusy periods), and light-signal schedules [5]. Meanwhile, the implementation of new functionalities in SUMO on the system-function level will allow for the instantiation of inductive charging stations and compatible vehicles. Operation parameters will have to be identified and/or set for the inductive charging system and traffic demand, to create a representative scenario for Braunschweig's urban traffic. Figure 9-2 depicts the above described method, highlighting interfaces between SUMO's current functionalities, required additional functionalities, and an external optimization framework.

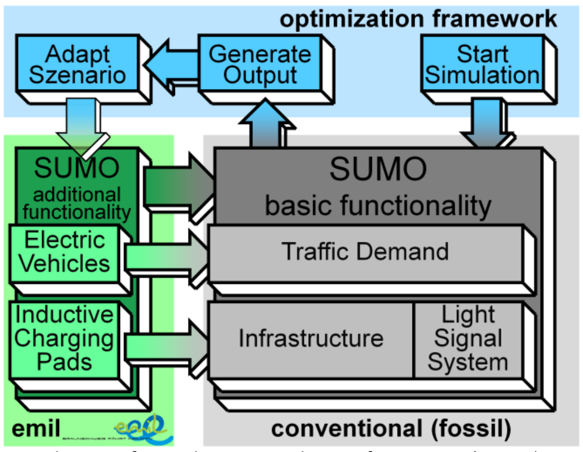


Figure 9-2: Project structure with interfaces between basic functionalities (gray), additional functionalities (green), and an optimization framework (blue)

9.4 State of the Art and Implementation

In order to generate an output about the consumed energy, an energy model will have to be implemented for instantiated vehicle objects. Numerous methods, functions and approaches exist that perform traffic quality assessment calculations, giving feedback about its characteristics. Implemented functionalities in SUMO include the vehicle- and lane-based HBEFA-emission [6] and HARMONOISE-noise [7] calculation and the corresponding generation of a suitable output.

In order to allow energy assessments and subsequent optimization, an additional framework has also been developed by Maia et. al. [8]. The work is based on a sophisticated vehicle model that takes into account mechanic and electric vehicle parameters and calculates the variation of the depth of discharge between discrete time steps. If a vehicle's movement requires a higher torque than its defined maximum, recalculations of the vehicle speed and acceleration take place in order to comply with the constraints of its components. The depth of discharge is subsequently calculated using an electrical traction model.

With the aim to enhance computation time, this paper presents the implementation of a similar vehicle model, which merely focuses on energy as the simulation output and reduces the complexity of required calculations. Additionally, this model evaluates the energetic state and calculates variations in energy content of corresponding vehicles without affecting their driving behavior. Additional benefits of this model include fewer required input arguments for instantiating vehicle objects.

9.4.1 Vehicle Energy Model

The change of one vehicle's energy content can be calculated by summing its kinetic, potential, and rotational energy gain components from one discrete time step to the following, and subtracting the losses caused by different resistance components [9]. The vehicle's energy $E_{\mathrm{veh}}[k]$ at the discrete time step k can thus be calculated by equation 1, with the known variables vehicle mass m, time variant vehicle speed $v[k]$, gravity acceleration g, time variant vehicle altitude $h[k]$, and moment of inertia of internal rotating elements J_{int}.

$$\begin{aligned} E_{\mathrm{veh}}[k] &= E_{\mathrm{kin}}[k] + E_{\mathrm{pot}}[k] + E_{\mathrm{rot,int}}[k] \qquad (1) \\ &= \frac{m}{2}\cdot v^2[k] + m\cdot g\cdot h[k] + \frac{J_{\mathrm{int}}}{2}\cdot v^2[k] \end{aligned}$$

In consideration of energy losses $\Delta E_{\mathrm{loss}}[k]$ caused by air, rolling, and curve resistance and constant consumers (e.g. air conditioning), the energy gain between time steps k and k+1 can be calculated by equation 2.

$$\Delta E_{\mathrm{gain}}[k] = E_{\mathrm{veh}}[k+1] - E_{\mathrm{veh}}[k] - \Delta E_{\mathrm{loss}}[k] \qquad (2)$$

The energy loss is made up of the components in equation 3, with the variables air density ρ_{air}, vehicle front surface area A_{veh}, air drag coefficient c_{w}, covered distance $s[k]$, rolling resistance coefficient c_{roll}, centripetal force F_{rad}, curve resistance coefficient c_{rad}, and the (average) power of constant consumers P_{const} [9].

$$\begin{aligned} \Delta E_{\mathrm{loss}}[k] &= \Delta E_{\mathrm{air}}[k] + \Delta E_{\mathrm{roll}}[k] + \Delta E_{\mathrm{curve}}[k] + \Delta E_{\mathrm{const}}[k] \qquad (3) \\ \Delta E_{\mathrm{air}}[k] &= \frac{1}{2}\rho_{\mathrm{air}}\cdot A_{\mathrm{veh}}\cdot c_{\mathrm{w}}\cdot v^2[k]\cdot|\Delta s[k]| \\ \Delta E_{\mathrm{roll}}[k] &= c_{\mathrm{roll}}\cdot m\cdot g\cdot|\Delta s[k]| \\ \Delta E_{\mathrm{curve}}[k] &= c_{\mathrm{rad}}\cdot\frac{m\cdot v^2[k]}{r[k]}\cdot|\Delta s[k]| \\ \Delta E_{\mathrm{const}}[k] &= P_{\mathrm{const}}\cdot\Delta t \end{aligned}$$

Depending on its sign, $\Delta E_{\mathrm{gain}}[k]$ is the amount of energy the vehicle has consumed or regained resulting from its movement. The variation of the energy contained in the vehicle's battery can further be calculated by equations 4 and 5 by introducing constant efficiency factors for recuperation η_{recup} ($\Delta E_{\mathrm{gain}}[k] > 0$) and propulsion η_{prop} ($\Delta E_{\mathrm{gain}}[k] < 0$).

$$E_{\mathrm{Bat}}[k+1] = E_{\mathrm{Bat}}[k] + \Delta E_{\mathrm{gain}}[k]\cdot\eta_{\mathrm{recup}} \qquad (4)$$

$$E_{\mathrm{Bat}}[k+1] = E_{\mathrm{Bat}}[k] + \Delta E_{\mathrm{gain}}[k]\cdot\eta_{\mathrm{prop}}^{-1} \qquad (5)$$

9.4.2 Vehicle Charging Model

For the purpose of evaluating a charging infrastructure, a new object will have to be implemented into SUMO that supplies compatible vehicles (or their batteries) with energy for their operation. The location of charging stations as well as their charging power and efficiency needs to be specifiable by the user. If a vehicle moves or stops above or within a system-specific proximity of such an infrastructure element, the energy content of its battery is charged according to equation 6, with charging power P_{chrg}, charging efficiency η_{chrg}, and duration between two discrete time steps Δt.

$$E_{\text{Bat}}[k+1] = E_{\text{Bat}}[k] + P_{\text{chrg}} \cdot \eta_{\text{chrg}} \cdot \Delta t \qquad (6)$$

Following the calculations of the energy variation between two discrete time steps, the battery's energy content is limited to the user-specifiable range

$$0 \le E_{\text{Bat}} \le E_{\text{Bat,max}} \, . \qquad (7)$$

Calculations of this energy model can be restricted to vehicles with $E_{\text{Bat,max}}>0$, further reducing computing times.

9.4.3 Simulation and Results

For the correct dimensioning of components and layout, the technology provider Bombardier Transportation GmbH has developed a sophisticated simulation model for the battery's charge and discharge. Since the vehicles, which are to be used in this project, are still in development, this output of this sophisticated model is the only reference for a representative parameterization. In order to show that the simplistic vehicle model presented above is capable of calculating the trend of a vehicle's energy content with adequate accuracy, it has been given the same route as input, as Bombardier's sophisticated model (Solaris Urbino 12). Figure 9-3 shows the vehicle's route and its topographic profile.

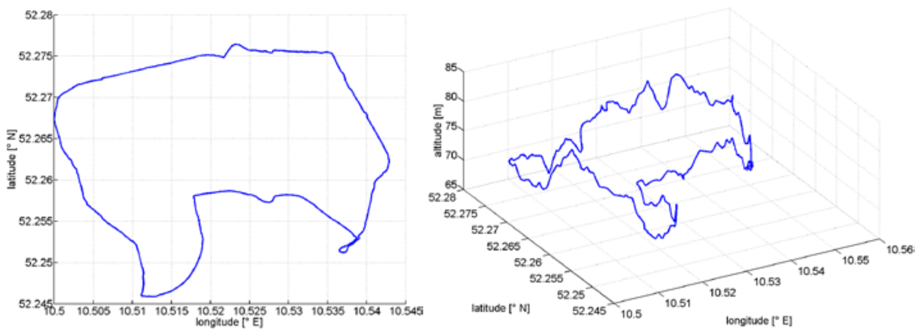


Figure 9-3: Designated bus route (left) and its topographic profile (right)

In a subsequent step, parameters for the newly developed vehicle model in SUMO have been determined that represent the reference behavior optimally, in the sense of least-squares. The reference (blue) and parameterized (red) simulation outputs for the same route as the model input are shown in Figure 9-4. The cumulated (Δt=1s) deviation of the two simulation outputs add up to E_{Error}=3.3998 kWh^2.

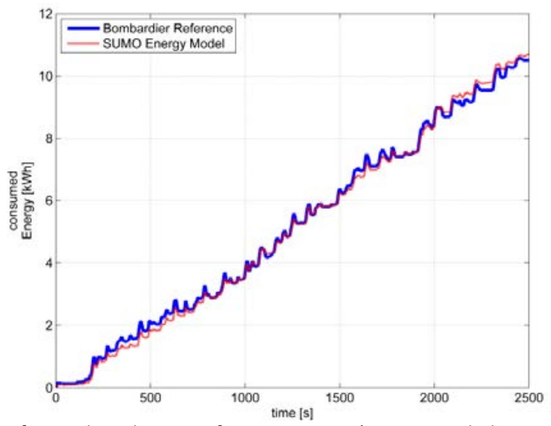


Figure 9-4: Simulation outputs of Bombardier's reference simulation and the newly implemented energy model with an optimal parameter set

9.5 Conclusion and Outlook

The identified parameter set can be used in the resulting function package of SUMO for the instantiation of a new traffic demand model with (representative) objects of the new vehicle class, including different vehicle types. This will allow for the development of different scenarios for Braunschweig’s traffic (including forecasts for electric vehicles) that can be further analyzed and optimized in regard of the new inductive charging infrastructure and its participants.

The development of an optimization framework with underlying algorithms will require an additional output of the specific simulation states by producing a feedback on the energy content of relevant participants. This output could be implemented in form of a custom device or detector. Potentials for optimization lie in the optimal positioning of the charging stations along the defined bus route. The optimization criteria can not only include a desired long battery life, but also travel time by synchronizing required charging times and the predictable waiting times for the entry and exit of passengers, light-signal systems, and remaining traffic. For the identification of optimal parameter sets, the utilization of genetic algorithms is intended in the further course of the project.

Simulation results can also include the evaluation of occupancy rates at different positions within the road network. This data can be used for the design and alignment of inductive charging pads that maximize their duty cycle and thus efficiency.

By implementing different scenarios for the amount of compatible vehicles in the future, it will also be possible to determine saturation points for the amount of participating vehicles, where operational interferences and obstructions among public transportation and between public and private vehicles can be expected.

9.6 Acknowledgements

The project *emil* is funded by the German Federal Ministry of Transport, Building and Urban Development (Bundesministerium für Verkehr, Bau und Stadt-entwicklung – BMVBS). We hereby thank all our project partners for their continuous and kind cooperation.

Bundesministerium
für Verkehr, Bau
und Stadtentwicklung

9.7 References

[1] Winter, M: Elektromobilität mit Lithium-Ionen-Technologie: Chancen, Herausforderungen, Alternativen. Proceedings of the HEV 2012, Hybrid and Electric Vehicles, Braunschweig (2012)

[2] Wansart, J.: Analyse von Strategien der Automobilindustrie zur Reduktion von CO2-Flottenemissionen und zur Markteinführung alternativer Antriebe: Ein systemdynamischer Ansatz am Beispiel der kalifornischen Gesetzgebung. Dissertation, Technische Universität Braunschweig, Springer Gabler, Wiesbaden (2012)

[3] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO – Simulation of Urban MObility: An Overview. SIMUL 2011, The Third International Conference on Advances in System Simulation, Barcelona (2011)

[4] Detering, S.: Kalibrierung und Validierung von Verkehrssimulationsmodellen zur Untersuchung von Verkehrsassistenzsystemen. Dissertation, Technische Universität Braunschweig (2011)

[5] Handbuch für die Bemessung von Straßenverkehrsanlagen (HBS). Forschungsgesellschaft für Straßen- und Verkehrswesen (2001)

[6] Keller, M., de Haan, P.: Handbuch Emissionsfaktoren des Straßenverkehrs 2.1 – Dokumentation. INFRAS, Bern/Heidelberg/Graz/Essen (2004)

[7] Nota, R., Barelds, R, van Maercke, D.: Harmonoise WP 3 Engineering method for road traffic and railway noise after validation and fine-tuning – Technical Report Deliverable 18, HARMONOISE (2005)

[8] Maia, R., Silva, M., Araújo, R., Nunes, U.: Electric Vehicle Simulator for energy Consumption Studies in Electric Mobility Systems. 2011 IEEE Forum in Integrated and Sustainable Transportation Systems, Vienna (2011)

[9] Mitschke, M., Wallentowitz, H. Dynamik der Kraftfahrzeuge. Springer, Berlin (2004)

10 A Framework for Electric Bus Powertrain Simulation in Urban Mobility Settings: coupling SUMO with a Matlab/Simulink nanoscopic model

Jose Macedo, Guilherme Soares, Zafeiris Kokkinogenis, Deborah Perrotta, Rosaldo J. F. Rossetti;
Artificial Intelligence and Computer Science Laboratory (LIACC), Department of Informatics Engineering (DEI), Portugal
Zafeiris Kokkinogenis;
Institute of Mechanical Engineering (IDMEC), Faculty of Engineering, University of Porto, Portugal
Deborah Perrotta;
Centro Algoritmi, Universidade do Minho, Portugal

10.1 Abstract

The increasing traffic volumes in urban areas result in heavy environmental condition. The necessity for a rapid and sustainable transport of people and goods became one of the high priorities in the public governance's agenda. Government agencies and organizations try to develop more stringent standards for the fuel consumption and gas emissions through reduction of the congestion on network infrastructures. Among the different solutions under exploration to tackle congestion problems is the development of Electric Vehicles for personal and public transport as an alternative to internal combustion engine vehicles. Albeit this direction seems to be promising, there are still open concerns associated with the consumption of energy and other performance measures for considering the implementation of electric buses in urban settings as a cost-effective solution. An important aspect in evaluating the performance and adequateness of such vehicles is the fact they must be immersed in the urban environment context. This work presents the distributed simulation architecture for electric bus powertrain simulation within a realistic urban mobility context. The architecture uses the microscopic simulator SUMO to represent the urban reality coupled with a nanoscopic model of an electric bus powertrain subsystem implemented in Matlab/Simulink. The paper discusses on how such a platform will be important for analysing the way traffic flow and its dynamics affect the performance of electric buses when there are obstructions or intense traffic conditions.

Keywords: Microscopic simulation, nanoscopic model, SUMO, Matlab/Simulink, electric vehicles, public transport.

10.2 Introduction

In the last decades we witnessed a large increase in traffic and transport demand that has created and aggravated capacity problems in the infrastructure causing traffic congestions and delays. Heavy traffic conditions not only affect the welfare of citizens from the economic point of view but they are also related to their health status speaking both psychologically, due to stress accumulated during their travels, and physically, due to high air pollution levels. In fact, a problem associated with the increasing use of personal vehicles is the carbon emissions. According to the 2009 Urban Mobility Report [1] the congestion led urban Americans to travel 4.2 billion hours more, which resulted in 2.8 billion gallons of extra fuel, an increase of more than 50% over the previous decade. Government agencies and organizations try to develop more stringent standards for the fuel consumption and gas emissions through reduction of the congestion on network infrastructures [2], [3].

In this sense, incentives and investments on public transport modes as well as research on more eco-sustainable solutions have been performed as attempts to minimize both the air pollution and congestion problems. One of the approaches currently investigated and implemented to provide an eco-sustainable solution for the public transport is related to the employment of electric bus powertrains in metropolitan transportation as an alternative to internal combustion engine buses [4]. However, there are still open issues related to the consumption of energy and other performance measures for considering the adoption of electric buses in urban scenarios as a cost-effective solution [5].

In order to analyse the adoption of electric buses as a cost-effective solution, it is necessary an evaluation of the performance and adequateness of such vehicles while immersed into an urban environment context. Albeit there are different tools and models to assess the behaviour of electric buses, such evaluations often lack the aforementioned integration with the traffic dynamics. This work intends to contribute to the improvement of transportation and traffic analysis tools and presents a flexible integrated architecture for electric bus powertrain simulation in urban mobility settings. Such a platform will be important for analysing how traffic flow and its dynamics affect the performance of the electric bus when there are obstructions or intense traffic conditions.

The paper describes how the microscopic traffic simulator SUMO (Simulation of Urban Mobility) [6] is used as a means to represent an urban traffic scenario and its integration with a model of an electric bus powertrain subsystem [7], implemented in MatLab/Simulink. The remainder of the paper is organized as follows. Section 10.3 introduces the proposed architecture. Section 10.4 presents a simple driving scenario in order to show the usefulness of the platform architecture. Section 10.5 draws some conclusion and discusses future developments.

10.3 Proposed architecture

Figure 10-1 depicts the software architecture for the electric bus simulation in a traffic environment. The proposed design is distributed combing the microscopic and nanoscopic aspects of the real system.

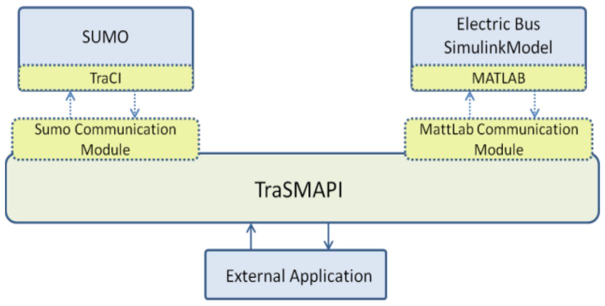

SUMO
TraCI
Electric Bus SimulinkModel
MATLAB
Sumo Communication Module
MattLab Communication Module
TraSMAPI
External Application

Figure 10-1: Proposed architecture.

The architecture consists of six modules, shortly described in subsections that follow.

10.3.1 SUMO Microscopic Traffic Simulator

SUMO (Simulation of Urban MObility) is a well-known open-source microscopic traffic simulation package that appears for the first time in 2001. Today SUMO is already in its 0.16 version and is one of the most commonly used microscopic traffic simulator by the research community. SUMO became an important tool in many researches in the urban traffic and transportation domains. It has gained its place among the academic community with many of scientific papers referring to it. Today, SUMO is not just a traffic simulator, but rather a suite of applications which help to prepare and to perform the simulation of traffic in complex settings.

SUMO can be seen as a pure microscopic traffic simulation: each vehicle is associated with an identifier, the departure time, and the vehicle's route through the network. It is also possible to describe each vehicle in more detail. For example, it is possible to define arrival properties of the vehicle, which lane it could use, its velocity and its position. Also, each vehicle can be assigned to a vehicle-type which describes its physical properties and variables of the used movement model. This is an important feature for this work since it allows us to make a one-to-one association between the bus vehicle in SUMO and the electric bus powertrain subsystem (EBPS) of the electric bus.

It represents and simulates the traffic environment that is the road infrastructures and the vehicles moving on it providing a microscopic level of systems resolution. Within this module the TraCI API is integrated for gaining access to the SUMO simulator.

10.3.2 The MatLab/Simulink Electric Bus Powertrain Subsystem

The Electric Bus Powertrain subsystem, or EBPS for short, is a mathematical model of an electric bus subsystem implemented in MATLAB's Simulink (Figure 10-2). MATLAB (MATrix LABoratory) is a high performance software suit for numerical computation, data visualization and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. It is an interactive system and a programming language for computing technical and scientific cooperation in general [8]. MATLAB features a family of specific applications called toolboxes, very important to most users of that suit.

Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Domains to which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others [9].

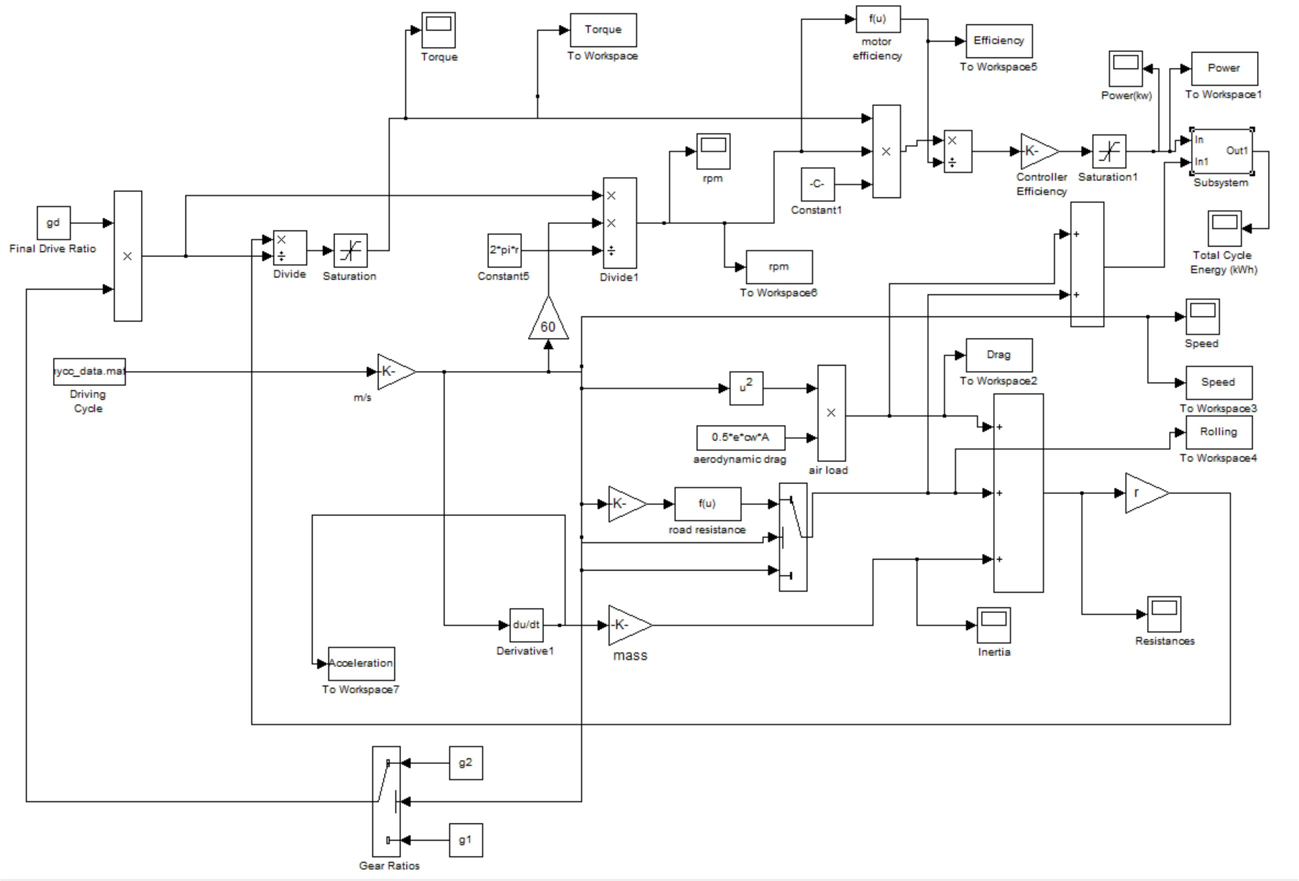


Figure 10-2: Main subsystem of EBPS model.

Simulink is a software package for modelling, simulating, and analysing dynamic systems. It supports linear and non-linear systems, modelled in continuous time, sampled time, or a hybrid of the two. Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. After defining a model, it is possible to simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in MATLAB's command window [10]. The simulation results can be put in the MATLAB workspace for post processing and visualization. And because MATLAB and Simulink are integrated, it is possible to simulate, analyse, and revise models in either environment at any point [11].

The model used has several subsystems, which are used to calculate specific parameters. One of these subsystems represents the vehicle's powertrain, taking into account the forces that work against its movement and the gear ratios involved.

An output of this subsystem computes the amount of required energy for a driving cycle to be completed. There is a third subsystem that calculates the amount of energy that may be possibly recovered from the regenerative braking, taking into account the kinetic energy of the vehicle. The two other subsystems are related to the batteries and the super-capacitors, evaluating whether they are capable of absorbing the energy from the braking [7]. Figure 10-2 illustrates the main subsystem of the model.

10.3.3 TraSMAPI

TrasMAPI is a tool for the simulation of dynamic control systems in road networks, with special emphasis on Multi-Agent Systems. This tool allows real-time communication with microscopic simulators providing a framework for the development of multi-agent solutions [12]. The abstraction over the simulator enables it to run different traffic simulators using the same API (Application Programming Interface) allowing, for example, the comparison of results of the same application in different simulators [13].

This tool can be seen as a generic microscopic traffic simulator API that allows a higher level of abstraction and transparency, in terms of variable or simulation management. It provides methods to control the simulation life-cycle and allows the coupling of an external application for simulation data management.

However, the importance of TraSMAPI for this project is mainly related with the communication module that it provides. TraSMAPI will be mainly used on the integration of both systems, namely SUMO and EBPS. In this sense, its abstraction layer for interaction with microscopic simulators brings great value for the integration, allowing an easier control of the simulation process.

10.3.4 The MatLab Communication Module

MatLab could be seen as an API for Simulink since Simulink models can be controlled by MatLab methods. Simulink models are standalone models and exist in mutual symbiosis with the MatLab environment. The only way to access them externally is through MatLab's methods and calls. For this reason a "control" using MatLab's interface to Simulink was applied. In this sense, this module extends TraSMAPI's API to communicate with Simulink through MatLab.

In order to establish communication with MatLab, the *matlabcontrol* API was used. The *matlabcontrol* is a Java API to interact with MatLab allowing its methods

invoke behaviour of Java objects [14]. Using this, it was only necessary to create a proxy to work as an image of the MatLab application. After that, all the calls to MatLab are performed through this proxy.

Establishing the communication with MatLab was the first part in the development of this module. The second part was to control the Simulink model's step-by-step simulation through Matlab commands. To do so, the *set_param()* function was chosen. This function allows starting, pausing, stopping and resuming the simulation, as well as advancing one simulation step at a time.

10.3.5 The SUMO Communication Module

Unlike Matlab/Simulink, SUMO already comes with an API that provides a communication protocol. In fact, the TraCI interface provides a set of methods that allow an easy interaction with the simulator's state variables. In this sense, this module implements TraSMAPI's API to communicate with SUMO through TraCI's API. This module is composed of a set of functions, implemented in Java that comprises the necessary commands to interact with SUMO. TraCI has an extensive number of methods, each one associated with an entity in the simulation. For the scope of this project, the only TraCI's methods that were used are relative to the vehicle entity and its speed.

10.3.6 External Application

To perform the integration, the communication modules herein described were used to implement the TraSMAPI's interface. In this way, although each simulation tool has its own implemented methods, the external application could control and communicate with both simulation tools using TraSMAPI generic interface.

The application is responsible for the synchronization and the data exchange between the two environments. The main steps of the program are:

1. Launch the simulators, and establish a connection with them;
2. Start the simulation on both systems at the same time, and keep them waiting;
3. Advance simulation one step;
4. Get the speed of a SUMO vehicle and send it to the EBPS simulation process;
5. Get the EBPS outputs;
6. Advance simulation one step;
7. Stop both simulations and disconnect from respective simulators;
8. Close simulators.

Steps 4 to 6 are performed in a cycle until the last simulation step. In this integration a SUMO vehicle is delegated by the EBPS Simulink model to act on behalf of it. Thus a bus vehicle performs a trip on a given route of a SUMO's network and sends feedback to Simulink. The EPBS model computes a number of metrics, defined in [7], based on the information received by the SUMO vehicle.

10.4 Simulation scenario

As it has been mentioned previously, the devised integrated platform intends to offer a valid tool to traffic managers and practitioners for analysing how traffic flow and its dynamics affect the performance of electric buses when there are obstructions or intense traffic conditions. In this section we present a scenario where two different traffic flows were specified, so that the EBPS could be exposed to different solicitations due to daily traffic conditions. This illustration serves only as a demonstration of the possible type of analysis one can conduct using the proposed tool.

For this purpose a new road network was designed and created for SUMO. The network is a model of the central area of Porto's down-town (Figure 10-3) and it has been extracted from OpenStreetMaps database. The choice of this urban traffic area is motivated by the fact that it is an area of the centre of the city with high traffic densities and a large number of traffic lights.

In order to provide an example of what kind of study the framework can perform for the EBPS analysis, the *Acceleration* and *TotalEnergyCycle* metrics have been chosen among others. Other metrics are defined in [7].

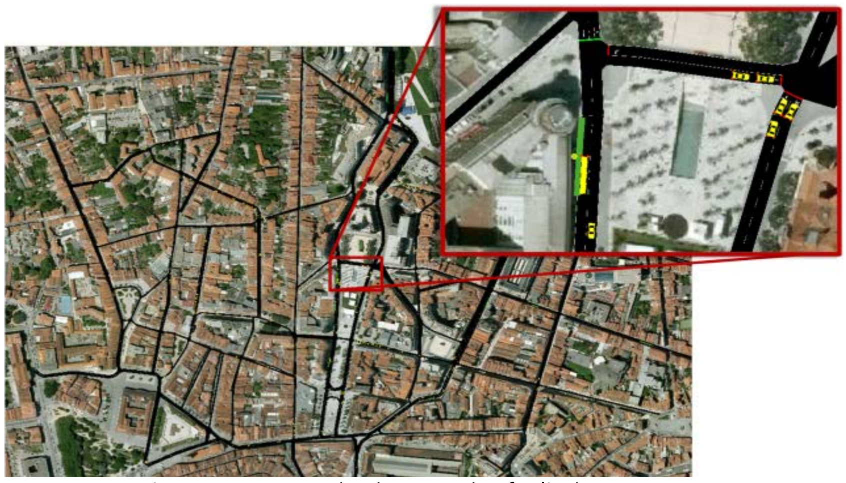


Figure 10-3: Test bed network of Aliados Avenue.

For the first set-up a repetition of five seconds has been used for the traffic flow generation. Since one second represents one step in simulation, this means that each trip is repeated by a new vehicle at every step. The second set-up of traffic flow has been generated with a repetition trip rate of one second. With this rate, rapidly the network begins to become overloaded, representing peak-hour periods.

The first parameter to be analysed is the acceleration profile of the electric bus under the different traffic conditions. It can be observed that under intense traffic conditions, there were performed tougher accelerations and decelerations (represented by the peaks on the plot of Figure 10-4). Besides that, the stop-and-go episodes were characterized by being performed on smaller periods of time when compared to low traffic. This type of behaviour have direct negative implications on the maintenance costs and on the efficiency of the bus, once the higher values of acceleration impact on the necessary energy for the bus to transverse a certain route.

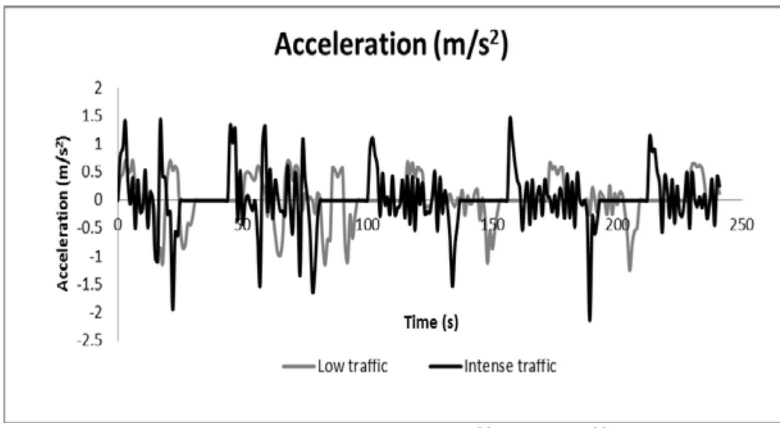


Figure 10-4: EBPS acceleration under different traffic conditions.

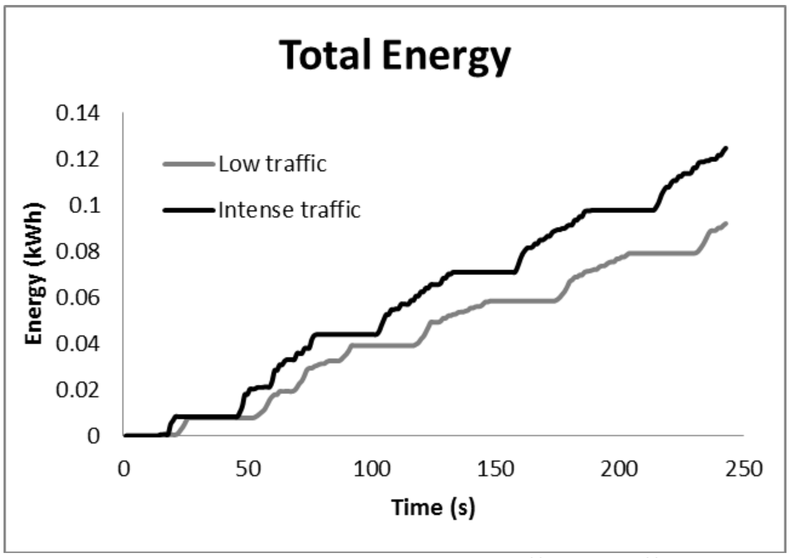


Figure 10-5: EBPS energy consumption under different traffic conditions.

After observing the acceleration profile, it was decided to analyse the amount of energy that the electric bus spent under the proposed conditions, as a way to corroborate the assumptions made previously. Not surprisingly, under intense traffic, the electric bus spent 36% more energy (0.125 kWh) when compared to the low traffic scenario (0.092 kWh), as it can be observed in Figure 10-5. This has immediate consequences on the costs associated with the recharge of the bus.

It could be concluded that intense traffic conditions affect negatively the performance of an electric bus, making it spend more energy to perform similar routes and also demanding more power from the bus motor, which implies on higher maintenance costs.

10.5 Conclusions

This paper discusses on the implementation of a distributed architecture for electric bus powertrain simulation in an urban mobility context using two types of simulation environments, namely the SUMO (Simulation of Urban Mobility) microscopic traffic simulator and the MatLab/Simulink environment. The proposed architecture envisions for a flexible approach to the integration of two simulators, one from the transportation area, and the other from automotive area. Its purpose is to offer a valid tool for traffic managers and practitioners so they can analyse how traffic flow and its dynamics affect the performance of electric buses (as well as other type of vehicles) when there are obstructions or intense traffic conditions. To illustrate the usefulness of the platform architecture a driving scenario has been presented and the preliminary results that can be obtained by the simulation have been shown.

The advent of promising technologies will increase the need for flexible R&D tools for such complex systems to be evaluated. Furthermore, there is an encouraging potential on this proposed approach able to bring together automotive and transportation research communities to work together on the development of Future Urban Transport Systems.

10.6 Acknowledgment

This project has been partially supported by FCT (Fundação para a Ciência e a Tecnologia), the Portuguese Agency for R&D, under grants SFRH/BD/67202/2009 and SFRH/BD/51256/2010.

10.7 References

[1] David Schrank and Tim Lomax. The 2009 URBAN MOBILITY REPORT. Technical report, Texas Transportation Institute, The Texas A&MUniversity System, 2009.

[2] R. Arnott, T. Rave, and R. Schöb. Alleviating urban traffic congestion. MIT Press Books, 1, 2007.

[3] Benjamin K. Sovacool. A transition to plug-in hybrid electric vehicles (PHEVs): why public health professionals must care. Journal of epidemiology and community health, 64(3):185–7, March 2010.

[4] N. Unger, T. C. Bond, J. S. Wang, D. M. Koch, S. Menon, D. T. Shindell, and S. Bauer. Attribution of climate forcing to economic sectors. Proceedings of the National Academy of Sciences, 107(8):3382–3387, 2010.

11 Driver's Attitude and its Influence on the Energy Waste of Electric Buses

Deborah Perrotta; José Luiz Macedo, Rosaldo J. F. Rossetti, Zafeiris Kokkinogenis;
Laboratório de Inteligência Artificial e Ciência de Computadores Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, Portugal
João Luiz Afonso and Deborah Perrotta;
Centro Algoritmi, Universidade do Minho, Portugal
Bernardo Ribeiro;
CEIIA - Centro para a Excelência e Inovação na Indústria Automóvel, Portugal

11.1 Abstract

The objective of this paper is to analyze the influence of different driver behaviors on the energy consumption of electric buses. It shows that risk-taking attitudes on traffic are not only dangerous to the driver, to the bus users and to the surroundings, but also promotes a poorer performance of the vehicle itself, increasing its energy consumption and reducing the amount of energy that can be recovered on regenerative braking.

Keywords: driver attitude; electric bus energy consumption; electric bus performance; risk-taking behavior.

11.2 Introduction

Electric vehicles are seen as one of the key players to address the issue of global warming through its operation with zero tailpipe emissions and energy efficiency improvement. In the contemporary world, many countries are working on alternatives to replace internal combustion engine vehicles. For instance, in Europe, decision N° 406/2009/EC of the European Parliament and of the Council of 23 April 2009 asks for the effort of the Member States on the reduction of greenhouse gases emissions by 20% until 2020 [1].

The 2007 Intergovernmental Panel on Climate Change report concluded that greenhouse gas emissions must be reduced by 50% to 85% by 2050 as an attempt to avoid many of the worst impacts of climate change. Reducing greenhouse gas emissions from transportation will likely require a broad range of strategies, such as increasing vehicle efficiency and reducing vehicle kilometers of travel. Public transportation can be one part of the solution [2], more specifically electric buses, which are quieter than regular buses, therefore promoting a better experience to public transport users, and also does not require much investment on infrastructure, as subways and trolleys for instance.

Nowadays, there are some electric buses in operation in some parts of the world and one of the main concerns is their high weight, which is mainly due to the amount of batteries they carry in order to have an adequate operation range. Nevertheless, it is crucial to investigate ways of optimizing an electric bus operation on the urban environment, promoting an efficient performance. This could lead in the future to a reduction of the bus weight, once it would require less energy to travel the same amount of kilometers, and thus less batteries.

When talking about the operation efficiency of internal combustion vehicles, there is one factor that plays an important role on the fuel consumption: the driver's behavior. However, at this point it is relevant to define what characterizes an aggressive driving. It is known that there are three aspects of driving behavior that have been labeled as aggressive in the driving literature: (a) intentional acts of physical, verbal, or gestured aggression; (b) negative emotions (e.g. anger) while driving; and (c) risk taking [3]. For this paper, the focus will be on the third category, namely risk taking.

The risk-taking category includes behaviors such as speeding, running red lights, weaving through traffic, maneuvering without signaling, and frequent lane changing; it also comprises the dangers from lapses of attention while driving, typical for those who use the cell phone, eat, drink, smoke, or adjust a car stereo. Any of these behaviors may occur without the presence of negative emotion or intent to harm [4].

Generally, this type of behavior has implications on fuel consumption (for internal combustion vehicles) or energy consumption (for electric vehicles), once it is characterized for the opposite behavior of the so-called eco(logical)-driving. An ecological strategy is to anticipate what is happening ahead, and drive in such a way so as to minimize acceleration and braking, cruising at the optimal speed, and to maximize coasting time at stops [5]. In other words, it may be represented by a soft driving, especially restricting acceleration rates [6].

In this paper, it is intended to analyze the implications of risk taking attitudes of a driver towards the energy consumption of electric buses when compared to regular drivers. In order to do that, a mathematical model for an electric bus powertrain was developed and further implemented on the simulation program *Simulink* (a simulation package of the *Matlab* environment). This model was further integrated to the traffic simulator *SUMO*, allowing the bus simulated on *Simulin*k to perform a route designed on *SUMO* that later gives feedback to *Simulink* in order to perform the calculations.

11.3 Methodology

An integrated simulation platform was developed in order to access the possible energy saving of a regular driver in comparison to a risk-taking one. This platform is characterized by the integration of two different simulators: a nanoscopic simulator represented by *Simulink* (*Matlab* environment), which accounts for the representation of the powertrain behavior, and a microscopic simulator represented by *SUMO*, which accounts for the routes that the bus is supposed to perform, the interactions of this bus with other vehicles, and the stops at bus stops and traffic lights.

The *Simulink* model calculates the torque needed to operate the vehicle at each moment, considering aerodynamic drag, tire rolling resistance, climbing grade and vehicle inertia effects [7]. The required total torque needed to drive the wheels at each moment at a given speed is converted to power, and further converted to energy, which is the energy required to perform the route designed at *SUMO* [8], [9]. This model also calculates the amount of energy that is wasted when the vehicle brakes and that could be further recovered through regenerative braking [10], as long as the analyzed electric bus powertrain has this ability.

As mentioned before, *SUMO* plays an important and central role on the calculations performed by *Simulink*. It allows the microscopic interactions between the bus and the other cars, making it possible to represent the dynamic behavior of the driver regarding acceleration and braking. Furthermore, it also allows the modeling of any desired route, including bus stops, which are crucial for the calculations.

A more simple approach was handled in order to facilitate the results analysis. Basically, the parameter used to differentiate the risk taking driver from the regular one is the acceleration rate, either on acceleration episodes or negative braking, behaviors that do not characterize a smooth drive [6]. Risk-taking drivers have a higher acceleration rate in both cases, and this parameter is crucial for the calculations performed on Simulink [7]. The route takes into consideration the traffic lights, the bus stops and the interaction with other vehicles, in order to have more realistic results.

11.4 Simulation Test-Bed

A random route was taken and modeled, considering as base reference the region of Aliados, in the city of Oporto, Portugal. On this route, some bus stops were defined and it was assumed that the bus would stop 20 seconds at each one of them. Stops at red traffic lights were also considered. In Figure 11-1 it can be observed that region of Aliados and a zoom-in on a certain area, where the large green rectangles are bus stops and the smaller yellow one is the bus, stopped in front of the bus stop (more specifically on the left avenue).

For calculation purposes, it was assumed that the absolute acceleration rates for the risk-taking driver are twice as the ones from regular drivers and that the deceleration rate is nearly 50% higher (Table 11-1). It is relevant to point out that none of these values surpass the technical restrictions of the electric bus.

Table 11-1: Acceleration/ Deceleration rates for simulation.

Driver’s profile	Acceleration (m/s^2)	Deceleration (m/s^2)
Regular	2.6	4.5
Risk taking	5.2	6.5

The simulation was then performed twice for exactly same route: one using regular acceleration rates, based on the normal operation of the bus, and another for the “aggressive” behavior, which for this paper is characterized by higher values of acceleration rates and faster braking.

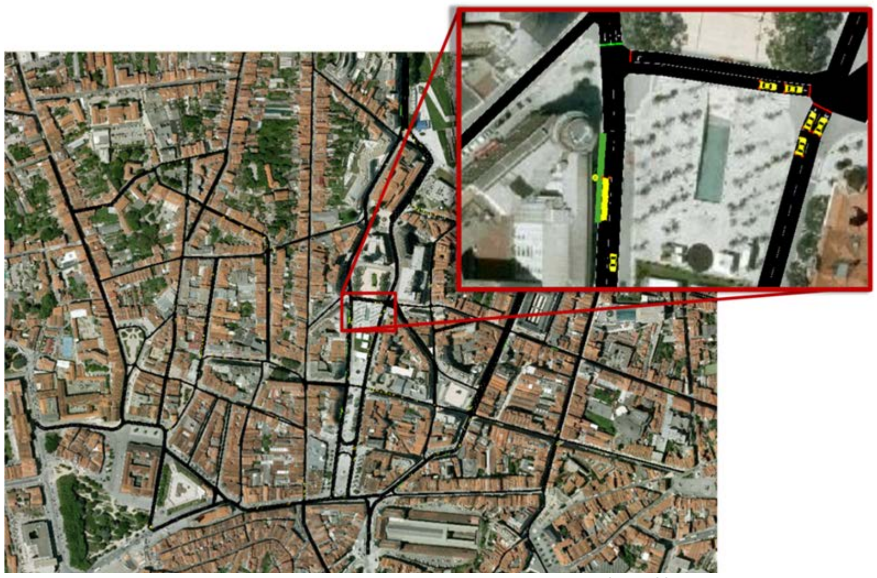


Figure 11-1: Aliados region and detailed representation of traffic on *SUMO* simulator.

11.5 Results

The first analyzed parameter was the amount of energy spent to perform the route. In Figure 11-2 it can be seen that the aggressive driver spent 17% more energy (0.100 kWh) than the regular driver (0.087 kWh) to complete exactly same route.

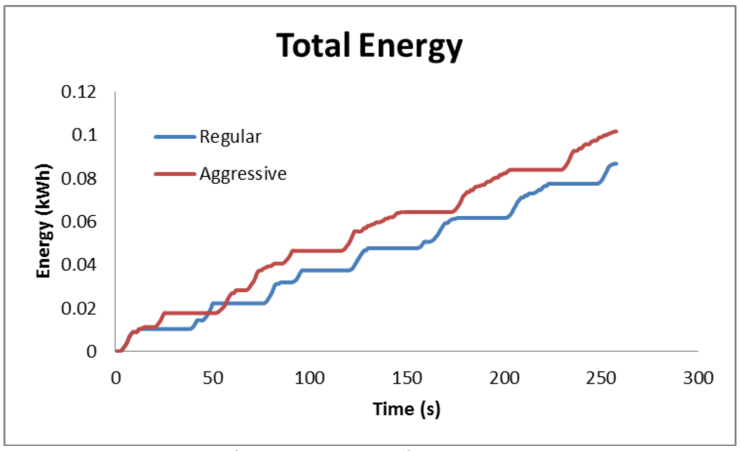


Figure 11-2: Comparative plot of the amount of energy spent by each electric bus driver.

The second analyzed parameter was the amount of energy that is wasted during the braking episodes on the resistance forces to the movement, namely resistance to the air, resistance to the ground and inertia. All these forces depend significantly on the acceleration rate, thus it is expected some discrepancy between the outcomes from both drivers.

In Figure 11-3, it can be observed that the aggressive driver wastes 0.014 kWh in all braking episodes of this route, while the regular driver spends half of this amount: 0.007 kWh. This parameter is extremely important to calculate the potential to recover energy on braking episodes. Having a risk taking attitude contributes negatively to this recovery, once this generates a higher boost of energy in such a small period of time (i.e., a braking episode), that batteries would struggle to absorb, due to their low power density [11].

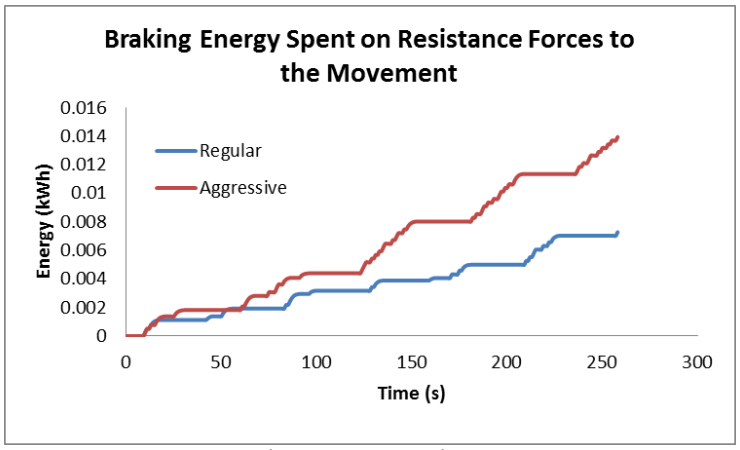


Figure 11-3: Comparative graph of the amount of energy wasted on braking episodes.

11.6 Conclusions and Future Work

In conclusion, with this work it became clear how badly can driver’s attitude affect the energy consumption of an electric bus. Not only is risk-taking behavior dangerous for the driver and the others but it also contributes negatively to an efficient driving performance. In this paper, the case of an electric bus was analyzed and it could be observed that a simple risk-taking attitude, such as accelerating or braking more intensely, has a huge influence on the bus performance, either regarding the amount of energy spent to complete a certain cycle, or regarding the potential of energy recovery from the regenerative braking episodes.

Future work intends to investigate deeper into this matter by the creation of simulation agents, which will mimic the human reaction to adverse traffic conditions, thus allowing a more detailed analysis.

11.7 Acknowledgment

This work is financed by FEDER Funds, through the Operational Programme for Competitiveness Factors – COMPETE under the project 13844. Deborah Perrotta thanks FCT (Fundação para a Ciência e Tecnologia) for PhD scholarship (SFRH/BD/51256/2010) under the scope of the MIT Portugal Program in Engineering Design and Advanced Manufacturing – Leaders for Technical Industries focus area.

11.8 References

[1] "Summaries for EU Lesgislation." [Online]. Available: http://europa.eu/legislation_summaries/energy/european_energy_policy/en0008_en.htm.

[2] Federal Transit Administration (FTA), "Public Transportation ' s Role in Responding to Climate Change," 2010.

[3] C. S. Dula and M. E. Ballard, "Development and Evaluation of a Measure of Dangerous, Aggressive, Negative Emotional, and Risky Driving1," Journal of Applied Social Psychology, vol. 33, no. 2, pp. 263–282, Feb. 2003.

[4] C. S. Dula and E. S. Geller, "Risky, aggressive, or emotional driving: Addressing the need for consistent communication in research," Journal of Safety Research, vol. 34, no. 5, pp. 559–566, Jan. 2003.

[5] M. Kamal, M. Mukai, J. Murata, and T. Kawabe, "Development of Ecological Driving Assist System Model Predictive Approach in Vehicle Control," in 16th ITS World Congress and Exhibition on Intelligent Transport Systems and Services, 2009, pp. 1–11.

[6] M. Miyatake, "Theoretical Study on Eco-Driving Technique for an Electric Vehicle Considering Traffic Signals," in Power Electronics and Drive 2011 IEEE Ninth International Conference on Systems (PEDS), 2011, no. December, pp. 5–8.

[7] D. Perrotta, A. Teixeira, H. Silva, and B. Ribeiro, "Electrical Bus Performance Modeling for Urban Environments," in SAE 2012 World Congress, 2012.

[8] J. Larminie and J. Lowry, Electric Vehicle Technology Explained. John Wiley & Sons Ltd, 2003, p. 62; 183–186.

[9] M. Ehsani, K. M. Rahman, and H. a. Toliyat, "Propulsion system design of electric and hybrid vehicles," IEEE Transactions on Industrial Electronics, vol. 44, no. 1, pp. 19–27, 1997.

[10] S. R. Cikanek and K. E. Bailey, "Regenerative braking system for a hybrid electric vehicle," in Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301), 2002, pp. 3129–3134.

[11] A. Chu and P. Braatz, "Comparison of commercial supercapacitors and high-power lithium-ion batteries for power-assist applications in hybrid electric vehicles I. Initial characterization," Journal of Power Sources, vol. 112, no. 1, pp. 236–246, Oct. 2002.

12 Sumo as a Service – Building up a Web service to interact with SUMO

Mario Krumnow
Dresden University of Technology, Chair of Traffic Control Systems and Process Automation, Dresden,Germany

12.1 Abstract

To interact with a simulation at runtime is often demanded within the scope of research studies. Therefore the microscopic traffic simulation software SUMO has a real-time I/O data interface (TraCI). This interface offers the possibility for a bidirectional communication between the user application and the simulation. Only few software implementations can handle that protocol, one of them is the Python TraCI Client being included into the simulation suite. Though depending on the preferred programming language, a specific implementation is needed. A solution designed to cover the problem is a standardized protocol interacting with SUMO, like the simple object access protocol (SOAP). This protocol can easily be integrated into a lot of programming languages, and is implemented as part of a Web service. This solution offers a lot of opportunities e.g. an unlimited number of clients.

Keywords: Microscopic Traffic Simulation, Web service, SOAP, WSDL

12.2 Motivation

The main reason for building a Web service is to make the communication with the Traffic Control Interface (TraCI) more comfortable, even for people who are not familiar with Python.

Due to a clear and user-friendly communication the SUMO user will be able to focus on the analysis of traffic and transportation data. Furthermore the Web service is platform-independent, which permits the user to utilize various kinds of programming languages. The Web service handles multiple connections simultaneously. This is a base requirement for using the micro simulation as a service in the field of traffic management centers. Therefore it makes no difference where the application is located or which programming language has been used to interact with the simulation.

12.3 Implementation of the Web service

12.3.1 Integration of TraCI

As a first implementation the programming language Java has been used to interact with SUMO, because Java is platform independent and very robust due to the embedded Exception Handling concept.

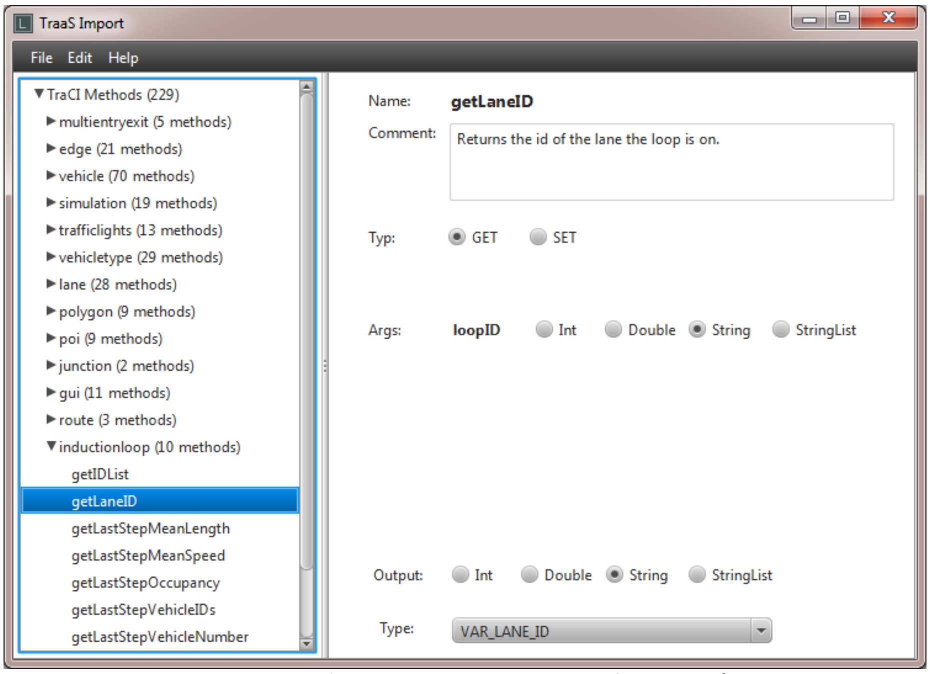
A project named traci4j[6] already implements a TraCI Handler that generates Byte Messages for the requests and responses for Java. So these parts of that code provides the base for a new Web service which is called TraaS (TraCI as a service).

In traci4j only a couple of TraCI functions has been implemented which show the main functionality of the written code. In order to offer as much functions as possible the software needs to be extended. The support of the subscription feature is not yet available.

A software tool has been developed which reads and interprets the available python code of the default TraCI client. The code is located in the installation package of SUMO. All recognized functions as well as the available comments have been stored in a single XML[7] File.

By then a determination in getter and setter methods has already been done. This means the functions have been marked with a couple of XML attributes. That import process is semi-automated due to the fact that the original Python source code does not follow a standardized schema.

In order to support this task a basic java software tool has been developed, which has the facility to show and edit the information of the XML within a graphical user interface (GUI). This tool uses the external library dom4j[8] for the XML parsing and JavaFX for the GUI part (Figure 12-1).



Name: **getLaneID**

Comment: Returns the id of the lane the loop is on.

Typ: [✓ GET] SET

Args: **loopID** Int Double [✓ String] StringList

Output: Int Double [✓ String] StringList

Type: VAR_LANE_ID

Figure 12-1: A graphical java tool to import the TraCI functions

After validating and editing the XML file over 200 methods have been named and described. The data type for each argument or returning value has been defined.

Subsequently another java program generates the required java classes by interpreting the XML file. This is an important part because files for the TraCI client as well as for the Java Web service are being created at the same time.

[6] http://traci4j.sourceforge.net/

[7] Extensible Markup Language

[8] http://dom4j.sourceforge.net/

To work as a Web service every value has to be serializable to be embedded in a SOAP Message. There is no problem with primitive data types like double or integer which work by default. For other data types like the SUMO StringList or the compound structures some new Java classes has been generated. As a result the class SumoStringlist is an implementation of the Java List Object[9]. Some more classes have been generated in order to represent all necessary values of the appropriate compounds within the Python code.

The Java classes have the same function names as those used in the original Python code. Additionally the order of the arguments has been preserved. Usually Java methods can be distinguished into two types: on the one hand the getter methods with a returning value and on the other hand setter methods without any returning value. All TraCI functions have been dedicated in one of those two issues.

In both cases the TraCI request message is constructed by a class called SumoCommand. Depending on the occurrence of a returning value the method is a getter or setter method.

For every command block an extra class has been generated. As a result there are at least thirteen different classes (Figure 12-1).

It has been necessary to change several parts of the original traci4j code to support all these new data types. Furthermore some unnecessary parts were removed to simplify the application. At the end of that process the developed software does not have any external references like log4j[10] or similar dependencies.

The way the TraCI functions are imported is going to be improved in the future. The aim is to make the order of the arguments changeable. Moreover Test cases should be generated by default within the importing process.

12.3.2 Building the Web service

To work as a Web service some java classes need to be added to offer the endpoint communication. This means that the service binds on a specific IP address and on a specific port. Consequently it is possible to have multiple Web server instances on a single server by using different ports. In Figure 12-2 the concept of the Web service is shown.

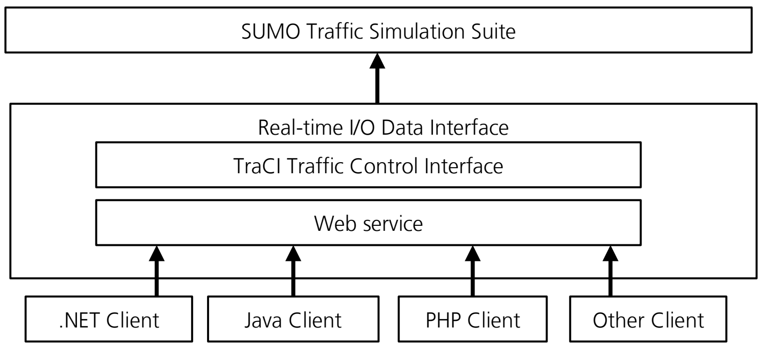

SUMO Traffic Simulation Suite
Real-time I/O Data Interface
TraCI Traffic Control Interface
Web service
.NET Client
Java Client
PHP Client
Other Client

Figure 12-2: Concept of the Web Service

To offer all the methods of the java TraCI client to the Web service another java class has been generated by the help of a script. All the public web methods have been marked with a special annotation. This is necessary to make the methods available for the Web service Client. For each method a prefix has been added to ease distinguishing between different

[9] Part of the java collection framework

[10] http://logging.apache.org/log4j/2.x/

command block functions by origin. Every method belonging to a vehicle, gets the prefix "Vehicle_" . For example the method for changing the lane of a vehicle is named Vehicle_changeLane(arg1, arg2).

By using the SOAP protocol the communication is based on XML messages, so the request and the response messages are encoded in XML. The bandwidth is much higher than the TraCI Byte communication because of the large overhead of XML. In Figure 12-3 an example of such an XML response message for a lane shape request is shown.

The relatively huge bandwidth caused by the SOAP messages is a disadvantage. However considering that the bandwidth will not be the bottle neck in times of gigabit Ethernet networks this approach is sufficient. A comparison between the TraCI Byte code communication and the corresponding SOAP communication would be interesting, especially regarding to the emerging latency times.

```
<?xml version="1.0"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:Lane_getShapeResponse xmlns:ns2="http://ws.tudresden.de/">
            <return>
                <coords>
                    <x>160.68</x>
                    <y>50.17</y>
                </coords>
                <coords>
                    <x>52.89</x>
                    <y>49.34</y>
                </coords>
            </return>
        </ns2:Lane_getShapeResponse>
    </S:Body>
</S:Envelope>
```

Figure 12-3: SOAP Message for a shape response

12.4 Performance of the Web service

In a first research a comparison has been made between the runtime using the original Python API and the TraaS Application. It is possible to use TraaS as a Java library, which can easily be integrated into a user Java application. In that case the main task is encoding and decoding of the TraCI Byte messages. This is equivalent to the SUMO Python Client. High loads of network packages will not appear. While running TraaS as a standalone Web service, the application handles all client connections. Moreover the encoding and decoding of the SOAP messages has to be done. Obviously the runtime increases because of the additional XML handling being used by the SOAP protocol.

Three different ways have been compared to interact with SUMO. Therefore a standard simulation case, inserting vehicles and increasing the simulation time, has been chosen. Firstly a Python Client has been written using the defaults Python API (Figure 12-4). Secondly TraaS has been applied as a referenced library for a user Java application (Figure 12-5). Thirdly a Java application, using TraaS as a standalone Web service, has been built. The last one is quite similar to the code snippet shown in Figure 12-5. The command line options have been equal for the three examples.

```
for i in range(duration):
    traci.simulationStep()
    traci.vehicle.add('v'+str(i), 'route', 0, 0, 13.8, 0, 'car')
```

Figure 12-4: Python example for the benchmark

```
for(int i=0; i<duration; i++){
    conn.do_timestep(1);
    conn.do_job_set(Vehicle.add("v"+i, "car", "route", 0, 0, 13.8, 0));
}
```

Figure 12-5: Java example for the benchmark

The 64 BIT version of the SUMO application has been used for that benchmark. The applications have been executed on an Intel® Core i3 Processor with an amount of 4 GB of physical RAM available. By variation of the duration value the results in Table 12-1 have been reached.

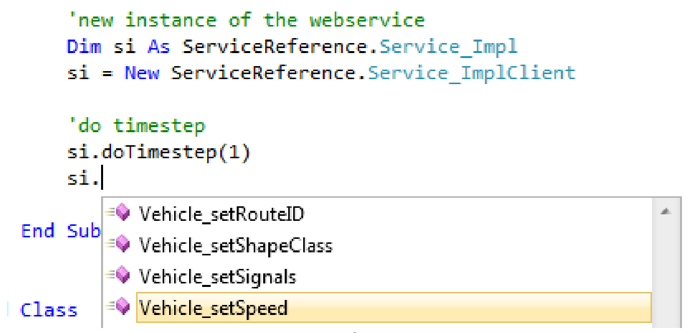
Table 12-1: Comparison of the different runtimes of the different applications.

Duration	**Python Code**	**TraaS as library**	**TraaS as web service**
500	1301 ms	1620 ms	3556 ms
1000	4803 ms	5300 ms	8640 ms
2000	19023 ms	20012 ms	25196 ms
5000	123174 ms	119071 ms	131590 ms
10000	481723 ms	470295 ms	509831 ms

The assumption that the runtime increases by using the Web service has been assured. This is important especially for simulations with small networks and a small traffic volume because the simulation takes significantly more time. In huge simulation scenarios the runtime of SUMO itself dominates the overall runtime. The time for communication is negligible short. As an interesting fact the runtime of TraaS used as a library is even shorter than the one of the originally Python example. To verify this assumption some more benchmarks need to be built.

12.5 Using the Web service

The Web service implements a start function ensuring SUMO to be launched with a defined configuration file and listening on a specified port. Furthermore the different clients are able to connect to the Web service and have access to all of the getter and setter methods. For real time simulations a timer application is required, which does a simulation step. Building up new clients is a straight forward process because ready-made tools like the wsimport binary in Java. The build-in tools in MS Visual Studio for the .NET programming languages are used to integrate the Web service. The integration in Visual Studio is shown in Figure 12-6.



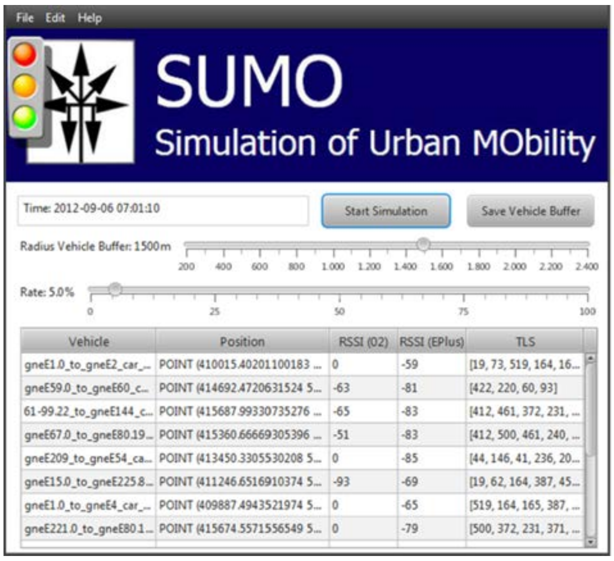
```
'new instance of the webservice
Dim si As ServiceReference.Service_Impl
si = New ServiceReference.Service_ImplClient

'do timestep
si.doTimestep(1)
si.
```

Figure 12-6: Example of a VB.NET Application

The Web service has already been used successfully by students and research engineers at Dresden University of Technology. Various scientific studies have shown the practical significance of the Web service and have helped to develop and improve its features. [4][5][6]

An example of a student research project [6] is shown in Figure 12-7. This work used TraaS as a library in order to interact with SUMO. A user-defined library has been used in front of the Web service to handle some default tasks, like the import and export of different data sources. A graphical user interface named SumoInteractive has been developed in Java. It offers the possibility to manipulate some vehicles in the simulation. By changing the values of some sliders the related parameters of the simulation are affected directly.



Time: 2012-09-06 07:01:10

Radius Vehicle Buffer: 1500m

Rate: 5.0%

Vehicle	Position	RSSI (02)	RSSI (EPlus)	TLS
gneE1.0_to_gneE2_car_...	POINT (410015.40201100183 ...	0	-59	[19, 73, 519, 164, 16...
gneE59.0_to_gneE60_c...	POINT (414692.4720631524 5...	-63	-81	[422, 220, 60, 93]
61-99.22_to_gneE144_c...	POINT (415687.99330735276 ...	-65	-83	[412, 461, 372, 231, ...
gneE67.0_to_gneE80.19...	POINT (415360.66669305396 ...	-51	-83	[412, 500, 461, 240, ...
gneE209_to_gneE54_ca...	POINT (413450.3305530208 5...	0	-85	[44, 146, 41, 236, 20...
gneE15.0_to_gneE225.8...	POINT (411246.6516910374 5...	-93	-69	[19, 62, 164, 387, 45...
gneE1.0_to_gneE4_car_...	POINT (409887.4943521974 5...	0	-65	[519, 164, 165, 387, ...
gneE221.0_to_gneE80.1...	POINT (415674.5571556549 5...	0	-79	[500, 372, 231, 371, ...

Figure 12-7: Example of a Client application

Another pending student work tries to implement the TraCI methods, offered by the Web service, into Mathworks® Matlab. These functions are available as basic toolbox functions, which can be easily integrated by drag and drop into the user project.

Consequently more detailed models of single vehicles become available. The accurate behavior of vehicles does not only depend on the car following model and the change lane model. In addition the specific attributes of a chosen engine can influence the target trajectories. Matlab is used for computing the corresponding gear and revolutions per minute. Hence the vehicles in the simulation adapt their velocity based on that computed Matlab model.

In the near future the Matlab Toolboxes should be generated automatically during the import process to have all the TraCI functions available in Matlab as well.

12.6 Conclusions

The use of a Web service in order to communicate with SUMO is very beneficial especially for educational purposes. Users can freely choose their preferred programming language and interact with the traffic simulation immediately. TraaS provides simple access to SUMO especially for users who are not familiar with Python.

Obviously the runtime of TraCI commands is longer while using the Web service. However this additional time is of less consequence in large simulation, because the runtime is determined mainly by the traffic volume.

A final target could be the integration of the Web service into the SUMO source code. With some new command line options the web service should be configured and activated.

In summer 2013 the project TraaS will become OpenSource under a General Public License (GPL) and will be hosted at SourceForge[11].

12.7 References

[1] SUMO – Simulation of Urban Mobility (2012). Institute of Transportation Systems, German Aerospace Center, Germany. http://sumo.sf.net. Accessed Aug. 29, 2012.

[2] Microscopic real-time simulation of Dresden using data from the traffic management system VAMOS, Proceedings of the 19th ITS World Congress Vienna, Austria, Oct.25, 2012

[3] Schaltzeitprognose verkehrsadaptiver Lichtsignalanlagen im Rahmen des Projektes EFA 2014/ 2, Mario Krumnow, VIMOS, Nov. 29.11.2012

[4] Arlt, A. (2012). Realitätsnahe Simulation des Verkehrsflusses auf der Süd-West-Umfahrung in Dresden mit SUMO unter Berücksichtigung des MIV und des ÖPNV (in German). Report student research project, Dresden University of Technology, Germany.

[5] Reiche, M. (2012). Vorher-Nachher-Analyse der Emissionsbelastung am Knotenpunkt Nürnberger Platz im Rahmen der ÖPNV-Bevorrechtigung des NSV-Projektes (in German). Report student research project, Dresden University of Technology, Germany.

[6] Wießner, E (2013). Analyse der Datenmengen der Car-2-Infrastructure Kommunikation durch realitätsnahe Simulation unterschiedlicher Testszenarien (in German). Report student research project, Dresden University of Technology, Germany.

[11] http://sourceforge.net/

[7] Krimmling, J., Franke, R., Engelmann, R., Körner, M. (2011). Erfahrungen mit dem vollautomatischen baulastträgerübergreifenden Betrieb der Dynamischen Wegweisungskomponente im Operativen Straßenverkehrsmanagementsystem VAMOS (in German). In Proceedings HEUREKA 2011 – Optimierung in Verkehr und Transport, Forschungsgesellschaft für Straßen- und Verkehrswesen .e.V., Cologne, Germany.

13 SUMOPy: an advanced simulation suite for SUMO

Joerg Schweizer;
University of Bologna, DICAM-transportation group, Bologna, Italy

13.1 Abstract

SUMOPy is intended to (1) expand the user-base of SUMO - Simulation of Urban MObility: by providing a user-friendly, yet flexible simulation suite for SUMO, including a GUI and scripting; (2) enhance the demand modeling capacity, in particular by providing an activity-based demand modeling; (3) easier management of existing vehicle types such as bicycles as well as new vehicle types like Personal Rapid Transit (PRT). The vision of SUMOPy is to simulate a synthetic population in a multi-modal transport environment, because it is believed that only in this way the net-effect of new transport modes and technologies can be assessed. The basic architecture and principles of operation are explained and some examples on how to use SUMOPy through both, GUI and scripting are presented. SUMOPy is still in an early stage of development and many of the current SUMO capabilities are not yet implemented. However this article explains which features will be added in the near future and how they fit in the existing structure.

Keywords: SUMO, Python, micro-simulation, synthetic population, PRT, GRT.

13.2 Introduction

SUMO - Simulation of Urban Mobility [1], rapidly developed into a flexible and powerful open-source micro-simulator for multi-modal urban traffic networks [2]. The features and the number of tools provided are constantly increasing, making simulations ever more realistic. At the present state, the different functionalities consist of a large number of binaries and scripts that act upon a large number of files, containing information on the network, the vehicles, districts, trips routes, configurations, and many other parameters. Scripts (mostly written in Python), binaries and data files exist in a dispersed manner. The flow of data processing is illustrated in Figure 13-1 (a).

In practice, a master script is necessary to hold all processes and data together in order run a simulation of a specific scenario in a controlled way. This approach is extremely flexible, but it can become very time consuming and error prone to find the various tools, combine their input and output and generate the various configuration files. Furthermore, it reduces the user-base of SUMO to those familiar with scripting and command line interfaces. Instead, SUMO has the potential to become a multi-disciplinary simulation platform if it becomes more accessible to disciplines and competences outside the field of information technologies.

This problem has been recognized and different graphical user interfaces have been developed. The traffic modeler (also named traffic generator) is a tool written in Java which helps to manage files, configure simulations and to evaluate and visualize results. SUMOPy is written entirely in the object-oriented script language Python, it uses wxWindows as GUI interface and NumPy for fast numerical array-type calculations. SUMOPy is similar to the traffic generator in that it simplifies the use of SUMO through a GUI. But SUMOPy is more than just a GUI, it is a suite that allows to access SUMO tools and binaries in a simple unified fashion. The distinguishing features are:

- SUMOPy has Python instances that can make direct use of most functions tools already available as Python code. This avoids time consuming writing and parsing after each step, as illustrated in Figure 13-1(b). However, intermediate results are still obtained by writing and parsing files, as shown in Sec. 13.3.1.
- SUMOPy has a Python command line interface that allows direct and interactive manipulation of SUMOPy instances.
- SUMOPy provides a library that greatly simplifies the scripting.

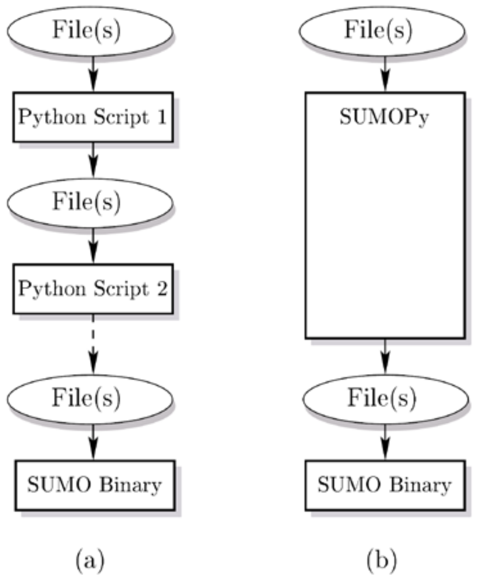


Figure 13-1: Illustration of data flow in a typical simulation, (a) with current SUMO distribution, (b) the *idealized* flow with SUMOPy extension.

The paper is organized as follows: Section 13.3 is a brief summary of the principles of operation, where data are stored, which functions are available; also included is the (current) SUMOPy file tree; Section 13.4 explains the GUI and Section 13.5 gives some brief scripting examples. Section 13.6 draws preliminary conclusions of the current SUMOPy development.

13.3 Principles of operation

The principles of operation of SUMOPy is that functions, processes or methods act upon a central data structure, called the *Scenario*, that holds together all necessary information to prepare and generate data to run a micro-simulation and to process its results (see Figure 13-2). Dependent on the specific study, the user must provide certain input data to the scenario, other input data can be generated and completed by applying functions and methods. The scenario structure is shown below (Sec. 13.3.1) and the main functions are briefly explained in the following Sec. 13.3.2, while the directories and files of SUMOPy are explained in Sec. 13.3.3. Both, scenario data and functions can be accessed or executed via GUI (see Sec. 13.4) or scripting (see Sec. 13.5).

13.3.1 The scenario data structure

The main data elements of the scenario are shown in Figure 13-2. The parts in gray are planned and not implemented at the time of writing this article. The main instances of the scenario are: net, demand, vehicle types and results. A major future extension will be a land-use structure which allows to generate a synthetic population.

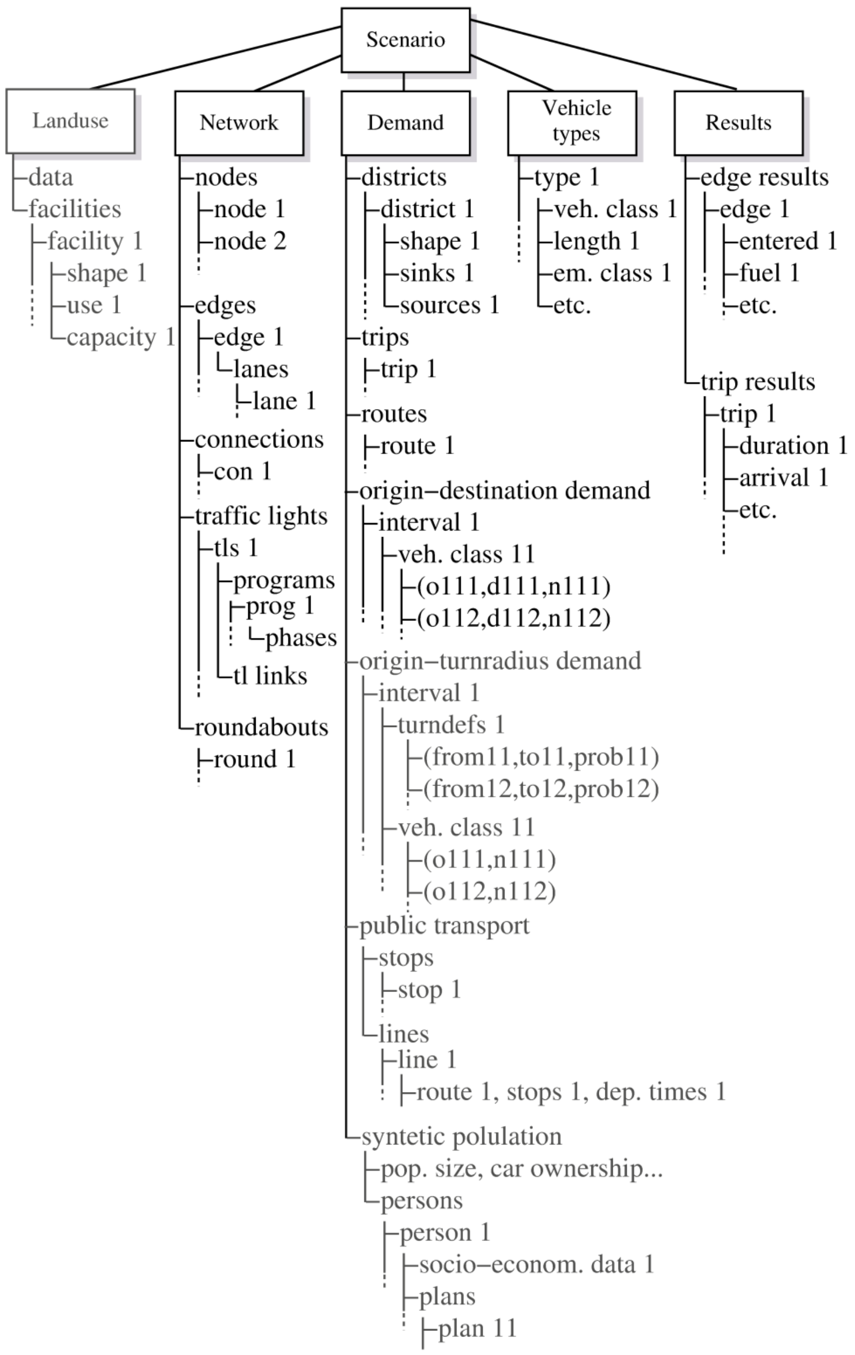


Figure 13-2: Data structure of the scenario, the central instance in SUMOPy, truncated at irrelevant branches. Demand structure is shown in more detail. Gray parts are future extensions.

Most of the data in Figure 13-2 are instances (except python primitives), that provide specific attributes and basic methods to add, get, set and delete the data they contain.

- *Network*: the network containing edges, lanes, nodes/junctions, connections, traffic-lights and roundabouts. The network instance is compatible with the network instance provided with the SUMO Toolbox, but has a many enhancements. The network can be read and written into the native SUMO net file.

- *Demand*: the demand instance holds all data and methods necessary to generate trips and routes. The central idea is that the trips, generated from different generation models (example: OD demand, turn-ratio demand, random trip generators and public transport lines) can be superimposed in a flexible manner.
- Currently only simple Origin-To-Destination matrices (ODMs) between districts for specific time intervals and vehicle classes is implemented. This data can be stored in the ODM files using XML format. Districts are defined in the traffic assignment zones (TAZ), which is editable through the GUI.
- *Vehicle types*: currently the vehicles instance represents only a table with vehicle types and their specific parameters, which are used in the scenario. Parameters can be edited through the GUI and saved in a XML file.
- *Results*: instance contains all output data from the simulation and provides methods to read and collect them. There are currently 2 structures implemented: edge oriented data and trip-oriented data. The edge data can be displayed with the GUI and examined interactively.

The planned data-structures (Figure 13-2 in grey) are:

- A land-use data structure, holding geo-referenced facilities (buildings, areas) and their respective use plus other transport relevant attributes.
- A demand instance that holds turn ratios.
- A demand instance that allows to add public transport stops and public transport lines (also through GUI).
- A demand instance that holds a virtual population. This is an important path towards activity based trip chaining in a multi-modal network simulation. The generator will make use of the land-use data structure too.
- More result types: time dependent edge and trip data.

Note that SUMOPy manages all files involved in a scenario, including naming reading and parsing whenever necessary, see Sec. 13.3.2

13.3.2 Main functions, methods and processes

Functions, methods and processes provided by the SUMOPy library can change and add data to the scenario. Methods within a main instance focus on adding/deleting and retrieving data, while keeping consistency with other data under the same structure. In addition, the scenario instance provides a large number of standard methods to import/export and transform data, including the SUMO micro-simulation. Some methods are either entirely written in Python (functions provided either by the SUMOPy library or from SUMO tools); or the methods dumps data to files, runs a SUMO binary and parses from output files back into the scenario data structure.

For more complex or long-lasting operations, a process instance is created, which lives at least for the duration of the process. The process can call other functions and checks their input and output. The process can be run as a separate threat or in the background (nohup still experimental). The currently implemented main scenario methods are:

- Import from osm: function to download and convert a network of any size from the OSM server (note that the quality of the OSM network may vary significantly). Input for this function is the bounding box or the area code. A network and polygon xml file will

be generated. This is just a convenience function for already implemented functions in the SUMO import toolbox.
- Network generator: the standard SUMO network generator can be accessed directly through python.
- Make TAZ: generates traffic assignment zones from district and edge information. Uses modified functions from SUMO district toolbox.
- Odm to trips: function to generate trips from the data contained in the district-to-district-oriented origin-destination demand data structure, entirely written in python, using some modified functions from the SUMO district toolbox.
- Dua route: runs standard SUMO router.
- Dua iterate: wrapper around the python script provided by the SUMO toolbox.

Note that the above methods are scenario class methods and called from within python. The underlying writing of configuration files and updating of internal data is taken care of by the scenario instance. Function parameters to configure a particular function can be passed as input arguments for the python method.

Future extensions include:

- Import for different network file formats (make a better use of netconvert and OD2TRIPS).
- Random trip generator (already implemented in SUMO toolbox)
- Jtrouter: to produce route files from turn-ratio data. All route generating functions could be called successively if respective demand data exists.
- TraCI Interface: runs a SUMO simulation which is controlled by SUMOPy through the Traffic Control Interface (TraCI). There are already python scripts available that will be integrated into the SUMOPy framework. Also easy to use custom control modules will be included.
- Rapid Transit (PRT) and Group Rapid Transit (GRT) control: one control module via TraCI will allow special PRT and GRT vehicle types to receive demand dependent routing. Such vehicles would stop at the stations and pick up or unload persons. If not used, they are automatically routed to another stop with demand or to a depot.

13.3.3 Directories and Files

The SUMOPy directory contains currently only the main python file to run SUMOPy in graphical interface mode: sumopy-gui.py. The other directories have the following contents:

- lib_meta contains universal class management and import/export methods that are *not* SUMO specific. The main library is called classmanager, which provides a set of Mixin classes that allow the management of a class and it attributes, including reading/saving, standard exports, labeling, naming and handling (deleting, inserting) of numpy array structures. Another important class is the process class which is a general framework to control a complex sequence of function calls, including file management.

- lib_meta_wx contains generic, *not* SUMO specific, GUI classes such as an object browser and a canvas editor, using the wxPython library (A python wrapper for wxWindows). These are mainly widgets to represent objects that inherit the class manager methods in the lib_meta.

- lib_sumopy is the central library that contains modules with the scenario and all SUMOPy compatible functions and processes. Some files are copies of modules found in the SUMO/tools directory, but with different function-call procedures. The files in more detail:

- o dualterate.py: modified function of module in SUMO/tools directory
- o edgesInDistricts.py: same functions as module in SUMO/tools directory
- o genOdm.py: all functions to generate trips from origin-destination data
- o net2.py: contains the net class and respective sub classes. The net class remains compatible with the net class in SUMO/tools. Currently this package contains also the demand and vehicle types, which will soon be relocated in separate packages.
- o osmBuild.py: same functions as module in SUMO/tools directory. There have been small modifications and extensions regarding function calls and file naming.
- o resultslib.py: contains results classes and sub classes
- o sumopylib.py: contains the scenario class and the main process classes to control sumo, duaiterate and osm download/convert commands.
- o tazTools.py: modified function of module in SUMO/tools directory.

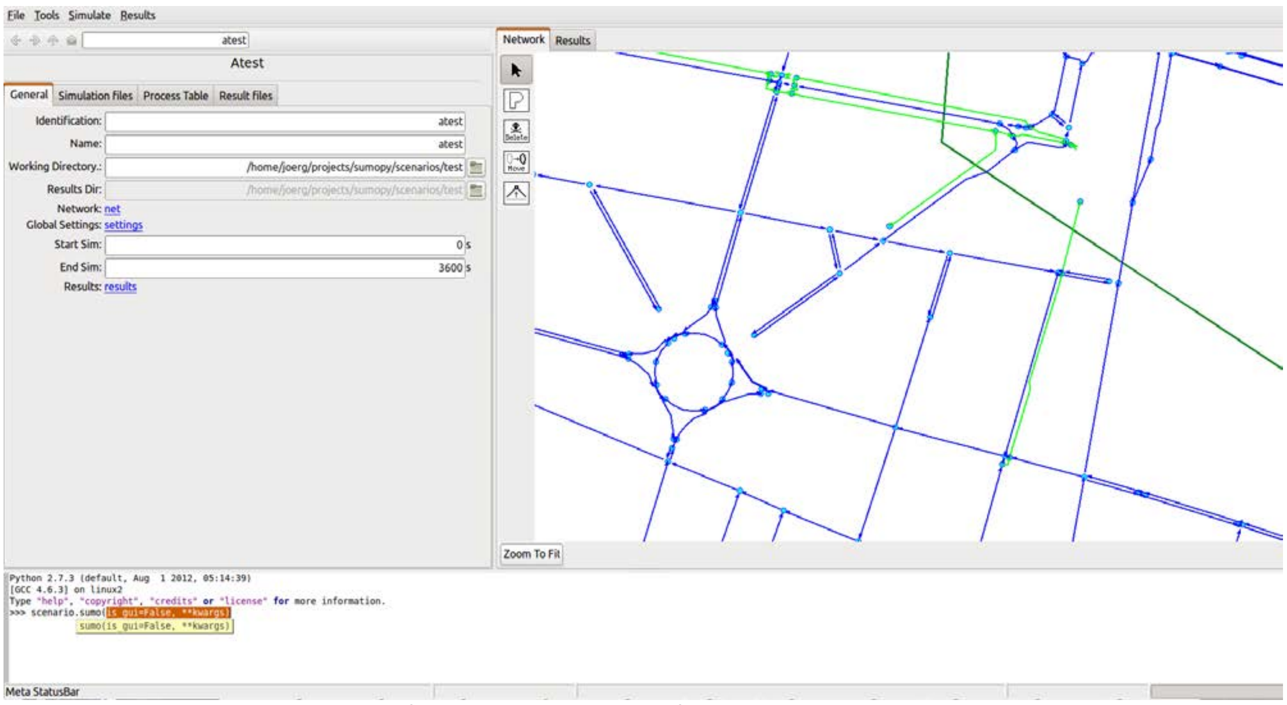
- lib_sumopy_wx: SUMOPy specific GUI classes built on the classes provided in lib_meta_wx.
    - o netview2.py: Currently the only module, which contains network and results viewing widget. Actually each visible network or demand object in the network2.py library has a “sister object” in this library, which represents its graphical appearance. These graphical sister objects are updated through event call-backs if attributes of the network object changes.

13.4 Graphical User Interface

The main window consist of three main elements: an *object browser*; *the network/results viewer* and a *command line interface (CLI)*, as shown in Figure 13-3.

The object browser is a powerful tool that allows to browse through all instances and child instances of a scenario. It is further possible to change some parameters interactively. All tables can be exported in CSV format.

The network viewer is interactive and single edges and nodes can be examined in combination with the object browser. Some edge and lane attributes, like maximum speed or access restrictions can be changed. There are currently some limited network editing functions implemented. The districts can be manually drawn and modified.



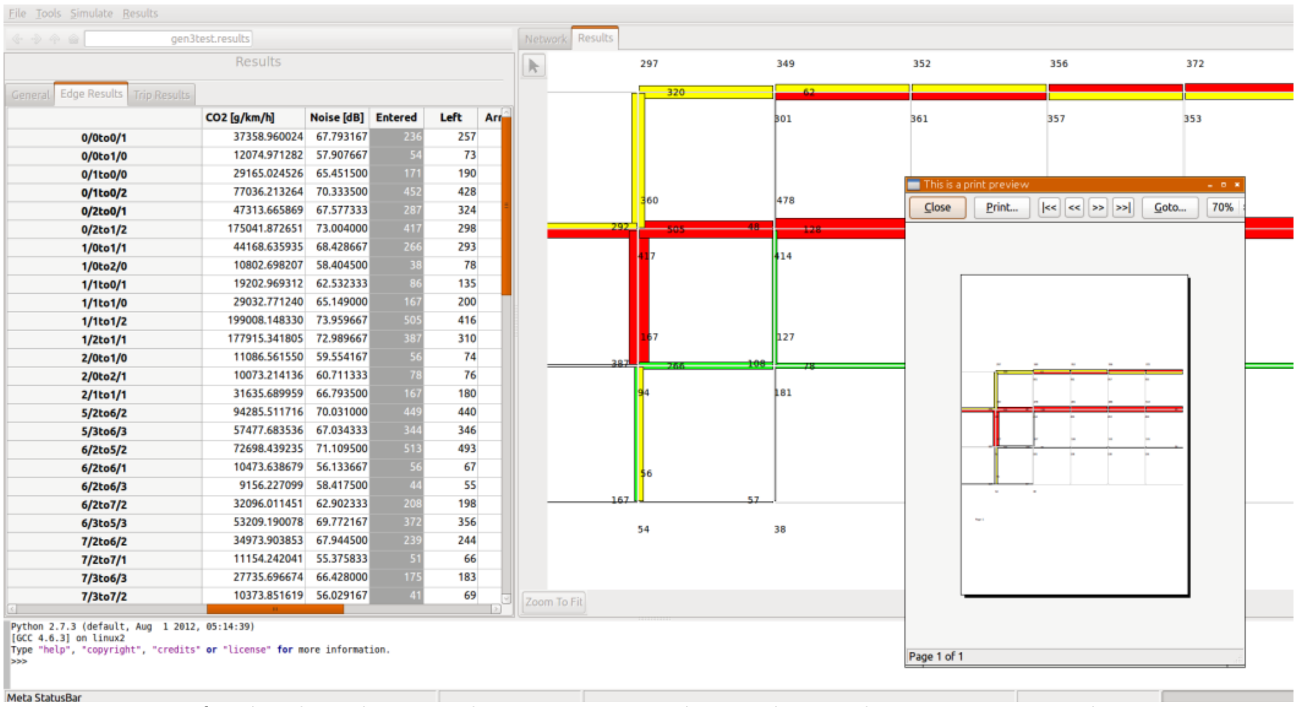
Identification: atest
Name: atest
Working Directory.: /home/joerg/projects/sumopy/scenarios/test
Results Dir: /home/joerg/projects/sumopy/scenarios/test
Network: net
Global Settings: settings
Start Sim: 0 s
End Sim: 3600 s
Results: results

```
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> scenario.sumo(is_gui=False, **kwargs)
              sumo(is_gui=False, **kwargs)
```

Figure 13-3: The main window of the SUMOPy's GUI: Left part is the scenario object browser; right part the network/results viewer where roads are in blue, bicycle lanes in green, delivery access roads in dark blue, nodes (junctions) in cyan; below in interactive command line interface.

The result fewer maps edge and trip oriented simulation results onto the network, see Figure 13-4. Edge oriented results can be examined interactively together with the object viewer.

The command line interface allows to interact with the scenario objects through Python commands. The interface includes a help for typing by prompting with a list of class methods and options to choose from.



	CO2 [g/km/h]	Noise [dB]	Entered	Left	Arr
0/0to0/1	37358.960024	67.793167	236	257	
0/0to1/0	12074.971282	57.907667	54	73	
0/1to0/0	29165.024526	65.451500	171	190	
0/1to0/2	77036.213264	70.333500	452	428	
0/2to0/1	47313.665869	67.577333	287	324	
0/2to1/2	175041.872651	73.004000	417	298	
1/0to1/1	44168.635935	68.428667	266	293	
1/0to2/0	10802.698207	58.404500	38	78	
1/1to0/1	19202.969312	62.532333	86	135	
1/1to1/0	29032.771240	65.149000	167	200	
1/1to1/2	199008.148330	73.959667	505	416	
1/2to1/1	177915.341805	72.989667	387	310	
2/0to1/0	11086.561550	59.554167	56	74	
2/0to2/1	10073.214136	60.711333	78	76	
2/1to1/1	31635.689959	66.793500	167	180	
5/2to6/2	94285.511716	70.031000	449	440	
5/3to6/3	57477.683536	67.034333	344	346	
6/2to5/2	72698.439235	71.109500	513	493	
6/2to6/1	10473.638679	56.133667	56	67	
6/2to6/3	9156.227099	58.417500	44	55	
6/2to7/2	32096.011451	62.902333	208	198	
6/3to5/3	53209.190078	69.772167	372	356	
7/2to6/2	34973.903853	67.944500	239	244	
7/2to7/1	11154.242041	55.375833	51	66	
7/3to6/3	27735.696674	66.428000	175	183	
7/3to7/2	10373.851619	56.029167	41	69	

```
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 13-4: Left side: object browser showing result in table. Background: interactive network result viewer. Foreground: print previewer allows to send network view or results view to printer or pdf file.

The main functions, like trip-generation and routing are menu driven. More complex functions like Open Street Map import, ODM editing or the SUMO simulation are processes and do have their own GUI.

In the near future, the network editing capabilities will be extended, allowing to add edges, nodes, stops, sensors and so forth. The basic architecture is already in place. Moreover the quality of the result view will be improved by adding scales and options how results are displayed.

13.5 Scripting

Scripting with SUMOPy allows to write complex simulation macros with a few lines of code. The Scenario instance can be initialized by giving a scenario name and a working directory, which will be the directory where all files are stored. The scenario name will be the root name of all simulation files. The following lines initiate a scenario, download from Open Street Map server (OSM) and convert it into a SUMO network and polygon file, no matter how big the area is.

Example code to initialize a scenario and download a network from Open street:

```
scenario=SumoScenario('my_scenario', 'my_scenario_dir')
scenario.import_osm(bbox=[44.451,11.2864,44.533,11.4007])
```

Districts, demand and vehicle types can be added successively to the scenario by calling the respective methods. Trip- and route generations are both methods of the scenario. The command to run SUMO while producing files for various outputs quite simple, as shown below.

Example code for running a SUMO simulation, simulating for one hour and generating different types of output:

```
scenario.sumo( simtime_start = 0, simtime_end = 3600,
               is_tripinfo = True, # Trips output
               is_edgedata = False,     # no Edges output
               is_lanedata = False,          # no Lane output
          is_edgeemissions = True, # Edge emissions
          is_edgenoise = False,        #     no     Edge     noise
          )
```

The results (here trip data and edge noise) will be read from the output files into the result data structure (see Figure 13-2) from where it can be viewed. In particular all data is stored in NumPy arrays which allows a fast processing.

Current work in progress is to make changes to the network through the command line interface immediately visible in the network viewer and result viewer through appropriate callback functions.

13.6 Conclusions

The basic concepts and architecture of SUMOPy have been outlined and the GUI as well as the scripting possibilities have been explained. Evidently the current version does only support a subset of SUMO’s potentials. Nevertheless, we have shown the next steps how SUMOPy can be extended to a user-friendly, yet powerful multimodal traffic micro-simulator.

The main development priorities are the TraCI module (with PRT and GRT vehicle controls) and a better network editing capabilities. Also the implementation of a virtual population would be an important step towards more realistic user models.

Yet, even with all the proposed extensions, SUMOPy is unlikely to compete with commercial micro-simulators. But its strength will be its openness and transparent architecture, which invites to implement specialized transport systems (like ITS, PRT or GRT) and test them within a complex transport demand.

Thus, SUMOPy has the potential to make SUMO accessible to a wider user group as it allows scientists to work more on the transport problem or on transport technologies rather than on programming issues.

13.7 References

[1] Michael Behrisch, Laura Bieker, Jakob Erdmann and Daniel Krajzewicz. SUMO - Simulation of Urban MObility: An Overview In: SIMUL 2011, The Third International Conference on Advances in System Simulation (2011).

[2] D. Krajzewicz, M. Hartinger, G. Hertkorn, P. Mieth, C. Rössel, P. Wagner, J. Ringel. The "Simulation of Urban MObility" package: An open source traffic simulation. In 2003 European Simulation and Modeling Conference (2003)

[3] Traffic modeler home page: http://trafficmodeler.sourceforge.net/

14 An Integrated Framework for Multi-Agent Traffic Simulation using SUMO and JADE

Guilherme Soares, Jose Macedo, Zafeiris Kokkinogenis and Rosaldo J. F. Rossetti;
Artificial Intelligence and Computer Science Laboratory (LIACC), Department of Informatics Engineering (DEI), Portugal
Zafeiris Kokkinogenis;
Institute of Mechanical Engineering (IDMEC), Faculty of Engineering, University of Porto, Portugal

14.1 Abstract

The rapid and ever-increasing population and urban activities have imposed a massive demand on Urban Transportation Systems (UTS). These systems were not prepared for such events, so traffic congestion and defective metropolitan systems were a direct consequence of this deficiency. The explosion of the computing technology brought together expertise from different scientific and technical disciplines giving birth to new computing and communication paradigms. Taking advantage of the modelling and simulation technologies we have devised a framework that combines the characteristics of the Multi-Agent System Development Framework, JADE, and the microscopic traffic simulator, SUMO, for the development and appraisal of multi-agent traffic solutions in contemporary traffic and transportation systems.

Keywords: SUMO, JADE, multi-agent systems, artificial societies, artificial transportation systems.

14.2 Introduction

The rapid and ever-increasing population and urban activities has imposed a massive demand on urban transportation systems [6]. Efficient transportation systems are crucial to an industrialized society and being its main communication infrastructure, rapid and effective solutions for traffic congestion are needed to prevent its negative impact on the city's social and economic welfare. A way to address this issue is resorting to the use of modelling and simulation as an imperative tool at decision maker's hands to effectively devise appropriate policies and management strategies.

The transportation domain is characterized by a high degree of uncertainty and dynamicity especially when considered in an urban context. Therefore, the use of simulation can result into improvements to the design and analysis of several management solutions for the optimization of the urban network throughput. Such an approach can provide us with the possibility of comparing studies between new infrastructure schemes or control algorithms without having to interfere with the real world directly, causing unnecessary and harmful disruptions.

Taking advantage of the simulation technologies a new generation of mobility systems, Intelligent Transportation Systems (ITS), arose allowing these algorithms to be implemented and polished before being deployed [4]. ITS is growing as the result of the synergy between Information and Communication Technologies and the Transportation Systems, which include vehicles and networks that move people and goods. Traditionally, mathematical equations have been used to describe the drivers and pedestrians movements taking into account several flow restraints, and used to tackle traffic related issues and to model them. According to this approach, traffic problems were handled as a whole, and the solution was a product of the fulfilment of all trips.

The formalization of the ITS concept is to be considered a great achievement by the transportation engineering community of practitioners and researchers. However, in the last few years the traffic and transportation domain has made a breakthrough in the way it is conceived, implemented and managed. The explosion of the computing technology in terms of applications we are experimenting in the last decade brought together expertise from different scientific and technical disciplines giving birth to new computing and communication paradigms. New type of socio-technical systems arose from such mutual conjunctions where people and technology live in mutual symbiosis. The transportation and, generally speaking urban domain could not be impermeable to such revolution. Indeed it proves to be a valid test-bed where such new social and technological paradigms can be applied.

Normally, in the development of traffic solutions the use of a simulator is very straightforward related to traffic flow and junctions management. Despite many attempts and published papers, the solutions presented to date do not make full use of the concept of intelligent agents. Additionally, the Multi-Agent Systems (MAS) approach has become recognized as a useful approach for modelling and simulating complex systems [5].

Keeping in mind the revolution in urban transportation mentioned above and guided by the need to design more human-centric economic and environmental solutions, a framework that generates an urban context, meaning a traffic network, respective infrastructure and its population, is necessary so that analysts and designers can study, develop and evaluate their policies and strategies.

In this paper, we present a framework that comprises all these requirements, providing community of researchers and practitioners with a tool that can instantiate an heterogeneous Artificial Society (AS) of drivers and Intelligent Traffic Light management solutions, immersed in a realistic traffic environment. The concept of AS can be used by traffic managers or government institutions as a test-bed for strategies and policies analysis towards the concept of social awareness for a sustainable use of resources.

Combining a powerful and standardized MAS development framework - JADE, with a large-scale microscopic traffic simulator - SUMO, our approach allows different types of studies, namely traffic control algorithms, service design, and also studies for the assessment of new policies and vehicle-to-vehicle communication applications.

The remainder of the paper is organized as follows. Section 14.3 discusses the methodological approach, proposing a general architecture underlying the characteristics of each component. Section 14.4 describes in detail the implementation procedure, whereas Section 14.5 presents a simple proof of concept scenario. Finally Section 14.6 draws some conclusion and points out future developments.

14.3 Methodology

Having to deal with atomic entities in the transportation domain the detail level comes down to the vehicle or better to the driver resolution. An Agent-Based Modelling approach seems to be the appropriate way to represent the road traffic environment, infrastructures and driver entities that live and interact in and with it. The microscopic traffic simulator chosen to provide the environment and traffic entities simulation is SUMO.

SUMO (**S**imulation of **U**rban **MO**bility) [1] is an open-source, highly portable, microscopic and multi-model traffic simulation package designed to handle large road networks and to establish a common test-bed for algorithms and models from traffic research.

SUMO is a microscopic traffic simulator, very popular in the research community, with a high number of scientific papers referring to it. Besides the mentioned features it also facilitates interoperability with external applications during run time using TraCI (**Tr**affic **C**ontrol **I**nterface) [8]. This interface uses a TCP based client/server architecture providing access to SUMO. It opens a port in SUMO's simulation process and awaits well-defined outbound commands.

On the other side of our approach is **JADE** (**J**ava **A**gent **DE**velopment Framework), a free software framework to develop agent-based applications. Its goal is to simplify the development while ensuring standard compliance through a comprehensive set of system services and agents. JADE is fully implemented in Java language and is compliant with the FIPA (Foundation for Intelligent Physical Agents) specifications.

This framework can be considered an agent middleware that implements an Agent Platform and a development framework. It deals with all those aspects that are not peculiar of the agent internals and that are independent of the applications, such as message transport, encoding and parsing, or agent life-cycle (AMS).

The agent platform can be dispersed on several computers, where each runs a single Java Virtual Machine (JVM). Each JVM is basically a container of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host.

TraSMAPI (**Tra**ffic **S**imulation **M**anager **A**pplication **P**rogramming **I**nterface) is another component in our approach, and can be seen as a generic microscopic traffic simulator API. TraSMAPI allows real-time communication between traffic-management agents and the environment created by various simulators.

Its API offers an abstraction level higher than the API of most common Microscopic Traffic Simulators so that, ideally, the solution should be independent of the microscopic simulator choice.

Taking into consideration all the general goal and software selection we have devised the following architecture depicted in Figure 14-1.

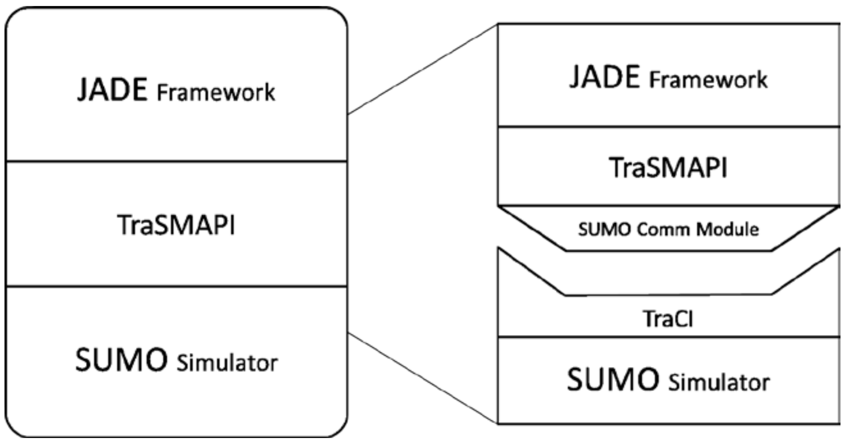


Figure 14-1: Framework Architecture.

The SUMO Simulator offers an API allowing access to its simulation state - TraCI [8]. For an external application to communicate with this software it must obey TraCI's communication protocol and messages types. The Sumo Communication Module attached to TraSMAPI, converts this low-level simulator's API into a higher-level one, which will be then used by our artificial society of drivers and Traffic Lights infrastructure. These will be implemented in JADE's MAS development framework coupled with TraSMAPI, as illustrated in Figure 14-1.

14.4 Implementation

Our goal is to have a heterogeneous artificial society of drivers in JADE's agent platform, each of its entities responsible for one vehicle in the SUMO's traffic environment.

However it would be very computationally expensive to simulate hundreds or thousands of vehicles and driver's decision-making in JADE. Hence, we have adopted the delegated-agent concept, which has been used in [7], to separate the tactical from the strategic layer of the agent, and execute them in parallel, thus improving performance, Figure 14-2.

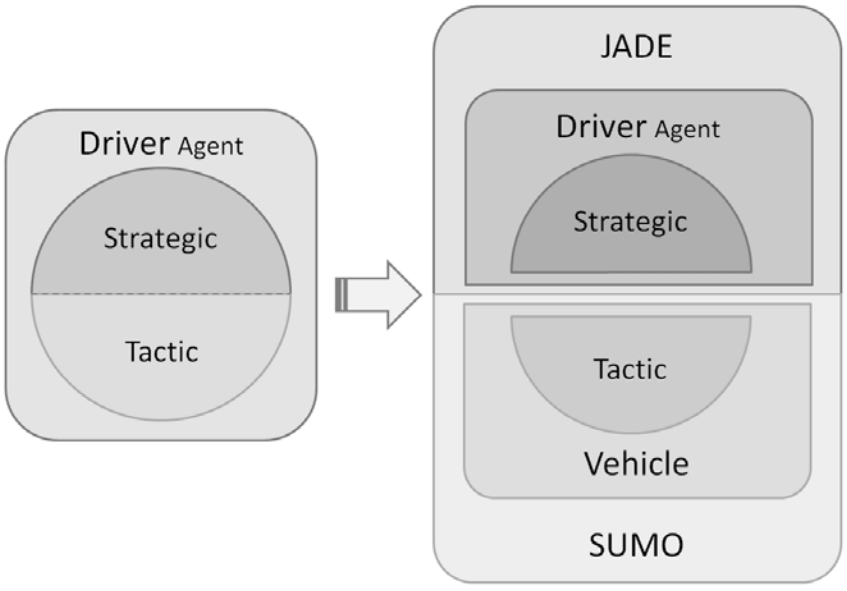


Figure 14-2: Driver's reasoning layers: tactic-reactive layer in SUMO and strategic-cognitive layer in JADE.

The tactic-reactive layer is delegated to SUMO's microscopic traffic simulation model, such as car following [2], leaving the cognitive-strategic layer to the agent implemented in JADE's agent platform.

Aiming at this purpose we must make possible the association between a Driver Agent, instantiated in JADE's agent platform and a vehicle simulated by the microscopic traffic simulator SUMO.

In order to achieve these ideas, we need to extend the scope of TrasMAPI, enabling it to build an abstraction over a vehicle entity, as depicted in Figure 14-3.

Thus, we have implemented the communication protocol concerning the vehicle's methods for variable retrieval or state change taking into account the compliance with the well-defined TraCI's instructions. For more detailed information, see TraCI's documentation and reference [8], where the protocol and message flow are presented and explained.

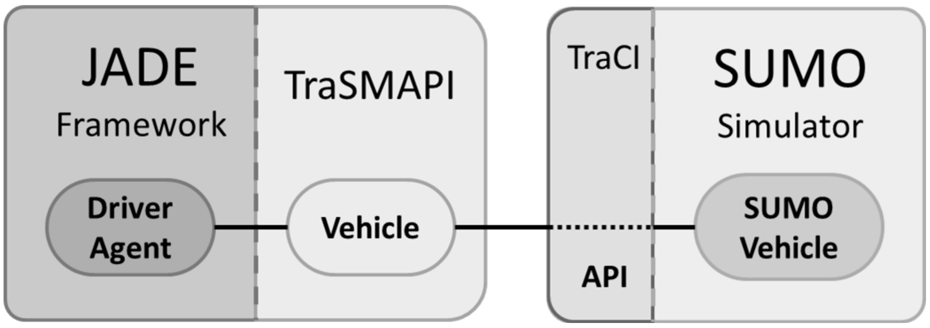


Figure 14-3: Vehicle entity abstraction through the architectural design

14.5 Experimentation Scenario

As a proof of concept for our test-bed, we have used and improved the German city of Eichstätt transportation network, using the JOSM application, correcting some intersections and lane cardinality, as seen in Figure 14-4. Graphically we added the GoogleMaps decals to improve the user immersion during the visualization of the simulation process in run time.

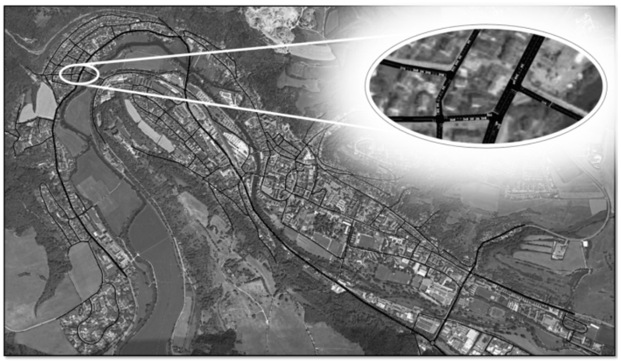


Figure 14-4: Eichstätt transportation network.

The configuration files used to load SUMO simulation are only the network file and GUI settings. In this experimental set-up we intend to reproduce the drivers' decision-making process in route choice accounting for previous travel times.

With our framework we can instantiate a Driver to each vehicle simulated in SUMO. Therefore one may use all the methods that this simulated instance has, such as change destination, speed, route, among others.

In the beginning of the simulation it is given to each entity of the AS a random origin edge and orientation and a different random destination. With this, the agents' reactive layer in SUMO can make use of its shortest path algorithms and calculate the best route based on time travel to accomplish each driver's desire.

As a proof of concept, each Driver sets a value to its travel time table in SUMO, so that when the rerouting by travel-time algorithm is invoked, it will take into account the new table's values. This attitude argues in favour of the drivers' awareness and decision-making capabilities.

The instantiated traffic-light entities are an extension of a previous experience concerning the concept of advisory-based traffic control. The interested reader is referred to [3] for more details and further discussion.

14.6 Conclusions

Simulation proved to be an effective approach to analysing and designing novel traffic solutions in socio-technical systems. Traffic systems have been subjected to a lot of improvements in last decades and travellers have, in general, witnessed a revolution in the way a trip is planned and performed in urban networks. Hence, facing the current traffic situation in most developed countries it is now imperative to foster new transportation solutions using state-of-the-art technologies.

User is now a central figure in the new vision of urban systems, and most simulators follow a macro/microscopic approach as an attempt to improve the representation of traffic flow and management rather the traveller behaviour. Therefore we present a powerful framework for instantiating heterogeneous multi-agent systems representing different aspects of the traffic domain such as the socio-technical issues, embedded in intelligent artefacts, aiming to designing more human-centric and hence sustainable solutions. For this purpose two well-supported and popular platforms in the researcher and practitioner communities have been used, namely the JADE framework for developing agent-based systems and the SUMO microscopic traffic simulator.

The resulting tool also reveals a great flexibility and potential for multi-agent system development in terms of the socio-econometric aspects of the traffic domain, since one can easily develop his own Artificial Societies of drivers (synthetic population) immersed in a realistic representation of the urban traffic environment, where each agents is represented with its own preferences and beliefs. Such a society can be used thus to design solutions based on individual or collective intelligence and participation (social-aware attitudes) or as a test-bed for policy evaluation by governmental institutions. Experimentations with such an AS could provide the community of researchers and practitioners with better insights into the formation of emergent mobility patterns and how information or knowledge can affect the drivers' decision-making process.

In the present work we have shown how to instantiate a synthetic population of drivers on top of the SUMO traffic environment. This population resembles and can be considered as an artificial society, since it has all necessary characteristics, namely coordination, competition and collaboration. A demonstration of using the tool is described where driver agents decide to change their destination or to follow a different route according to their individual desires.

Moreover, we have also extended the ability of our framework to build artificial agents for traffic management through traffic light control. With this integrated tool, one can build and test with multi-domain test-beds. From vehicle-to-X (V2X, where X stands for vehicle/infrastructure/grid) scenarios to more complex driver/traffic-light interaction, practitioners and engineers may evaluate contemporary ITS solutions in their very essence, simulated entities can are now endowed with reasoning abilities in proposed artificial society of travellers and ITS-technologies.

14.7 Acknowledgments

This project has been partially supported by FCT (*Fundação para a Ciência e a Tecnologia*), the Portuguese Agency for R&D, under grants SFRH/BD/67202/2009 and SFRH/BD/51256/2010.

14.8 References

[1] M Behrisch, L Bieker, J Erdmann, and D Krajzewicz. SUMO - Simulation of Urban MObility: An Overview. In SIMUL 2011, The Third International Conference on Advances in System Simulation, pages 63-68, Barcelona, Spain, 2011.

[2] S Krauss, P Wagner, and C Gawron. Metastable states in a microscopic model of traffic flow. Physical Review E, 55(5):5597, 997.

[3] J Macedo, M Soares, I Timoteo, and R J F Rossetti. An approach to advisory-based traffic control. Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on, pages 1-6, 2012.

[4] J C Miles and A J Walker. The potential application of artifficial intelligence in transport. Intelligent Transport Systems, IEE Proceedings, 153(3):183-198, 2006.

[5] Lisa Jean Moya and Andreas Tolk. Towards a taxonomy of agents and multi-agent systems. In Proceedings of the 2007 spring simulation multiconference - Volume 2, SpringSim '07, pages 11-18, San Diego, CA, USA, 2007. Society for Computer Simulation International.

[6] F I G Report. The Need for Spatial Information Management Rapid Urbanization and Mega Cities :. Number 48.

[7] Joachim Wahle, Ana Lúcia C Bazzan, Franziska Klügl, and Michael Schreckenberg. The impact of real-time information in a two-route scenario using agent-based simulation. Transportation Research Part C: Emerging Technologies, 10(56):399-417, October 2002.

[8] Axel Wegener, Michal Piórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. TraCI: an interface for coupling road traffic and network simulators. In Proceedings of the 11th communications and networking simulation symposium, CNS '08, pages 155-163, New York, NY, USA, 2008. ACM.

15 dSUMO: Towards a Distributed SUMO

Quentin Bragard, Anthony Ventresque and Liam Murphy;
School of Computer Science and Informatics, Lero@UCD, University College Dublin, Ireland

15.1 Abstract

Microscopic urban mobility simulations consist of modelling a city's road network and infrastructure, and to run autonomous individual vehicles to understand accurately what is going on in the city. However, when the scale of the problem space is large or when the processing time is critical, performing such simulations might be problematic as they are very computationally expensive applications. In this paper, we propose to leverage the power of many computing resources to perform quicker or larger microscopic simulations, keeping the same accuracy as the classical simulation running on a single computing unit. We have implemented a distributed version of SUMO, called *dSUMO*. We show in this paper that the accuracy of the simulation in SUMO is not impacted by the distribution and we give some preliminary results regarding the performance of dSUMO compared to SUMO.

Keywords: Urban Traffic Simulation, Distributed Simulation, Road Network Partitioning.

15.2 Introduction

Traffic congestion is a major problem for most urbanised societies leading to delays experienced by commuters, accidents in dense traffic, etc. which costs the society and individuals. If the 86% of the population of the developed countries that are predicted to live in cities by 2050 [8] use their vehicles as we do now, it will only get worse. In Dublin for instance, 34.6% of the active population commute by car [1], even after a huge effort from the municipality to reduce this number (50% in 2000 [14]). Dublin City Council implemented several new policies in the period to obtain this result: incentives to use alternative transport modes, modification of the infrastructure (e.g., cycle lanes, tramway), etc.

In this regard, urban traffic simulations can be useful at design time (road planning) and at run time: (i) the example of Dublin where several policies were implemented in the early 2000s, and led to almost 30% reduction in the number of commuters show that it is possible to have an impact on the city traffic by an adequate urban planning; (ii) correct decisions taken during incidents by emergency personnel can prevent congestion or ease their evacuation. There exist several types of traffic simulations: statistical models which predict certain values (e.g., number of vehicles, travel time) in the road network using stochastic methods; macro-simulation which consists in modelling the traffic flows (including lane changing for instance); and micro-simulation, representing every vehicle in the model by an agent that makes decisions based on predefined behaviour, context, etc. In this paper we focus on micro-simulations (also known as microscopic simulations) as they have two clear advantages. First, they operate at the vehicle level and thus they allow the user to observe the behaviour of individuals; decision makers can literally see the impact of the infrastructure and their policy on the traffic. (Most solutions now have some sort of GUI which shows with many details the real environment. It is thus very useful for decision makers to understand what is going on, what is not optimal in the system, and what the consequences of their decisions are.) Secondly, they are the most accurate simulation tools, as they model in great detail; the drivers' behaviours, their environment, etc. These two characteristics make micro-simulations the best tool in our scenarios where we target an accurate understanding of the evolution of the traffic with regard to some specific infrastructure design, and some exact short term prediction.

However, both cases are complex: while in the first case the challenge resides in the complexity of the simulated model, the processing time is critical in the second case. Microscopic simulations are not known to perform well for large scale simulations or for fast prediction, which seems to exclude micro-simulations for our scenarios. But, there exists a classical solution to this problem of scale/speed: distributing the processing [19]. So the question is now: can we come up with a distributed version of a traffic simulator which allows to scale up or speed up the simulations?

In this paper, we present a distributed version of the well-known [10] microscopic mobility simulator SUMO [3], that we called *dSUMO*. In dSUMO simulations that normally run on a single CPU of a single machine can run on several cores or a single machine and on remote machines alike. dSUMO clearly targets large scale and very quick simulations that might not be able to run on single machines.

In dSUMO every machine can run various dSUMO nodes (e.g., one per core or one per machine) and nodes distributed on remote machines or local cores alike communicate using sockets. The SUMO instances located in every dSUMO nodes are driven by some dSUMO components (called *Runners* or *Handlers*) and the distribution is for the end-user totally transparent: vehicles are transferred from one dSUMO node to another, simulation can be paused, simulation models are partitioned and each partition allocated to a node, etc. In the rest of this paper we first give an overview of dSUMO (Section 15.3). Then we present the partitioning methods that we can use for an implementation of dSUMO (Section 15.4). This is followed by a description of dSUMO architecture, communication protocol and partition borders management (Sections 15.5, 15.6, 15.7). After that we demonstrate that dSUMO scales-up and speeds-up pretty well, while it does not seem to introduce errors (Section 15.8). Finally we conclude the paper and give some possible future directions and plans for dSUMO in Section 15.9.

15.3 dSUMO: Overview

The goal of a distributed simulator is to run a a single machine (e.g., single CPU) simulator on various machines without anyone really noticing it. For instance, we want here to run several instances of SUMO on different CPUs of a single machine or remote machines, and we want to make sure this is transparent for the users: for them there is no apparent difference between the two modes of running SUMO. Every bit that makes the system distributed, the localisation of the running instances, the distribution of data, etc., needs to be hidden from the user. To achieve this objective, dSUMO has to address several problems:

- partitioning of the data: i.e., how do we split a single environment (map, cars, etc.) to fit in the running nodes we have?
- communication between instances of the running system: i.e., what protocols do we need when dSUMO nodes want to send meaningful information to their neighbours?
- synchronisation between dSUMO nodes: i.e., what information is needed to be exchanged between partitions while they run their individual simulations and their neighbours need some of their information (position of cars when they are close to the borders)?

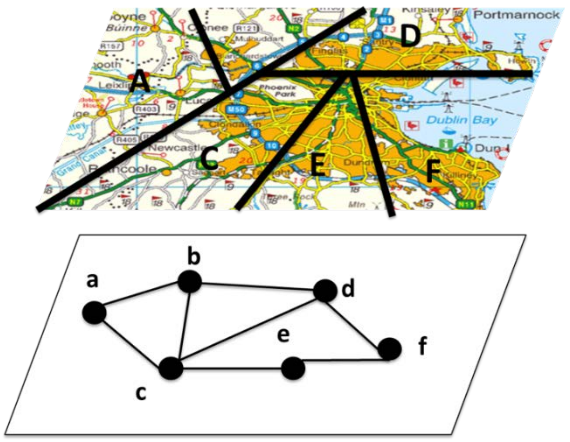


Figure 15-1: A map is partitioned and each region is assigned to a computing node in dSUMO.

Figure 15-1 describes the partitioning of a map and the allocation of each partition to a computing node in dSUMO: partition **A** in the map shown in the upper half of the figure is managed by node **a** in the bottom part of the same figure, etc. Actually we are not interested in the map itself, but in the road network, and if a vehicle travels on a road segment that crosses the border between two partitions, e.g., **A** to **B**, then in dSUMO a message is sent from the origin node (**a**) to the destination node **(b)**. These messages describe the vehicle characteristics (type, speed, position, destination, route, etc.) and generate a new vehicle in the destination node, **b** in our example of a vehicle crossing the border **A/B**. Section 3 describes the state-of-the-art partitioning algorithms that can be used for the same task in dSUMO, while Section 4 shows the technical composition of each dSUMO node.

The synchronisation protocol of dSUMO is depicted in Figure 15-2. SUMO is a step based simulation and we do not modify this behaviour. At the end of each step every node sends synchronisation messages to all its neighbours: vehicles that are crossing the borders, position of vehicles that have just crossed as the car following model needs it, etc. Then, when a node receives a message from all its neighbours, it knows they have all finished their step, and it can move on to the next step. This gives to the system some interesting properties (reliability, failure detection, etc.) that we describe in Section 15.6.

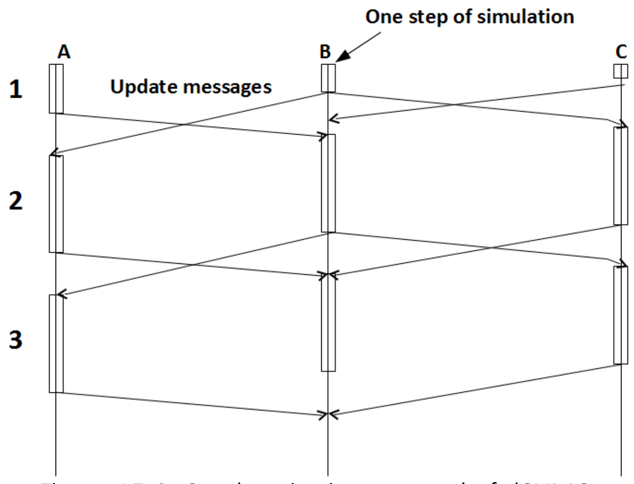


Figure 15-2: Synchronisation protocol of dSUMO.

Vehicles in SUMO adapt their behaviour according to the vehicles preceding them, and then we have to take into account in dSUMO that there is potentially a lack of information for vehicles when they are close to the border between partitions. We propose to update the position, speed, etc. of vehicles that just crossed the border to their original node, in order to let the vehicles know that there are obstacles just after the border (see Section 15-7).

15.4 Partitioning

The classical first step when setting up a distributed system is the partitioning of processes and data. It consists in splitting instructions and environment and to put it onto the several nodes that will run the distributed system. Some constraints: processing power of the nodes, characteristics of the connections, etc. drive the partitioning. Here we cannot partition the processes as every dSUMO node runs a similar SUMO instance. So all we care about is the partitioning of the data and the distribution to the various dSUMO nodes.

Every simulation model in SUMO is composed of a map, a population and detectors. We do not care here about the detectors as setting them and so on is similar to SUMO -- note that we do not provide here a global logging system for the detectors and each computing node will record the outputs of the detectors that it hosts. You can see in Figure 15-1 a map of the city of Dublin and a possible partitioning of it. Every partition is assigned to a dSUMO node, i.e. its road network and the vehicles that are running on it. Classically, partitioning a map is done using a space partitioning algorithm. But we advocate that our use case (road network) is closer to graph partitioning than classical space partitioning for distributed simulations. The rest of this section addresses the classical partitioning methods, our approach and a solution we presented in a previous publication: *SParTSim* [22].

15.4.1 Space Partitioning

Distributed simulations usually rely on space partitioning for interest management, i.e., management of communications and synchronisations of agents in a number of nearby space partitions. The agents subscribe to these partitions close to them and receive events and updates that may interest them. This is exactly the scheme used in massively multiplayer online games, where participants (e.g. players' avatars, non-player characters) join and leave partitions as they move on the large virtual environment distributed on several servers [5].

Space partitioning schemes can be divided into two classes: uniform partitioning and non-uniform partitioning. Uniform partitioning split the virtual space into regular, uniform, static partitions, usually rectangles [21] or hexagons [13] (see Figure 15-3).

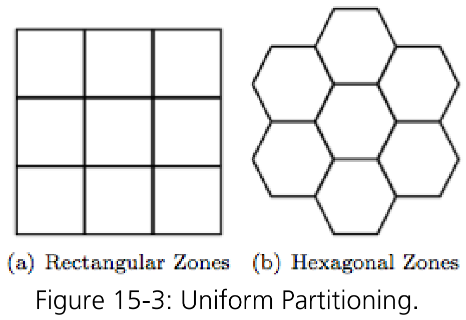


Figure 15-3: Uniform Partitioning.

Choosing between squares/rectangles and hexagons has garnered a lot of attention and usually depends on application and so on [16], [23]. To overcome the problem of unbalanced load on zones, authors of [17] propose a solution which creates overlays on the simple elements (rectangles or hexagons). This is less regular but allows to avoid to waste some zones when there is no or little activity on them. This approach is more flexible and can help developers to define precise partition shapes according to terrain, etc., but increases the overhead of managing the areas of interests: finding the neighbouring areas and so on. Figure 15-4 shows a possible non regular square partitioning over Dublin map.

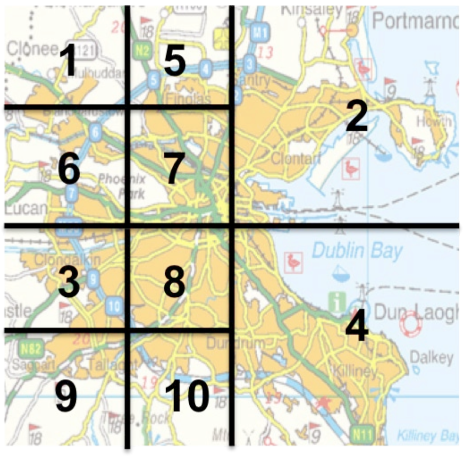


Figure 15-4: Non regular Partitioning

Figure 15-5 shows some irregular overlay on a uniform square partitioning.

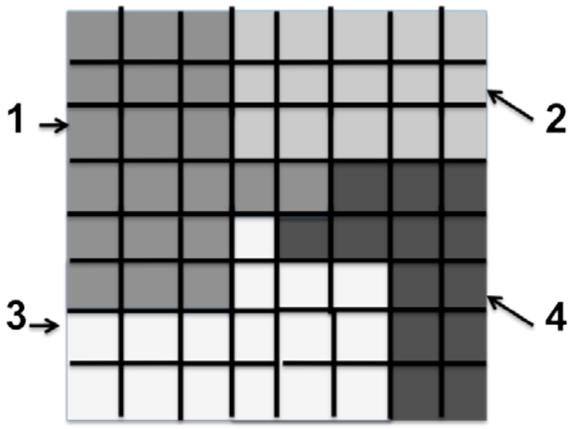


Figure 15-5: Irregular overlays.

On the other hand, non-uniform partitioning schemes allow partitions with different shape, size, etc. Most of the time these solutions employ a hierarchical structure to store the relationships between partitions to ease the retrieval and navigation between them. Authors in [2] defines a system where partitions have an arbitrary shape and the relations between them are stored in a binary space partitioning tree (BSP tree). [18] gives another partitions schemes presentation where they describe four types: quadtree, k-dimensional tree (k-d tree), constrained k-d tree, and region growing. The decision regarding the kind of structure usually depends on some properties that the developers want to achieve: balancing, flexibility, processing-time, etc. There exists some improvements using, for instance, clustering of quad trees [20] to lower the processing time of finding neighbours. Another approach uses Voronoi diagrams [9] which have very nice mathematical properties (all points in the Voronoi region are closer to the centre of its region than the centre of any other region) but are heavy to compute.

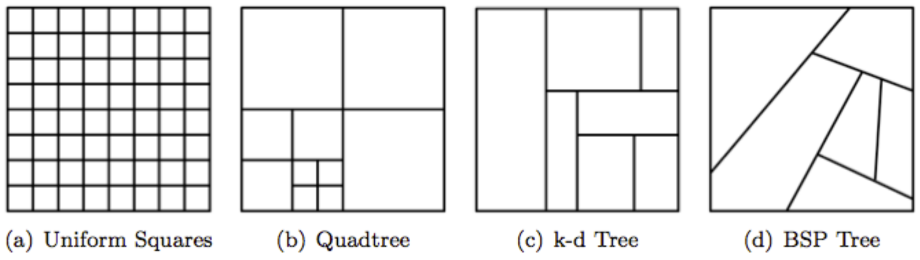


Figure 15-6: Non regular partitions.

15.4.2 Graph Partitioning

Graph partitioning is everywhere in computer science and engineering: distributed computing, computer vision, very large scale integration for circuit layout composition, telephone network design, physical mapping of DNA, route planning, clustering, etc. (e.g., [4, 12]). The general presentation of this problem consists in splitting a graph such that the number of vertices in each partition is balanced and the number of cut edges is minimised. In our case, we obviously interpret that as maximising the load balancing between partitions and minimising the communication, i.e. the number of vehicles that cross the borders between partitions. This is a very complex problem, most instances being NP-hard [7] and many heuristics have been proposed until now [11][12][15].

We think that space partitioning techniques described above are not really relevant for urban traffic simulation as the space is composed of a road network. The nature of this road network is such that graph partitioning makes more sense. But they are usually very complex algorithms, may be a little too complex for simple graphs like road networks. Our challenge is then to define a road network partitioning that is as efficient as graph partitioning techniques and as effective as space partitioning ones.

15.4.3 SParTSim Algorithm

SParTSIM stands for Space Partitioning guided by road network for distributed Traffic Simulations. It defines a hierarchical partitioning based on the road network, which is by nature hierarchical (there are several levels of roads). In Ireland for instance there are six levels in the road hierarchy: National Primary Roads and Motorways, National Secondary Roads, Regional Roads and three levels of Local Roads. Figure 15-7 shows three of those levels for Dublin: motorway M50 (strong solid line), National primary roads N1-4, N7 and N11 (tighter solid line), and some local primary roads (dashed lines).

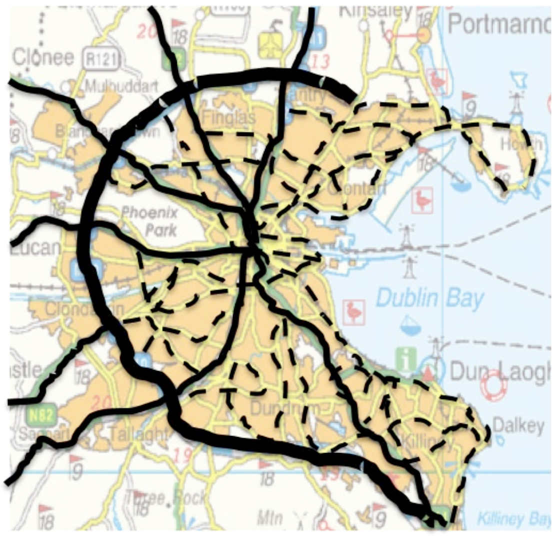


Figure 15-7: Three top road levels for Dublin.

SParTSIM uses this hierarchy, following the idea proposed in Metis [11]. In such hierarchical partitioning scheme the complex graph is simplified in a version easily partitioned, and takes gradually back its original size, refining the original partitioning during the process.

SParTSim also applies a region-growing technique: it takes an initial vertex of the road graph and grows a region by adding new nearby vertices. Each region grows in all the directions and competes with others for vertices, until there is no more vertex available. Regions, or partitions, then trade their vertices to balance their size. See our paper [22] for more details.

15.5 Architecture of dSUMO

In this section, we present the global architecture of our solution, while we show in more details the communication/synchronisation in the next section and the management of the borders, probably the most complex part of distributed simulation, in the second next section. In short, the architecture of dSUMO relies on a few numbers of objects whose purpose is to manage the simulation of a dSUMO node and to communicate with other dSUMO nodes. See Figure 8 for more details.

Every physical machine in dSUMO runs a *dSUMO Container* which is a global environment for the different *dSUMO nodes*, each of them running an instance of SUMO. A classical deployment consists in running as many nodes on a machine as it has cores: e.g., on a dual core machine, there will be a single Container running two nodes. The Container offers the interfaces for communication with other distant nodes (*Server* and *Clients*), as well as the manager for the simulation (*Handler*). Figure 15-8 presents the different modules and their connections. There is only one Server per Container, but as many Clients as neighbours the dSUMO nodes have (they share the clients). Similarly, only one Handler is needed to command the simulation in the Container. Finally, instances of SUMO are managed by their respective *Runners*. In this section we give some details on each of these modules.

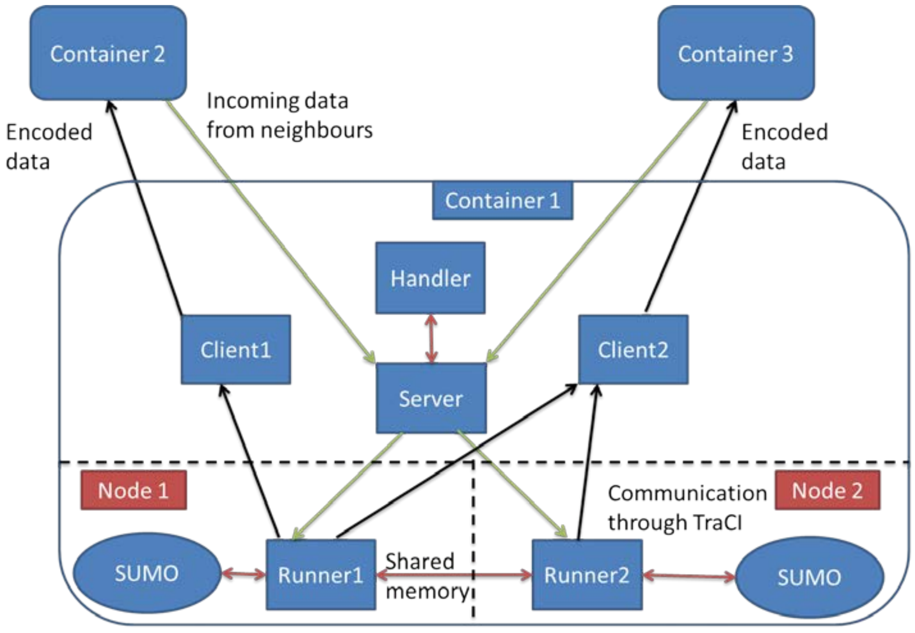


Figure 15-8: Architecture of dSUMO.

15.5.1 Handler

The Handler serves as an interface with the operator (e.g., a user or a script that commands the simulation). It uses the other modules and sends them orders: pause/resume, dump the dSUMO node's state to a file, add/remove cars, stop cars to simulate accidents, etc. It is extensible and aims at providing all the functionalities that you can get from SUMO.

15.5.2 Runner

The Runner is the most important element of the dSUMO system. This module interacts directly with the SUMO instance through TraCI and does the setup, the evolution, etc. First, the Runner creates the environment needed to run the distributed simulation: it creates the links with other nodes (distant or local to the Container) and maps the road ends on its partition to these other neighbouring nodes, loads the vehicles in the simulation, etc. Then, when the system is set up, the Runner looks after the vehicles leaving or joining the partition its managing: (i) it instantiates the coming vehicles' route in its partition (ii) it collects information about vehicles close to the border but in other partitions and gives this information to its own vehicles close to the border (iii) it detects when vehicles are leaving its partition and transfers them to other nodes. Finally, it transfers to the each client the data they need to send to their respecting neighbour.

15.5.3 Client

When a dSUMO container c_i enters the system, i.e. a new machine is used, it also creates one Client per neighbour of its dSUMO nodes $n_1^{c_1}, \dots, n_j^{c_j}$ to handle the connections with them. At the end of every step, the Runners of nodes $n_1^{c_1}, \dots, n_j^{c_j}$ prepare the information they need to send to their neighbours and pass it to the corresponding Clients. When the Clients have all these messages from the Runners, they format the message and send it to the Server they are connected to. The message will be ended by the EoS messages[12].

15.5.4 Server

Servers are the components responsible for collecting messages from all the Clients of the dSUMO node's neighbours. These messages are translated into TraCI instructions and create new vehicles in the dSUMO node's simulation. Servers also inform TraCI when the simulation can move one step forward, i.e. when they receive the EoS messages[13] and are sure that no event can reach the node before the next step of the simulation.

15.6 Communication in dSUMO

A car reaching the border between two partitions in the simulation corresponds to a vehicle that needs to be transferred from a node to one of its neighbours. Technically, the dSUMO node's Runner gets the information that a vehicle is reaching the border with another partition and looks up in its dictionary to and the corresponding node. It then sends a message to the relevant Client which forwards the info to the neighbour's Server.

We have implemented the connection between dSUMO Clients and Servers using sockets to ensure the reliability of connection while keeping a good performance.

We have defined several types of message in dSUMO:

- A vehicle message contains the important information regarding the vehicle crossing the border: id, type (SUMO definition which contains all the characteristics of the car), speed, route.
- There are also bootstrap messages which synchronise the connection with something like a “ping” to the neighbour's Server.
- When the connection between nodes is established, the Servers acknowledge with a connection accepted message.

[12] See later section 15.6

[13] See later section 5; these messages inform that all the synchronisation events, i.e., vehicles crossing the border between partitions, have been sent.

- End of Step (EoS) are messages sent by Clients to their neighbours when all the vehicles have reached the border at the end of a step. They tell the neighbour that nothing can come from the node before the next step.
- The Back control message is used to control the replication of a car which just crossed a border. It contains the speed that a vehicle will have the next step in order to propagate the impact of a slowing down or a traffic jam.

15.7 Management of the borders

Managing the border is a critical element in distributed simulations. Vehicles need indeed to know what is after the border in order to make the correct decisions regarding their speed and so on. SUMO for instance has a strong car following model and the knowledge of the position and behaviour of the preceding car is very important for every vehicle. For instance, if there is a traffic jam on node *A*, close to its border with node *B*, vehicles coming from *B* which are closed to cross the border have to know the situation in order to change their route, slow down, stop, etc. If we do not have a very accurate border management in dSUMO then there might be a difference between the classical single CPU SUMO simulation and dSUMO which is not desirable.

15.7.1 Sending a car

When a car reaches the border of a partition, it needs to be sent to the node managing the nearby partition in order to continue its trip. Runners in dSUMO encode the most relevant information regarding the vehicle such as id, speed, position on the lane, route and type into a message before passing it to a Client who will send it through a socket to the proper neighbour. Once the message reaches the next node's Server, the information is decoded and the vehicle is created inside the SUMO instance by the Runner.

Actually dSUMO does not use the classical SUMO/TraCI method for creating vehicles. The actual implementation of the method that creates vehicles put them in a queue and adds them at the next step, not at the current one, to make sure the vehicles can safely be added. The problem for us is that it would cause a gap in the simulation.

Our solution consists in adding the newly created vehicle to the list of already existing vehicles. This new method allows the vehicles to move at the step $t + 1$ (if t is the step when they cross the border) instead of being added at $t + 1$ and moving at $t + 2$.

15.7.2 Back-controlling a car

Interest management is probably the most difficult thing with distributed simulations. The problem is that the car following model implemented in SUMO requires every car to know the characteristics (position, speed, etc.) of the car preceding it. Which, translated in dSUMO concepts, means that a car crossing the border between partitions **A** and **B** cannot disappear from **A** if there are cars still following it in **A**.

In dSUMO every road segment split by a border is then duplicated, and exists in both partitions. Let say road segment **RN1** is split by the border between **A** and **B**. **RN1** exists in **A** and **B**, but we say that the lane going from **A** to **B** is managed by **B**, and vice-versa. When car c_1 crosses the border, dSUMO node **A**'s Runner create a message with c_1, sends it to **A**'s Client responsible for the link with **B**, and this one sends a message to B's Server. The car has crossed the border -- and has disappeared from **A**. Now we do not want cars following c_1 to lose it from sight as its behaviour (speed, etc.) is important. **B** will then sends c_1's characteristics back to **A** at every step, until there is another vehicle that crosses the border or c_1 leaves the segment **RN1**.

Technically, we rely on methods used to move all the cars. We modified them to work for single cars and to have no impact on the overall simulation. They manage to send to the original partition where the car comes from the speed and position, and allows following cars to adapt theirs.

15.8 Validation

Our validation has two main objectives:

- to evaluate the impact of the distribution on the accuracy of the simulation, i.e., we check if the position of vehicles on dSUMO differs from the one with the centralised SUMO.
- to observe the execution time of dSUMO and to compare it to SUMO.

15.8.1 Validation Set-up

We use a grid network composed of 20x20 junctions linked by double way-single lane road segments of 200m. The distributed simulations run on partitions of either 10x10 junctions (4 partitions). The road links between junctions are still double way-single lane segments. All vehicles have a North-South or South-North predefined route (starting 40 at a time, on all of the 20 top and bottom junctions each turn). The machines used are either Pentium dual core T4300 (2.1GHz) with 4GB of RAM, or Pentium dual core P6200 (2.13GHz) with 4GB of RAM.

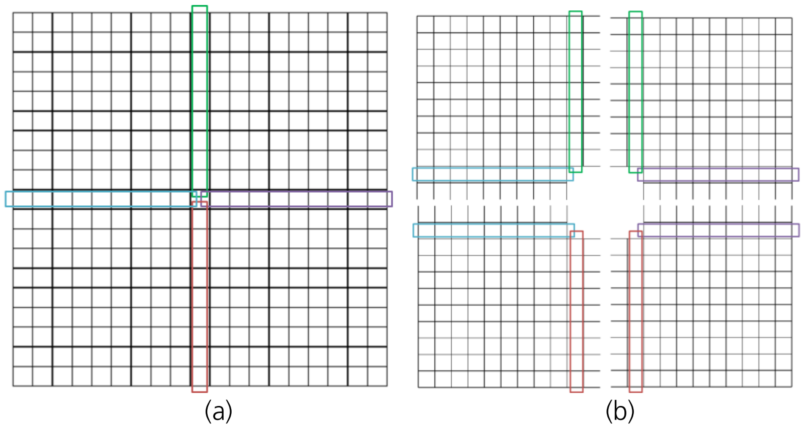


Figure 15-9: (a) Centralised version of the road network (b) Distributed version of the road network.

15.8.2 Accuracy of dSUMO

To prove that our distributed version of SUMO has no impact on the accuracy of the simulation, we need to show that with the same settings, at every step, the same cars on dSUMO and SUMO are at the same position and have the same speed. We identified two relevant and distinct scenarios which can measure the impact of the distribution:

- A very simple scenario with the basic settings described in the previous section. One car is sent on each vertical road every step. We then stop the simulation at some specific step and compare the position and the speed of every car in SUMO (centralised) with their equivalent in dSUMO.
- We quickly realised that traffic jams just after the borders are the more tricky situations, as cars need to be warned in advance when they reach the border that they will have to stop or slow down. In the second scenario, we stop some cars few steps

after they cross the borders, generating traffic congestion. When the jam is propagated to the previous partition, we stop the simulation and compare SUMO and dSUMO.

Note that for this comparative study, we had to set to 0 the random sigma value that gives some variability to vehicles behaviours in SUMO. This sigma value depends on the number of steps the car has spent in the simulation so far (which explains why this value is always the same if you run several times the same experiments with the same settings). Our problem here was that when vehicles pass from one partition to another, they appear as new for the SUMO instance of the dSUMO node and then have a different sigma than their counterparts in the centralised SUMO. Thus generating artificial gaps between SUMO and dSUMO. Null sigma means there is no randomness in acceleration and so on and, centralised and decentralised versions can be compared.

Our accuracy metrics corresponds to the percentage of cars that are at exactly the same position and have the same speed in SUMO (centralised) and dSUMO.

Table 15-1: Accuracy comparison of SUMO (centralised) and dSUMO.

Scenario	End of simulation (step number)	# of vehicles	Accuracy
Simple	150	2440	100%
Congestion	190	3080	99.38%

Our first attempt to run the first scenario proved to be a failure: it seemed cars are not created at the same absolute position in the distributed SUMO compared to the centralised one. It appeared to be because the road segment does not have the exact same length if they are a cul-de-sac or if they lead to another crossroad. Now, every road segments split during the partitioning becomes a cul-de-sac, and hence the length of the segments was artificially slightly different. We fixed this problem by adding another road segment after the border. As you can see in Table 15-1 the accuracy is the maximum, which means that the simulations with SUMO or dSUMO are 100% similar.

It is not the case with the congestion scenario though. After investigating the matter, it seems that there is an unexpected safety check in SUMO before a car is created on the new node. It obviously makes sense to create vehicles only if they can be added to the simulation. In our scenario though, SUMO retains some cars, while we know there is room for them: they are present on the previous node at the exact same place, proving that the car has the place to be added on the next node also. We will need more time to see where is this SUMO security/safety test and whether we can remove it safely. Anyway the accuracy is very high and we are rather satisfied.

15.8.3 Execution Time

As any other distributed system, distributed simulations need a lot of synchronisation and communication. In our case, nodes exchange messages representing cars that are passing the borders, cars that are close to the borders and may have an impact on the simulation, etc. and also messages informing that a step has been processed. And as for any other distributed system, this communication element is an overhead when we compare dSUMO to SUMO (which does not need this synchronisation step). Figure 15-10 shows the synchronisation time required for simulations with four partitions on the same machine (thus using scheduling between the two cores) and on two different machines. In average, we have around 0.32s, when all dSUMO instances are on the same computer, and around 0.38s, when they are on two remote computers. This is a bigger value than what we expected, and seems to be an issue for our idea of using dSUMO for very large scale or faster than real time simulations. We could argue that in the field of distributed simulations, techniques exist to reduce this synchronization time such as optimistic synchronisation mechanisms [6] (synchronisation is done only every *x* steps, at the risk of rolling back the simulation). But most of these solutions tend to sacrifice accuracy, which is a hard constraint in our case, and the reason why we pick micro-simulation. Another option could be to use other IPC techniques than sockets (e.g., distributed shared memory) or better communication library than the one we use currently. Note also that the synchronisation time is constant and seems to be independent of the number of cars we transfer and likely the number of nodes we have in the system.

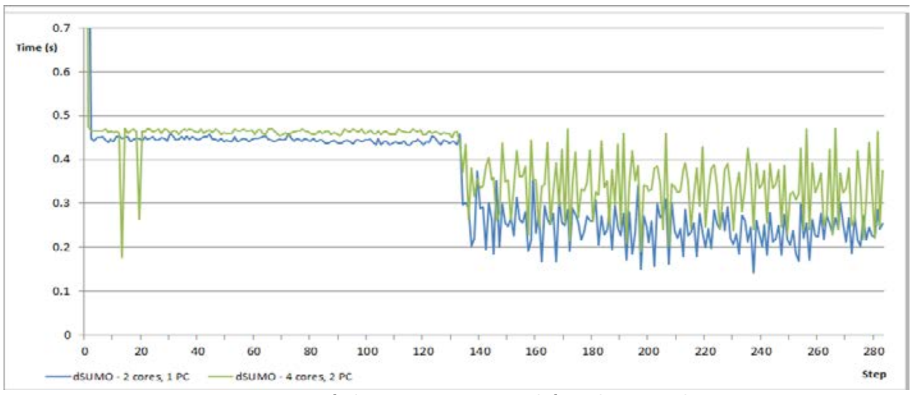


Figure 15-10: Evolution of the time required for the synchronisation step.

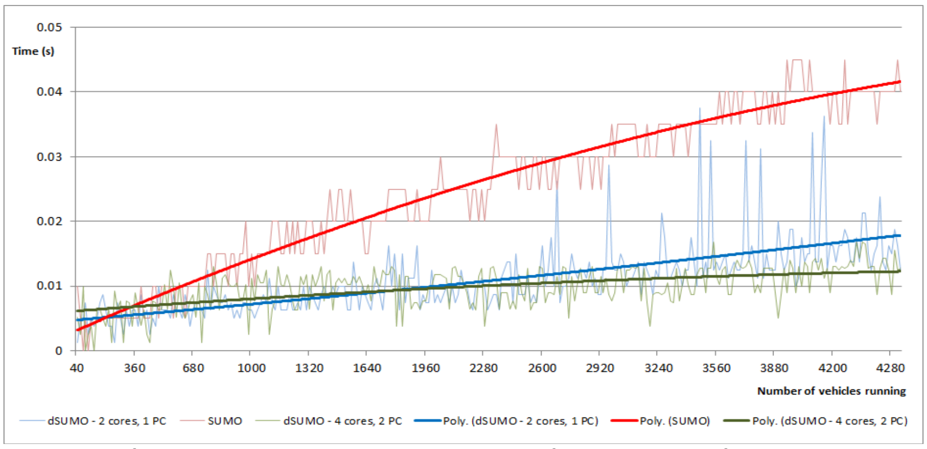


Figure 15-11: Evolution of time required per step depending of the number of cars present on the simulation.

Now, if we compare only the execution time of SUMO (reminder: every dSUMO node runs a SUMO instance) we have another perspective on the performance of dSUMO. Figure 15-11 shows the execution time of SUMO only for (i) SUMO (centralised solution) (ii) dSUMO when 4 nodes run on a single machine (iii) dSUMO when 4 nodes run on two remote machines. We can see that four instances of SUMO running on four different nodes outperform SUMO by an expected factor of about four. Which is very good as it means that for very large simulations SUMO would not scale while adding nodes to dSUMO will minimise the processing time (and remember that synchronisation is a constant). It was of course expected as the time needed by SUMO for each step is strongly dependant on the number of cars and follow a linear function. With an efficient partitioning algorithm, we can divide the time for each step by the number of nodes we use.

15.9 Conclusion

In this paper, we have presented dSUMO, a solution towards a distributed version of SUMO. We have listed the possibilities regarding road network partitioning, and defined an architecture where simple elements, e.g. cores of CPUs, can host running instances of SUMO. These nodes are linked to other nodes through sockets and exchange messages corresponding to vehicles transferred, vehicles that are close to the borders and may have an impact on the simulation, end of step, etc. There is no central entity in dSUMO, which makes it scalable and less prone to failures.

Our first experiments show that dSUMO is very accurate, although we have identified an issue that we need to fix before releasing our system: some vehicles are not created by the SUMO instances when they pass from one partition to another, likely for some safety reasons.

Communication is a bigger overhead than we expected and our small evaluations do not show that dSUMO definitely outperforms SUMO. However, if we look at execution time only, independently of synchronization process, then dSUMO reveals its nature: it is distributed and hence the load is balanced over several computing nodes. Which is of course what we expected and makes dSUMO promising anyway. Specially since we observed that synchronization is a constant and does not seem to vary greatly with the number of cars exchanged or the number of nodes.

We expect to be able to release soon a first version of dSUMO, for users feedback and so on.

15.10 Acknowledgement

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie)

15.11 References

[1] Dublin city council workplace travel plan. http://www.dublincity.ie/RoadsandTraffic/Documents/DCC_Travel_Plan_v2_1[1].doc

[2] Barrus, J.W., Waters, R.C., Anderson, D.B.: Locales and Beacons: Efficient and Precise Support For Large Multi-User Virtual Environments. In: VRAIS, pp. 204–213 (1996)

[3] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: Sumo - simulation of urban mobility: An overview. In: SIMUL 2011, The Third International Conference on Advances in System Simulation. Barcelona, Spain (2011)

[4] Bhatt, S.N., Leighton, F.T.: A framework for solving VLSI graph layout problems. Journal of Computer and System Sciences 28(2), 300–343 (1984)

[5] DMSO: High Level Architecture Interface Specification Version 1.3 (1998). URL http://hla.dmso.mil

[6] Fujimoto, R.M.: Parallel and Distribution Simulation Systems, 1st edn. John Wiley & Sons, Inc., New York, NY, USA (1999)

[7] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)

[8] Habitat, U.: State of the world's cities 2010/2011 - cities for all: Bridging the urban divide. http://www.unhabitat.org/pmss/listItemDetails.aspx?publicationID=2917

[9] Hu, S.Y., Chen, J.F., Chen, T.H.: VON: A Scalable Peer-to-Peer Network for Virtual Environments. IEEE Network 20, 22–31 (2006)

[10] Joerer, S., Dressler, F., Sommer, C.: Comparing apples and oranges?: trends in ivc simulations. In: Proceedings of the ninth ACM international workshop on Vehicular inter-networking, systems, and applications, VANET '12, pp. 27–32. ACM, New York, NY, USA (2012).

[11] Karypis,G.,Kumar,V.:Afastandhighqualitymultilevelschemeforpartitioningirregular graphs. SIAM J. Sci. Comput. 20(1), 359–392 (1998)

[12] Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 49(2), 291–307 (1970)

[13] Macedonia, M.R., Zyda, M.J., Pratt, D.R., Brutzman, D.P., Barham, P.T.: Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments. In: VRAIS, pp. 2–10 (1995)

[14] Morgenroth, E.L.W.: Analysis of the economic, employment and social profile of the greater dublin region. http://www.dublinpact.ie/pdfs/profile.pdf (2001)

[15] Pothen, A., Simon, H.D., Liou, K.P.: Partitioning sparse matrices with eigenvectors of graphs. SIAM Journal on Matrix Analysis and Applications 11(3), 430–452 (1990)

[16] Prasetya, K., Wu, Z.D.: Performance Analysis of Game world Partitioning Methods for Multiplayer Mobile Gaming. In: ACM SIGCOMM Workshop on Network and System Support for Games, pp. 72–77 (2008)

[17] Srinivasan, S., Supinski, B.R.D.: Multicasting in DIS: A Unified Solution. In: Electronic Conference on Scalability in Training Simulation (1995)

[18] Steed, A., Abou-Haidar, R.: Partitioning Crowded Virtual Environments. In: VRST, pp. 7–14 (2003)

[19] Tanenbaum, A.S., van Steen, M.: Distributed systems - principles and paradigms (2. ed.). Pearson Education (2007)

[20] Van Hook, D.J., Rak, S.J., Calvin, J.: Approaches to RTI Implementation of HLA Data Distribution Management Services. In: Workshop on Standards for the Interoperability of Distributed Simulations (1997)

[21] Van Hook, D.J., Rak, S.J., Calvin, J.O.: Approaches to Relevance Filtering. In: Workshop on Standards for the Interoperability of Distributed Simulations, pp. 26–30 (1994)

[22] Ventresque, A., Bragard, Q., Liu, E., Nowak, D., Murphy, L., Theodoropoulos, G., Liu, Q.: Spartsim: A space partitioning guided by road network for distributed traffic simulations. In: DS-RT. Dublin, Ireland (2012)

[23] Yu, A.P., Vuong, S.T.: MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In: workshop on Network and oper- ating systems support for digital audio and video, pp. 99–104 (2005)

16 Microscopic traffic simulation for evaluation of a cooperative variable speed limit system

Ellen Grumert, Andreas Tapani;
Swedish National Road and Transport Research Institute (VTI) and Linköping University, Department of Science and Technology (ITN), Linköping, Sweden

16.1 Abstract

Variable Speed Limit Systems (VSLS) where gantries are showing speed limits based on, for example, traffic volume or other road or traffic conditions exists on motorways in many countries. The aim of the VSLS is usually to improve traffic efficiency. Lately, cooperative intelligent transport systems allowing for communication between vehicles and/or vehicles and the infrastructure has received increasing interest. A cooperative VSLS, utilizing communication between vehicles and the infrastructure, could possibly result in further improved traffic efficiency and decreased exhaust emissions and fuel consumption. In this paper, a cooperative VSLS is evaluated by the use of a microscopic traffic simulator. The cooperative VSLS is described together with a discussion on how the modeling of the cooperative VSLS and the assumptions made regarding vehicles behavior will impact the final results of the evaluation. Results from the simulations are presented with in terms of traffic efficiency and exhaust emissions.

Keywords: microscopic traffic simulation, SUMO, Variable Speed Limit Systems (VSLS), cooperative systems, environmental impacts, traffic efficiency

16.2 Introduction

Intelligent Transport Systems (ITS) implemented as a part of the infrastructure, such as variable speed limit systems, or as autonomous systems in the vehicles, such as intelligent speed adaptation and adaptive cruise control, are commonly suggested to improve traffic conditions on the roads across Europe, and in the rest of the world. For ITS to provide solutions to problems related to safety, efficiency and environmental impacts require good system designs and widely deployed systems. Appropriate evaluation methods are therefore of high importance. A number of different evaluation methods exist and the choice of method is dependent on the purpose with the evaluation. Field-Operational-Tests (FOTs), with ordinary drivers driving ITS-equipped vehicles in real-world environments, are used for evaluation of driver behavior and vehicle dynamics. Driving simulation studies feature ordinary drivers to test ITS with respect to driver behavior in a simulated environment. Interviews, focus groups etc., used together with the before mentioned evaluation methods or as standalone evaluation methods, are used for evaluation of users' attitudes towards ITS. Many of the existing methods have to be performed on a large scale in order to give reliable results, leading to great expenses.

One less costly alternative is to use traffic simulation tools to estimate the effects of implementing a new ITS. Microscopic traffic simulation where vehicles' movements are simulated on the basis of core models, car-following and lane-changing, in a built up traffic environment, is a commonly used simulation method. By using microscopic traffic simulation, testing of different scenarios and different shares of vehicles equipped with the new proposed ITS is made possible. Microscopic traffic simulation is useful for estimating potential effects of a future ITS early in the development process, before the system is actually implemented. Although microscopic traffic simulation tools have shown to be useful for evaluating ITS, it is a simplified version of the reality that is described by the models. The design of the models implemented and the assumptions made have great impacts on the final results and needs to be treated carefully.

Today, an active area of research and development in ITS are cooperative systems that utilize communication between individual vehicles and the infrastructure. Many of the benefits of cooperative ITS are created by the possibilities to give individual information and guidance to the vehicles. Microscopic traffic simulation is of special interest when estimating effects of cooperative ITS since it allows modeling of individual vehicle behavior.

In this paper, we study impacts of a cooperative Variable Speed Limit System (VSLS) by the use of the microscopic traffic simulator SUMO [1]. The implementation of the cooperative VSLS in SUMO is described, together with an investigation of how the core models of the simulation, such as lane-changing and car-following, will influence the results of the simulation based evaluation of the cooperative VSLS. Assumptions made in the modeling process are presented, together with a discussion on how these assumptions have an effect on the simulation results in terms of traffic efficiency and environmental impacts. The contribution of the paper is an illustration of how model properties influence the results of the microscopic traffic simulation based evaluation of the cooperative VSLS.

The reminder of the article is organized as follows. In the next section, a short literature review is presented of evaluations of ITS using microscopic traffic simulation. A review of results from evaluations of already implemented VSLS is also given. In section 3, the design of the studied cooperative VSLS is described. Section 4 presents the simulation tool SUMO including the core models and review of previous studies using SUMO. How the proposed cooperative VSLS is implemented in SUMO is described in section 5. Section 6 gives the results from simulation experiments comparing the cooperative VSLS to a standard VSLS with respect to exhaust emissions and traffic efficiency. In section 7, effects of the assumptions made in the modeling process are discussed. Finally, in section 8 conclusions from the study is given together with suggestions for further research.

16.3 Literature review

Numerous studies exists using microscopic traffic simulation to evaluate effects of ITS. Both in-vehicle applications and roadside ITS have been considered in the studies, see e.g. [2] and [3]. In [4], an investigation is done of how ITS applications could be modeled in a microscopic traffic simulator. Ramp metering algorithms, as well as a bus priority strategy for signalized junctions are considered. An evaluation of the system performance of a variable speed limit system is done in [3], and in [5] different strategies for reducing high crash risk situations for a variable speed limit system have been evaluated. In [6] three different ITS measures have been investigated with the purpose of improving the traffic conditions in South Africa. Cooperative systems have been evaluated both with respect to the effects the systems have on the traffic flow, and with respect the communication solution used to send information to the vehicles. The impacts on traffic flow characteristics when using a cooperative adaptive cruise control is investigated in [2]. Coupling of a network simulator and a traffic simulator has been done in order to get a realistic approach for simulating the transmission of information between vehicles and vehicles and the infrastructure. Effects on the road traffic as a result of the way the transmission of information is simulated can also be evaluated with this approach [7].

VSLS exists on many roads across Europe today and earlier studies of already implemented VSLS have showed that they have a potential to improve traffic efficiency and road safety [8]. Harmonized flows is one of the main benefits with the system [9]. There are also indications of reduced environmental impacts due to VSLS [10].

16.4 A cooperative VSLS

The proposed cooperative VSLS makes use of an existing VSLS with gantries showing recommended, or compulsory, variable speed limits depending on the specific traffic conditions on the road stretch covered by the system. The difference from the existing VSLS is that the cooperative VSLS makes use of I2V communication, so that the recommended or compulsory variable speed limits, normally shown on the gantries, are communicated to the drivers via an in-vehicle application. In this way the drivers can get more detailed information about new traffic conditions on the road before the gantry becomes visible, allowing them to act on a speed change at an earlier stage in time. This may result in more harmonized flows, and therefore increased energy efficiency, and reduced environmental impacts of motorway traffic, compared to the effects of today's VSLS.

The VSLS function included in the Stockholm Motorway Control System (MCS) and the Dutch MCS [11] is used as reference system for the proposed cooperative VSLS. The VSLS function consists of detectors measuring the current traffic conditions such as average velocity on the road, and gantries showing recommended or compulsory speed limits based on the traffic conditions. If the mean speed goes below a certain threshold, new lower speed limits are shown on the gantries. Not only the gantry closest to the detection point show a lower speed limit, a number of gantries further upstream are also showing lower speed limits. The speed limits shown on gantries upstream of the detection point are higher than the speed limit shown on the gantry closest to the detection point to avoid sudden changes in speed limit from one road segment to the next.

In the existing VSLS, the gantries are used for showing drivers information about the speed limits. In the cooperative VSLS, the gantries function as roadside units sending out information to the individual vehicles about speed limits via a V2I communication channel. The speed limits given to the vehicles are calculated based on the distances that the vehicles are from the new speed limit, the current speed of the vehicles and the reference speed. As a result of this approach, when the vehicles reaches the new speed limits they have been given speed recommendations at predefined time intervals resulting in a smooth deceleration/acceleration towards the new speed limit point. The speed limits for the VSLS and the cooperative VSLS are assumed to be compulsory. The differences between the standard VSLS and the cooperative VSLS can be summarized in the following bullets:

- With the cooperative VSLS, information about the variable speed limit is received at an earlier point in time.
- With the cooperative VSLS, vehicles receive individual speed limits determined by their current speed and position.

The proposed cooperative VSLS is illustrated in Figure 16-1. The arrows indicate how the variable speed limit information is communicated to the vehicles.

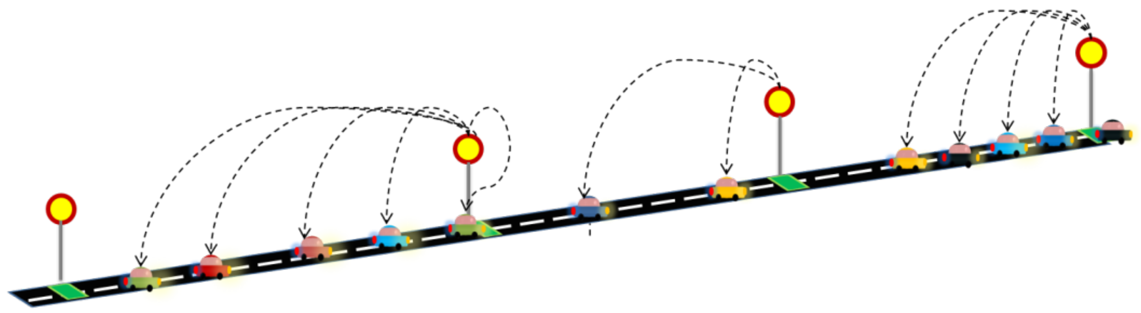


Figure 16-1: Illustration of the VSLS with communication between vehicles and infrastructure.

For both the VSLS and the cooperative VSLS the maximum allowed speed on the road is assumed to be 120 km/h. The average velocities given at the detectors are calculated by the use of a smoothed harmonic average speed [11],

$$\frac{1}{\tilde{v}_{t+1}} = \alpha \cdot \frac{1}{v_{measured}} + (1-\alpha) \cdot \frac{1}{\tilde{v}_t},$$

where $\tilde{v}_t$ is the calculated mean speed at time t, $v_{measured}$ is the measured speed at the detectors and α is the smoothing parameter. A smoothing parameter of 0.25 used to take into account approximately the 4 last vehicles.

If the average velocity at a detector goes below a certain threshold, which is set to 45 km/h, the speed limit is updated. The speed limit at the gantry where the detected average velocity is below the threshold is set to 60 km/h and the following two gantries upstream of this detector are set to 80 km/h and 100 km/h, respectively. If the average velocity at a detector where the gantry is showing 60 km/h goes above a certain threshold, which is set to 55 km/h, the compulsory speed limit is updated to 120 km/h. The speed limits and the thresholds are based on a study of the existing MCS and the new speed limit system that is currently being implemented in Sweden [12]. It is assumed that the most restrictive lane is determining the speed limit, i.e. the lane with the lowest average velocity is considered when calculating the speed limit for all lanes. The speed limits are updated every fourth second.

For the cooperative VSLS, the speed limits that are continuously calculated based on the smoothed harmonic average speed, are used as reference speeds to determine speed recommendations that are given to the drivers at specific points in time. The vehicles receive updates of the speed limit information via communication with roadside units during the whole road segment between consecutive gantries (i.e. 500 meters). The speed recommendations given to the vehicles are calculated based on the acceleration needed to reach the speed limit at the position of the gantry. First the acceleration of the vehicles, needed to adapt to the given speed limit when reaching the next downstream gantry is calculated according to

$$a_{t,i} = \frac{v_{t,j}^2 - u_{t,i}^2}{2 \cdot s_{t,ij}}, \tag{1}$$

where $v_{t,j}$ is the speed communicated from roadside unit j (located in front of vehicle i) at time t, $u_{t,i}$ is the speed of vehicle i at time t, $s_{t,ij}$ is the distance between vehicle i and roadside unit j (located in front of vehicle i) at time t and $a_{t,i}$ is the acceleration of vehicle i at time t needed to adapt to speed limit, $v_{t,j}$, when located at a certain place, $s_{t,ij}$, on the road segment.

The acceleration given by Equation 1 is then used to calculate a recommended speed,

$$u_{t,i}^{rec} = u_{t,i} + a_{t,i} \cdot T, \tag{2}$$

where $u_{t,i}$ is the speed of vehicle i at time t, $a_{t,i}$ is the acceleration calculated above, and T is the time until next update of the speed recommendation.

With recommended speeds given by Equation 2 slow moving vehicles could appear when a queue is dissolved and the vehicles' current speeds becomes very low. Also, in Equation 1 the acceleration needed for a vehicle with speed, $u_{t,i}$, and distance, $s_{t,ij}$, from the roadside unit, to have the recommended speed, $v_{t,j}$, is calculated. This is independent of time. Therefore when the recommended speed is calculated by looking time T into the future the speed could be above the maximum allowed speed on road, leading to speeding vehicles. In order to avoid very slow moving vehicles and speeding vehicles the final recommended speed is given by,

$$u_{t,i}^{frec} = max\left(u_{min}, min\left(u_{t,i}^{rec}, u_{max}\right)\right), \tag{3}$$

where $u_{t,i}^{rec}$ is the recommended speed calculated from Equation 2, u_{min} is the minimum recommended speed, and u_{max} is the maximum recommended speed.

It is assumed that all vehicles receive the information sent out from the roadside units, regardless of their position within the road segment and that the given speed recommendations are obeyed.

When setting the time until next update of the recommended speed it is important that the update interval is comfortable for the drivers and also that no big jumps in speed between two updates occur, resulting in unnecessary sudden accelerations or decelerations. In this initial experiment the update time interval is set to 1 second. While it is a small interval that may affect the driver attention negatively due to speed recommendations of high frequency, a smooth increase/decrease of the speed can be achieved, except when there is a change in the speed limit. One possibility in order to avoid this issue could be to integrate the variable speed limits with an adaptive cruise control, allowing the vehicle itself, rather than the driver, to adapt to the speed limits.

16.5 The microscopic traffic simulator - SUMO

The open source microscopic traffic simulation tool Simulation of Urban Mobility, SUMO ([1, 13]), is used to model the VSLS and the cooperative VSLS. The main purpose when developing the tool has been, and is, to have a common simulation platform to get comparable results when used among different users, and where it is easy to change and understand the code. SUMO is distributed as open-source code which makes it easy to extend the model. New applications and development of existing models are done continuously with much contribution from the users of the tool. The current version of SUMO is multi-modal, space continuous and time discrete. The coding is done in C++. A Traffic Control Interface (TraCI) enables communication between SUMO and an externally developed script making it possible to incorporated new applications in the simulation. The simulation can be viewed in runtime through a Graphical User Interface (GUI) or by looking at the results afterwards (excluding the visable option). XML-descriptions are used for the network building and the demand, and can either be designed directly in SUMO or imported from for example VISSUM, Vissim, Shapefiles, OSM, etc., and converted to XML-descriptions. To evaluate the environmental effects of the simulated traffic in SUMO, the HBEFA [14] emission factors are applied.

16.5.1 Previous applications

SUMO has been continuously developed since 2000 and several projects, such as the European projects iTETRIS [15] and DRIVE C2X [16], have used SUMO over the years. In [7] an investigation of how integration of a network simulator and SUMO could be used to realistically simulate the transmission of information between vehicles and vehicles and the infrastructure in urban areas. The coupling of simulators could also be used for evaluation of the effects the simulated communications between vehicles and vehicles and the infrastructure will have on the road traffic. Two different inter vehicle communication protocols used for incident warnings in vehicle ad hoc networks is evaluated in [17] by using a network simulator together with SUMO. SUMO has also been frequently used for evaluation of traffic light control systems, see e.g. [18] where an agent based traffic light control algorithm was implemented and evaluated by the use of SUMO. Other systems have also been investigated such as platooning of autonomous vehicles by assuming inter vehicle communication, see e.g. [19]. This application was done by changing the car-following model used in SUMO to allow for platooning. No network simulation was used since it was assumed that all vehicles in the platoon can receive the information sent.

16.5.2 Core models

Car-following and lane-changing are two fundamental models included in most microscopic traffic simulation models. In SUMO the default car-following model is a space continuous, time discrete model based on the Gipps-model, developed and described by [20]. The idea is that the final speed in which a vehicle travels in is the maximum of: (1) the maximum speed the vehicle can drive in, or are allowed to drive in, v_{max}, and (2) a safe speed, v_{safe}, which allows the vehicle to drive in a safe way without collisions. The model is taking into account the fact that the acceleration and deceleration has some lower and upper bound, i.e. $(-b \leq \frac{dv}{dt} \leq a,\ a, b > 0)$.

In the model developed by [20] it is, as a difference from the original Gipps-model, taken into account that the driver is not perfect in holding the desired speed. The drivers' imperfection is modeled as a stochastic deceleration. This means that when the desired speed has been calculated the final speed the considered vehicle will keep in the next time step is calculated as

$$v(t) = max(0, v_{des} - ra\epsilon),$$

where v_{des} is the desired speed at time t, ϵ is the driver imperfection for the vehicle under consideration, r is a random number between 0 and 1, and a is the maximum acceleration of the vehicle under consideration.

The desired speed is updated as

$$v_{des} = min\big(v_{safe}, v(t-1) + a\Delta t, v_{max}\big), \qquad (4)$$

where $v(t)$ is the velocity of the vehicle under consideration at time t, Δt is the time step, v_{max} is the maximum velocity for the vehicle under consideration and v_{safe} is the safe velocity in which the vehicle under consideration can drive in a collision-free and safe way.

The safe velocity (as described in [13]), v_{safe}, at time t, is derived from the assumption that the vehicle under consideration has the ability to stop before colliding with another vehicle,

$$v_{safe}(t) = -\tau b + \sqrt{(\tau b)^2 + v_{leader}(t-1)^2 + 2bg_{leader}(t-1)},$$

where $v_{safe}(t)$ is the safe velocity for the vehicle under consideration at time t, τ is the driver's reaction time for the vehicle under consideration, b is the maximum deceleration ability for the vehicle under consideration, $v_{leader}(t)$

is the velocity of the vehicle in front of the vehicle under consideration at time t, $g_{leader}(t)$ is the gap between the vehicle under consideration and the vehicle in front at time t. $g_{leader} = x_{leader} - x_{follower} - l$, where x_{leader} is the leaders position, $x_{follower}$ is the position of the considered vehicle and l is the length of the considered vehicle.

The braking distance and a reaction time is also taken into consideration, giving the following condition on the gap between the vehicles

$$d\big(v_{follower}\big) + v_{follower}\tau \leq d(v_{leader}) + g_{leader},$$

where $d(v)$ is braking distance described as a function of the vehicles velocity, $v_{follower}$ is the velocity of the considered vehicle and τ is the driver's reaction time .

In SUMO the follower and the leader are assumed to have the same maximum deceleration abilities and the braking distance is approximated by the function,

$$d(v) = \frac{v^2}{2b}.$$

Some further extension has been made to the original Gipps-model:

- The ability of acceleration is decreasing with increasing speed (modeled using a linear function)
- The driver's imperfection is reduced when accelerating from low velocities (two cases is considered depending on the vehicle's speed)

The car-following model is described and discussed in detail in [20] and [13].

The other main part in a micro simulation model is the lane-changing model. The lane-changing model has been continuously developed since the first version of SUMO. The current model takes into account the drivers' route and lane changing requirements in order to be able to continue on the route to the vehicle's given destination.

First, the distance needed for the vehicle to be able to change lane is calculated, taking into account if the vehicle is driving with a behavior corresponding to highway or urban traffic environments, as

$$d_{lanechange,i}(t) = \begin{cases} v_i(t)\alpha_1 + 2l_i, & if\ v_i(t) \leq v_{threshold}, \\ v_i(t)\alpha_2 + 2l_i, & othervise, \end{cases}$$

where $d_{lanechange,i\,(t)}$ is the distance vehicle i needs to change lane at time t , $v_i(t)$ is the speed of vehicle i at time t, $v_{threshold}$ is the threshold used to separate highway and urban behavior, currently set to $14\,m/s$ in SUMO, α_1 and α_2 are scaling factors, currently set to 5 resp. $15\ s$ in SUMO, l_i is the length of vehicle i.

A benefit of changing the lane is also calculated based on the safe speed on the current lane and the safe speed on the lane considered as a possible lane change opportunity,

$$b_{l_n}(t) = \frac{v_{poss}(t, l_n) - v_{poss}(t, l_c)}{v_{\max}(l_c)},$$

where $b_{l_n}(t)$ is the benefit of changing to lane l_n at time t, l_n is the neighboring lane, l_c is the current lane, $v_{poss}(t, l)$ is the safe velocity for the vehicle on lane l at time t and $v_{\max}(l)$ is the maximum velocity (on free flow) the vehicle can drive in on lane l.

The benefit is saved as a memory variable used to determine the vehicles desire to change lane. If the benefit for a specific vehicle at time t is greater than 0 the calculated benefit of changing lanes is added to the current value of the memory variable, increasing the vehicles desire to change to the specific lane, and if the benefit at time t smaller than 0 the current value of the memory variable is divided with 2, decreasing the desire to change to the specific lane. The vehicle is changing lane if the memory variable reaches over a certain threshold. If the vehicle has no option to change lane due to not enough space on the neighboring lane in order to move in a safe collision free way the vehicle starts to interact with the vehicles in the front and the back according to,

$$v_{next}(t) = \begin{cases} v_{decel}(t), & if\ blocking/blocked\ \ at\ own\ back\ and\ front, \\ v_{decel}(t), & if\ \ blocking/blocked\ \ at\ own\ front, \\ v_{accel}(t), & if\ \ blocking/blocked\ \ at\ own\ back, \end{cases}$$

where $v_{next}(t)$ is the speed after adaption to vehicles in from and in back of the vehicle under consideration, $v_{decel}(t)$ is the speed of the vehicle under consideration after decelerating, $v_{accel}(t)$ is the speed of the vehicle under consideration after accelerating. The acceleration and deceleration speed is calculated according to,

$$v_{accel}(t) = v_{cf}(t) + \frac{v_{max}(t)}{2},$$

$$v_{decel}(t) = v_{cf}(t) + \frac{v_{min}(t)}{2},$$

where $v_{cf}(t)$ is the speed at time t given by the car-following model (including imperfection), $v_{max}(t)$ is the speed at time t after maximum possible acceleration (in accordance with the car-following model), $v_{min}(t)$ is the speed at time t after maximum possible deceleration (in accordance with the car-following model) .

One problem with this model according to [13] is that when the vehicle is approaching a queue that is standing still and the lane must be used to continuing the route, the vehicle does not change into another lane in order to pass the queue.

16.6 Implementation of a cooperative VSLS in SUMO

The VSLS implemented in SUMO is a simplified version of the system that exists on the E4 in Stockholm. The cooperative VSLS is implemented based on the same assumptions as the existing VSLS, such as calculations of average velocity, location of gantries, etc., together with additional assumption taking into account the functionalities related to the cooperative part. The cooperative VSLS requires that the variable speed limits are continuously updated and communicated to the vehicles. By using SUMO’s Traffic Control Interface (TraCI), vehicle specific data can be accessed during runtime, allowing for single vehicle control during the simulation. The VSLS algorithm and the additional cooperative part are implemented in python script, and the final new maximum allowed speed, needed for calculating the desired speed in the car following model (see Equation 4) is communicated to SUMO via the TraCI. See Figure 16-2 for an illustration of the structure.

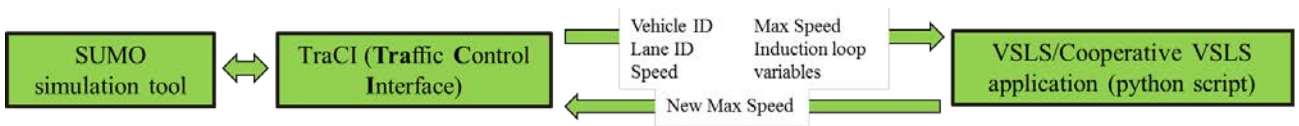


Figure 16-2: Illustration of how SUMO's Traffic Control Interface (TraCI) is used to model the VSLS and the cooperative VSLS.

An open traffic system consisting of one stretch of motorway, without on- or off-ramps, is used to simulate the VSLS and the cooperative VSLS. The reason for choosing this elementary design is to be able to easily evaluate the effects cooperative systems technology will have on an existing VSLS. The simulated road is divided into 500m segments, with one detector and one gantry per lane on each segment. See Figure 16-3 for an illustration of the considered VSLS and cooperative VSLS.

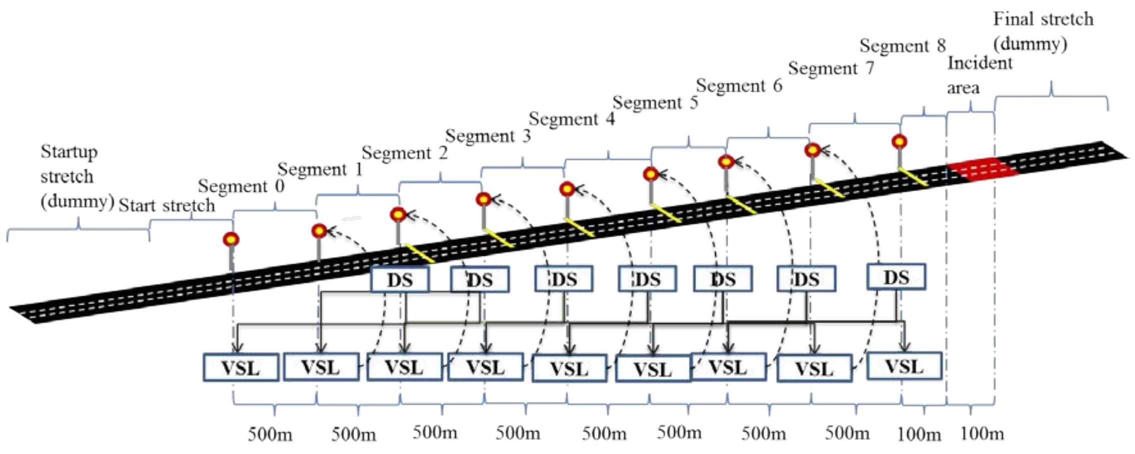


Figure 16-3: Illustration of the considered VSLS and cooperative VSLS (VSL: Variable Speed Limit, DS: Detector System).

The simulation is performed during a half-an-hour interval. The flow during that period is held constant at 4400 vehicles per hour, which is around 70% of the flow at capacity level. In order to get an activation of the VSLS and the cooperative VSLS an incident is modeled 100 meters upstream the last detector station by decreasing the speed at a 100 meter long segment, to around 25 km/h after 10 minutes. The speed decrease is active for 10 minutes. The effect of this is that the vehicles are slowing down and a queue starts to build upstream of the incident area reaching one or more of the detector stations. This will activate the system giving a picture of how the two systems behave during congested conditions and a comparison between the two systems can be made.

When implementing the VSLS it is assumed that all vehicles follow the given speed limits and that the vehicles are able to read the speed limits shown on the gantries within a range of 150 meters. Approaching vehicles are updating their maximum allowed speed when located within the given range

Two different scenarios are modeled for the cooperative VSLS. In the first scenario the cooperative VSLS algorithm (see Equation 3) is applied for both increase and decrease in the recommended speed. In the second scenario the cooperative VSLS is applied only for the decrease of recommended speed, and for the increase of recommended speed, i.e. when the queue is dissolved and vehicles can drive in the maximum speed again, the vehicles are given the maximum allowed speed for the whole stretch. The reason for doing this is to be able to take advantages of getting information earlier in the cooperative VSLS case and to be able to possibly dissolve the queue faster. In the first scenario a smooth acceleration is seen as a more important factor.

The vehicle parameters used in the simulation are set to standard values used in SUMO version 0.16.1 (see [1] for a detailed documentation of the parameters used), except for the speed deviation and the speed factor. The speed factor is set to 1.1 and the speed deviation is set to 0.1 to get a more realistic behavior, resulting in vehicles having different preferences of how much above or below the roads' legal speed limits they want to drive. The randomness of the simulations performed is taken under consideration by performing 30 replications of the simulation for both the VSLS and the cooperative VSLS. The standard output generated from SUMO is used to evaluate effects on vehicle exhaust emissions. Acceleration and deceleration, and travel time output is based on raw data collected from SUMO during the simulation by use of the TraCI.

16.7 Results

Comparing the results from the two systems gives a good indication of if the systems behave similar or not. The assumptions and simplifications done in SUMO is done for both systems, except for the cooperative VSLS and VSLS specific parts, such as assuming readable speed limits on gantries from a distance of 150 meters for the VSLS and below, and that vehicles get continuous information in the cooperative VSLS case. The performance indicators considered are the density distribution of accelerations/decelerations, travel time, total amount of exhaust emissions and total amount of fuel consumptions within the simulation. The travel time is considered both with respect to travel time per 30-second intervals and with respect to the total travel time spent in the system. The exhaust emissions included in the results are CO_2, PM_x and NO_x.

By comparing the density distribution of accelerations/decelerations for the two systems a measure of the effectiveness of the systems is given. High values of acceleration/deceleration indicates strong acceleration and strong braking (deceleration), resulting in higher fuel consumption and excess exhaust emissions. The accelerations/decelerations are calculated after the simulation by using speed and time output, communicated from SUMO via TraCI. In Figure 16-4 the empirical probability density functions of accelerations/decelerations, resulting from the simulations, are shown. The empirical probability density functions are shown for the VSLS and for the two scenarios using cooperative VSLS.

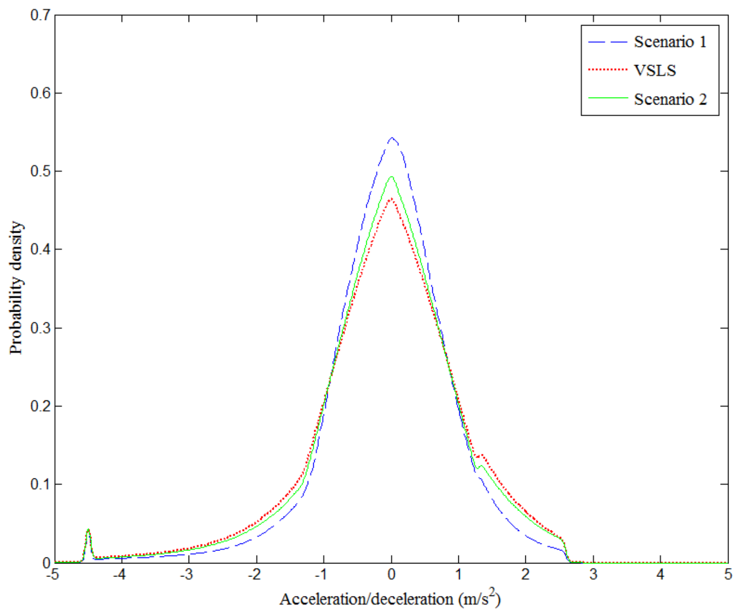


Figure 16-4: Empirical probability density function for the cooperative VSLS (two scenarios) and the VSLS.

It is clear from the figure that for the two cooperative VSLS scenarios the accelerations/decelerations are more centered around zero. The second scenario, where the increase of speed is faster due to using maximum speed as communicated recommended speed rather than the speed from the cooperative VSLS algorithm, give a lower amount of accelerations/decelerations centered around zero. This is intuitive since the vehicles accelerate to the maximum speed faster when given the maximum speed during the whole stretch compared to when the cooperative VSLS algorithm is used. The tails for the VSLS is also thicker than in the cooperative VSLS case, showing a larger amount of higher accelerations and decelerations. Also here the difference between scenario two for the cooperative VSLS and the VSLS is much smaller than when comparing scenario one for the cooperative VSLS and the VSLS. This is also intuitive. What is surprising is the fact that in scenario two for the cooperative VSLS the amount of higher values of decelerations is also higher. The reason for this might be that the behavior when the maximum allowed speed is active as recommended speed, the vehicles drives faster giving higher decelerations when a slower moving vehicle is approached. A two-sample Kolmogorov-Smirnov test, looking at the distance between the two empirical distributions, is used to evaluate if the difference between the density functions are significant. The comparison between the two scenarios for the cooperative VSLS and the VSLS shows that the distributions are indeed different, both for the VSLS and scenario one of the cooperative VSLS and for the VSLS and scenario two of the cooperative VSLS, $p<0.001$. The effect size is of course bigger between the VSLS and scenario two of the cooperative VSLS. In Figure 16-4 it can be seen that especially the amount of higher accelerations are bigger for the VSLS, and also for the cooperative VSLS scenario two (which is closer to the VSLS), indicating that in the case where vehicles are going from lower to higher speeds the VSLS system gives harder accelerations. This could also have an effect on the travel time since higher amounts of high accelerations would probably also lead to that vehicles travel the stretch faster. The travel time for one single vehicle is defined as the time at which the vehicle leaves the stretch minus the time the vehicle enter the stretch. A mean value is then given by taking the average over vehicles leaving the system within 30 second intervals. The travel times for the different scenarios are shown in Figure 16-5.

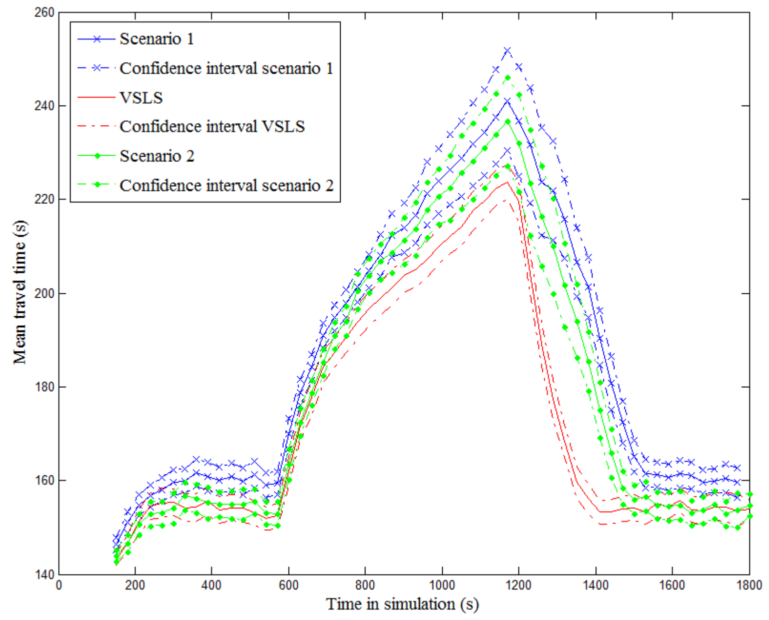


Figure 16-5: Travel time for the VSLS and the two scenarios of the cooperative VSLS, together with upper and lower bounds of the 95 % confidence intervals.

When looking at the travel time for the two scenarios of the cooperative VSLS and the VSLS differences can be seen. As expected the cooperative VSLS gives higher travel times for both scenarios than the VSLS during the period when the systems are active. The difference in travel time between scenario two of the cooperative VSLS and the VSLS is smaller than the difference in travel time between scenario one of the cooperative VSLS and the VSLS. This is also in line with the expectations.

Also by looking at the total time spent (TTS) in the systems, see Table 16-2, it can be concluded that the VSLS is more time effective than the cooperative VSLS, both compared with scenario one and scenario two. TTS in scenario two for the cooperative VSLS is in between TTS in scenario one for the cooperative VSLS and TTS for the VSLS. This is also in line with the expectations.

Table 16-2: Total time spent in the system by all vehicles for the VSLS and the two scenarios for the cooperative VSLS.

	Mean TTS (h)	**Confidence interval**
VSLS	95.1	[94.6, 95.7]
cooperative VSLS scenario 1	102.0	[101.4, 102.6]
cooperative VSLS scenario 2	99.1	[98.5, 99.6]

The larger frequency of high acceleration and deceleration rates for the VSLS case indicate more frequent hard braking and strong acceleration, which most probably will result in higher exhaust emissions and increased fuel consumption. The HBEFA-based emissions and fuel consumption calculated in SUMO gives an indication of if this is the case or not. In Table 16-3 it can be seen that the cooperative VSLS for both scenarios result in lower fuel consumption than the VSLS. The same can be seen for CO2, PMx and NOx emissions in Table 16-4. By looking at the 95 % confidence intervals (assuming normally distributed simulation results) it can be seen that the intervals are not overlapping, indicating a significant difference between the two systems for both scenarios.

Table 16-3: Total amount of fuel consumption and exhaust emissions during the whole simulation for the VSLS and the two scenarios of the cooperative VSLS. cooperative VSLS(1) refers to scenario 1 for the cooperative VSLS and cooperative VSLS(2) refers to scenario 2 for the cooperative VSLS.

	VSLS	**cooperative VSLS(1)**	**Difference cooperative VSLS(1) vs. VSLS**	**cooperative VSLS(2)**	**Difference cooperative VSLS(2) vs. VSLS (%)**
Total amount of Fuel (l)	625.1	554.3	70.8 (11.3%)	608.7	16.4 (2.6%)
Confidence interval	[623.5, 626.6]	[553.8, 554.9]		[607.6, 609.8]	
Total amount of CO_2 (kg)	1567.8	1390.4	177.4 (11.3%)	1526.7	41.1 (2.6%)
Confidence interval	[1564.0, 1571.6]	[1389.1, 1391.7]		[1524.0, 1529.5]	
Total amount of PM_x (g)	164.5	137.0	27.5 (16.8%)	158.2	6.3 (3.8%)
Confidence interval	[163.9, 165.1]	[136.8, 137.1]		[157.8, 158.6]	
Total amount of NO_x (g)	3476.0	3014.3	461.7 (13.28%)	3363.2	112.8 (3.3%)
Confidence interval	[3467.6, 3484.4]	[3011.9, 3016.7]		[3357.2, 3369.2]	

16.8 Discussion

When implementing the VSLS system in SUMO simplifications have to be made along with assumptions in order for the traffic to behave as desired. Some of the assumptions and simplifications are related to the functionalities of the system to be studied, while others are related to models and functionalities used in SUMO. Some of the most important remarks are discussed here.

The VSLS algorithm implemented in Stockholm today is based on a simple algorithm calculating the new recommended speed based on the current conditions on the road looking at only average speed. This means that only the conditions at the detector stations are considered and a quite long queue could therefore build up before the system is activated, prolonging the time until the queue is dissolved. This can also be seen in the simulation, especially since the modeled incident starts 100 meters from the first detector station. When looking at the simulation graphically and by investigating the variable speed limits, a delay of around 30 seconds from the time where the actual slowdown of vehicles occurs to when it is detected by the system can be seen. It is clear that this is due to the fact that incidents can only be detected at certain locations in the network. But also the simple design of the VSLS algorithm, detecting incidents only based on a threshold looking at mean speed might be a reason for late detection of the incident. It is unclear, without implementing other types of algorithms, if this is the case, and also if a change of variable speed limit algorithm would influence both systems in the same way.

Another concern related to the system design is the algorithm used to calculate the cooperative system's recommended speeds. This is a first relatively simple algorithm only taking into account the vehicles' speed, the speed shown on the gantry and the distance between the two. A more sophisticated model might result in larger effects. The benefit with this model is that it is simple but yet has an effect on both environmental impact indicators and the frequency of high and low accelerations.

The simulations featured a homogenous flow. The vehicle speeds are drawn from a single speed distribution and all vehicles have the same size. This gives a uniform behavior of the vehicles in the simulation. In SUMO assumptions regarding the vehicles could be set different allowing for a more heterogeneous flow, with for example additional slow-moving vehicles included to represent more realistic traffic. Also, the estimation of environmental impacts is based on one type of vehicles according to HBEFA and this might not be representative for the real world where different type of vehicles produces very different level of exhaust emissions. The effect of this in SUMO is that the flow on the road is more harmonized from the beginning leading to a more harmonized behavior. But when a comparison between the cooperative VSLS and the VSLS is made, the composition of vehicles may be of lesser importance since it is the relative change from one system to another that is of interest in the evaluation. The question is if the change in effect is higher with more homogenous flow or not.

In SUMO the vehicles are assigned a speed factor and a speed deviation in order to get a more realistic flow, where a final vehicle specific speed multiplier is drawn from a normal distribution and multiplied with the maximum allowed speed on the road. This results in that a specific vehicle can have a desired speed that is lower or higher than the speed limits assigned to the road. When modeling the VSLS and the cooperative VSLS new recommended speed limits are calculated and stored in the TraCI python script, with input from SUMO (current speed of the vehicle and current position on the road). The final recommended speeds are communicated back to SUMO via TraCI as a maximum allowed speed for each vehicle, and not by setting the new speed limits on the road equal to the calculated speed recommendations. This is to allow for specific recommendations for each vehicle in the cooperative VSLS case. The maximum allowed speed is used in the calculation of the desired speed in the car following model (see Equation 4). In SUMO the maximum allowed speed assigned to a vehicle cannot be overruled. This means that even though the vehicles behave according to a given speed distribution, where the vehicles are assigned a specific speed multiplier the speed is truncated at the maximum allowed speed. Therefore the behavior of the vehicles might be different from what is normally observed in the real world.

The distance in which the vehicles can read the speed limit shown on the gantry when applying the VSLS is assumed to be 150 meters. In SUMO this means that when a vehicle arrives to an area where the speed limit on the gantry is visible the vehicle changes its speed accordingly. A new maximum allowed speed is communicated to each vehicle in SUMO and used as input in the car-following model, as described above. One other possibility would be to construct the road in such a way so that the stretch of 500 meters are divided into two links, one with the length of 350 meter and one with the length of 150 meter. The speed limit for the vehicles within the 150 meter stretch is then changed simultaneously. This would solve the issue with truncating the desired speed at the maximum allowed speed and instead use the distribution for maximum allowed speed on the road with a specific speed multiplier for each vehicle. The reason for not doing this is that the modeling assumptions for two systems would be different. The cooperative system would take the speed limits into account by assigning each vehicle a maximum allowed speed equal the speed limit, truncating the vehicles maximum speed exactly at the speed limit, and the VSLS would assign each vehicle a specific maximum allowed speed drawn from a speed distribution, in a way that some vehicles would have maximum allowed speeds above the speed limit. A between system comparison would therefore become difficult.

When applying the cooperative VSLS, the vehicles' are not immediately conforming to the new speed limit but are attaining the recommended speed at the location of the next downstream gantry. This is in line with the MCS system implemented in Stockholm today where recommended speed are shown instead of compulsory speed limits and the only speed limit that has to be followed strictly is the actual speed limit on the road when the system is not active. This results in higher speeds than the given speed recommendations. Consequently, given the assumptions made in the modeling of the cooperative VSLS, the simulation results in lower speeds than in a real system with recommended speeds. Of course one also have to consider the fact that the drivers' probably does not want to get to big fluctuations in the recommendations from one time to another so some discrete limits should probably be used. In this study, differences in speed recommendations are ignored if the difference between the equipped vehicle's speed and the recommended speed is less than 0.5 km/h. This is a small difference and larger differences could be considered to get a system that is more likely to be accepted by the drivers in reality. This is not taken into account in this study.

When looking at the different scenarios for the cooperative VSLS it is clear that decreased travel time comes with a price. Increasing the possibility to faster accelerations also increases the emissions and the amount of high accelerations and decelerations. Although the difference between the VSLS and scenario two for the cooperative VSLS is not that big with respect to acceleration/deceleration, emissions and travel time there is a difference and the question is what the main purpose with the system is.

It is also important to highlight that the lower frequency of high accelerations/decelerations seen for the cooperative VSLS could potentially lead to fewer accidents compared to the standard VSLS case.

To get the behavior of the simulated traffic system to be as close as possible to reality there is a need for model calibration. However, when comparing two systems with the same assumptions and simplifications, as is the case in this paper, the need for calibration can be reduced. And also, when a calibration is performed it is done for one certain location and for a specific time interval. The conditions on the road might be different at other points in time and at other locations. This is especially the case when looking at randomly occurring incidents and congestion. Traffic conditions vary from one day to another without any obvious reason.

Another thing that is worth mentioning is that when using the TraCI, the simulation gets slowed down due to an extra step where calculations needs to be performed outside of SUMO. For the network in question and with the flow as described in this paper one simulation takes about three minutes to run. This could potentially be a problem if the network is extended.

16.9 Acknowledgement

This study is funded by the Swedish Transport Administration.

16.10 Conclusion

The microscopic traffic simulation tool SUMO has showed to be a useful tool for evaluating the cooperative VSLS presented in this paper, and by the use of TraCI additional studies of the cooperative system are made possible. The results, showing positive effects on the frequency of high accelerations/decelerations and the measures of environmental impact, indicate that cooperative VSLS can be applied for traffic flow harmonization to decrease environmental impacts. However, as the frequency of high accelerations/decelerations is decreased, travel time is increased. This can be a result of the variable speed limit algorithm not reflecting the actual flow conditions on the road. But it can also be the case that the cooperative VSLS will result in longer travel times independent of algorithm. The fact that the exhaust emissions are reduced significantly is a benefit of the cooperative VSLS. Also the smoother behavior detected in the cooperative VSLS case due to the lower frequency of high accelerations/deceleration is beneficial. This could also probably result in improved safety due to more harmonized vehicle speeds. A trade-off between increased travel time on one hand, and, on the other hand, lower emissions and smoother traffic flow, potentially leading to fewer accidents, needs to be considered.

The variable speed limit system is very limited by the structure of the control algorithm and other more advanced methods could probably give larger effects. Another aspect that needs to be considered is the assumptions and simplifications made in traffic simulation model. The more realistic traffic behavior, the better does the results reflect the reality.

Further investigations of the model assumptions together with extensions of the existing model have the potential to improve the results. One possible extension, for which greater impacts can be assumed, is to use a control strategy for the VSLS that respond better to the actual conditions on the road. In this case it will be important to take into account that control strategies focusing on improved efficiency, might give a decrease in safety. Another extension can be to make use of communication between the vehicles, possibly giving a more harmonized flow on a local and global basis. Also, extensions involving multiple cooperative systems, such as lane changing assistance and adaptive cruise control, could be considered.

16.11 References

[1] SUMO homepage, http://sourceforge.net/apps/mediawiki/sumo

[2] Van Arem, B., Van Driel, C. J. G. and Visser, R.: The impact of cooperative adaptive cruise control on traffic-flow characteristics. IEEE Transactions on Intelligent Transportation Systems 7 (4), 429-436 (2006)

[3] Allaby, P., Hellinga, B. and Bullock, M.: Variable Speed Limits: Safety and Operational Impacts of a Candidate Control Strategy for Freeway Applications. Intelligent Transportation Systems, IEEE Transactions on 8 (4), 671-680 (2007)

[4] Čapek, K., Pitkänen, J. P. and Niittymäki, J.: Evaluating the impacts of ITS applications using microscopic traffic simulators. Advances in Transportation Studies (25), 5-14 (2011)

[5] Lee, C., Hellinga, B. and Saccomanno, F.: Evaluation of variable speed limits to improve traffic safety. Transportation Research Part C: Emerging Technologies 14 (3), 213-228 (2006)

[6] Vanderschuren, M.: Safety improvements through Intelligent Transport Systems: A South African case study based on microscopic simulation modelling. Accident Analysis and Prevention 40 (2), 807-817 (2008)

[7] Schumacher, H., Schack, M. and Kurner, T.: Coupling of simulators for the investigation of car-to-x communication aspects. In: Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific, pp. 58-63 (2009)

[8] van den Hoogen, E. and Smulders, S.: Control by variable speed signs: results of the Dutch experiment. In: Road Traffic Monitoring and Control, 1994., Seventh International Conference on, pp. 145-149 (1994)

[9] Smulders, S. A. and Helleman, D. E.: Variable speed control: state-of-the-art and synthesis. In: Road Transport Information and Control, 1998. 9th International Conference on (Conf. Publ. No. 454), pp. 155-159 (1998)

[10] Highway Agency: M25, Controlled Motorway, Summary report. Report, (2007)

[11] van Toorenburg, J. A. C. and de Kok, M. L.: Automatic Incident Detection in the Motorway Control System MTM. Technical report (1999)

[12] Lind, G. and Strömgren, P.: Säkerhetseffekter av trafikledning och ITS, Litteraturinventering, v0.9. Report (2011)

[13] Krajzewicz, D.: Traffic Simulation with SUMO - Simulation of Urban Mobility, In: J. Barceló (eds.) Fundamentals of Traffic Simulation. International Series in Operations Research & Management Science, vol. 145, pp. 269-293. Springer Verlag, De (2010)

[14] HEBFA, http://www.hbefa.net

[15] iTetris, http://www.ict-itetris.eu

[16] DRIVE C2X, http://www.drive-c2x.eu/project

[17] Sommer, C., Zheng, Y., German, R. and Dressler, F.: Simulating the influence of IVC on road traffic using bidirectionally coupled simulators. IEEE INFOCOM Workshops 2008, 1 (2008)

[18] Krajzewicz, D., Brockfeld, E., Mikat, J., Ringel, J., Rossel, C., Tuchscheerer, W., Wagner, P. and Wosler, R.: Simulation of modern Traffic Lights Control Systems using the open source Traffic Simulation SUMO. In: Industrial simulation conference; ISC'2005, pp. 299-304. Ghent, EUROSIS (2005)

[19] Fernandes, P. and Nunes, U.: Platooning of autonomous vehicles with intervehicle communications in SUMO traffic simulator. In: 13th International IEEE Conference on Intelligent Transportation Systems, ITSC 2010, pp. 1313-1318 (2010)

[20] Krauß, S.: Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics,Thesis, Köln University, Germany (1998)

17 Robust and Hybrid Location Management for Urban Vehicular Environments

Aisling O' Driscoll, Dirk Pesch;
NIMBUS Centre for Embedded Systems Research, Cork Institute of Technology (CIT), Cork, Ireland

17.1 Abstract

Unicast geo-routing protocols are reliant on a robust location service protocol to successfully seek the destination vehicle's location. Furthermore successful V2X communication is reliant on the geo-routing protocol to successfully deliver the packet. In both cases successful packet delivery and robustness of the protocols is paramount, the failure of either renders communication a failure. In order to maximise packet delivery, this paper proposes a framework comprised of a location service, the Urban Vehicular Location Service (UVLS), and a geo-routing scheme, the Infrastructure Enhanced Geo-Routing Protocol (IEGRP), that exploits infrastructure where available to function in completely distributed, partially connected and fully infrastructure based networks. Unlike previous location service schemes that are typically fully distributed or centralised only, UVLS is the first location service to address robustness and locality awareness over hybrid vehicular networks with its performance further optimised via location caching, opportunistic location resolution and the IEGRP routing protocol.

Keywords: location services, geo-routing, V2X, RSU, hybrid vehicular networking.

17.2 Introduction

While the primary focus of vehicular applications has traditionally been towards traffic safety services, promising traffic management, entertainment and location based applications have also been proposed. When enabling V2X unicast routing, the benefits of stateless geo-routing protocols, also referred to as position or location based routing, over their topology based counterparts, are widely recognised. However in order to make a forwarding decision to successfully deliver a packet, a geo-routing protocol must firstly determine the location of the destination vehicle, otherwise all communication fails. It is therefore reliant on a protocol known as a location service to provide timely and accurate location information towards which it can route. Without a robust and reliable location service to ensure location server availability and query resolution, geo-routing ultimately fails. Furthermore, inaccurate location information will invalidate the operation of the geo-routing algorithm leading to incorrect forwarding decisions. Even if an ideal location service is assumed, an assumption typically made in quantitative geo-routing evaluations, packet delivery is not guaranteed as a result of the frequently disconnected nature of VANETs. Therefore robustness of the geo-routing protocol, in order to maximise packet delivery, is also paramount.

It should be further noted that all too frequently either the location service schemes or the geo routing protocols that use them, only consider fully distributed networks and are not equipped to avail of road-side infrastructure, if available, to maximise delivery. The alternative scenario is that they assume the existence of full infrastructure and ignore the consequences of frequent disconnections. Few routing or location service protocols can adapt to operate in fully distributed, particularly equipped or fully infrastructure based topologies. As a fully deployed RSU infrastructure may be labour intensive to deploy and may not be economically feasible in the short term, partial infrastructure in urban environments is a more imminent reality, particularly in the early years while vehicular networks are being adopted. Not only does the utilisation of RSUs in multi-hop routing facilitate wide area vehicular communications as it can be assumed that RSUs are connected via high bandwidth links, but it can also overcome holes in the network to improve the delivery rate for short distance communications.

The ability to adapt and exploit infrastructure, where available, can facilitate robust routing and location service protocol performance. Without such robustness and if packet delivery is not maximised, all other routing and location service protocol optimisations are rendered useless. The success of V2X applications is only feasible if connectivity can be provided and packet delivery is maximised, which is dependent on either high vehicular density to sustain multi-hop communications or on the availability of fixed infrastructure in the form of RSUs.

This paper therefore presents a location management framework that exploits the presence of infrastructure where available but that can also overcome its slow introduction providing a hybrid and robust location service, UVLS and routing protocol, IEGRP, for city and suburban environments.

17.3 Related Work

Current unicast routing algorithms typically provide a completely distributed routing solution or very recently a few authors have proposed centralised only approaches. Distributed vehicular unicast routing protocols can be categorised as position-based [1]-[10], delay tolerant [11]-[15] or QoS based [16], [17]. None of these routing protocols consider interaction with partial or full infrastructure and would thus treat a RSU as a “regular” neighbour in the routing algorithm. Very recently, the benefits of utilising fixed infrastructure as a routing complement to the vehicular ad-hoc network, in order to improve reliable end to end multi-hop communications, has been recognised [18]-[22] but assumes a high proliferation of RSUs. A plethora of location service protocols have been proposed to date. Majority of these are designed for MANET environment [23]-[45] and do not cope well with the high dynamism of vehicles and restricted mobility patterns in accordance with the road topology. Distributed vehicular equivalents [46]-[58] have been proposed but also suffer from limitations, in particular with reduced density, long path lengths and the challenging radio conditions associated with urban vehicular environments. Centralised only location services have also been proposed [59], [60] but do not consider partial infrastructure deployment. Very few of these location service algorithms are quantitatively compared against vehicular equivalents, favouring MANET based counterparts or utilise routing schemes with known packet delivery performance drawbacks in vehicular networks.

17.4 Urban Vehicular Location Service (UVLS)

The protocol design of UVLS is now described with IEGRP described in Section 17.5. A location service protocol needs to address how to recruit and maintain location servers, when to handover location information, send location updates as well as how the updated location information should be disseminated and finally how to determine the location servers of a target node given its ID in order to successfully resolve its location.

Vehicles must firstly determine the frequency with which they should update their location information. UVLS uses a combination of time and distance based update triggers. A vehicle updates when its travels a distance that exceeds the distance threshold or based on an expired timer, whichever occurs first. The timer based update is triggered based on the time taken to travel the distance threshold based on the vehicles average speed. Importantly, once a vehicle determines that it must update its location information, it must firstly determine the appropriate location server. In order to do this, UVLS vehicles transmit their location information to their closest intersection. Static nodes within range of intersections are always favoured as location servers. If infrastructure does not exist at an intersection, UVLS selects a candidate vehicle to act as a mobile location server based on its proximity to the centre of the intersection. Each vehicle is able to determine this based on information exchanged in extended beacon messages. As a periodic beaconing mechanism is employed by all geo-routing protocols, there is no additional overhead introduced from location server handover, preventing loss of location information. A timer is associated with the update that allows servers to invalidate entries shortly after a new update entry is expected. This timer is calculated based on the current time, the time the update was issued and the speed the vehicle was travelling at the time of issue. To facilitate queries for vehicles that are not in the vicinity of this location server, the location server also forwards the updated location information to the location server at the intersection with the closest succeeding ID within a bounded zone, determined via a common hashing function such as SHA-1 (Secure Hashing Algorithm). The update is therefore distance-bounded. The UVLS update algorithm is shown in Figure 17-1.

```
1:  if (dist_since_last_update > distance_thresh) or
    (timer_expiry = TRUE) then
2:     send_update (veh_ID, X, Y, S, Dir) to the closest intersection;
3:     update_trigger_time = dist_thresh / veh_avg_speed;
4:     if (interim_forwarding_node is not Location Server) then
5:        cache_location_update;
6:        Set update_expiry = (dist_thresh/S) * 1.5;
7:     else
8:        cache_location_update and set update_expiry;
9:        if (closest_successor (hash (int_X, int_Y),
          hash (veh_ID)) == false) then
10:         Get closest_successor in zone and forward
11:         Return to step 4;
12:       end if
13:    end if
14: end if
```

Figure 17-1: UVLS Location Update Algorithm.

In order to resolve location queries and as vehicular application traffic is typically localised, vehicles firstly attempt to resolve a query in their surrounding area. The source vehicle unicasts the query in every direction available according to the road topology. Each vehicle to receive the query packet will check its own location database to determine whether they can resolve the destination. If an up-to-date entry exists, a reply message is generated and transmitted towards the source vehicle, where this location information is used to communicate with the destination. Alternatively if no entry exists in the location database, the query is forwarded. If one of the forwarding vehicles is in range of an intersection or an area where other exit roads exist it replicates the query packet in those new directions. Each vehicle will only process a query packet that it has heard for the first time. The query packet has a TTL which expires after five hops to prevent looping around the network. The source vehicle sets a timer on the transmission of the query with the local query deemed to have failed if the timer expires.

Unlike the C2CNet location service or any similar quorum approaches, UVLS uses unicast packets rather than local broadcasts and therefore minimises the number of nodes that need to be involved in the query or reply process, or involved in the location caching process.

If a reply has not been received and the local query timer expires, the source vehicle will deduce that the local quorum location service has failed. The source vehicle will then route the query to the location server at the closest intersection. The location server will consult its own location database and assuming that an entry does not exist will take a particular action depending on the location server type. Should the location server be a fixed RSU, it will check if infrastructure is available at the intersection with the closest succeeding ID in the bounded zone. The query packet is then unicast to this intersection. If infrastructure is not available, the query is multicast between RSUs that subsequently query their own location databases. A timer is set within which the originating RSU expects the query to be resolved. If the timer expires and the originating location server has still not received a location reply, non-infrastructure assisted intersections are queried. Firstly the originating location server queries the mobile location server at the closest succeeding ID intersection in the local query zone. If this does not resolve the query, surrounding non-infrastructure assisted intersection are searched in an expanded ring search. However if the location server is mobile, it transmits the query as a unicast to the intersection with the closest succeeding ID in the bounded zone. As above, if a reply is not received within the timer expiry time, surrounding regions are queried, moving to progressively further regions if a reply is not received. Pseudocode outlining the UVLS query algorithm is shown in Figure 17-2.

```
1: if (data packet is received from upper layer) then
2:    location = check_nbr_table (queried_veh_ID);
3:    if (location == NULL) then
4:      location = check_cache_table (queried_veh_ID);
5:      if (location == NULL) then
6:        send_local_query_unicast (queried_veh_ID) in all directions;
7:        Set local TTL and expiry_timer;
8:        if (expiry_timer < current_time) then
9:          Go to step 13;
10:       else if (reply packet received)
```

```
11:          Go to step 45;
12:        end if
13:        forward_query_to_closest_intersection     (queried_veh_ID);
14:      if (interim_forwarding_node is not Location Server) then
15:          cache_loc_update (src_veh_ID, X, Y, S, Dir);
16:         Return to step 13;
17:        else:
18:          cache_loc_update (src_veh_ID, X, Y, S, Dir);
19:          if (RSU == true) then
20:              location = check_loc_table (queried_veh_ID);
21:            if (location != NULL) then
22:              Go to step 45;
23:            else
24:              if (closest_successor_in_local_zone == RSU) then
25:                 forward_query (queried_veh_ID);
26:               else
27:                multicast_query_to_rsus (queried_veh_ID);
28:                 set expiry_timer;
29:                if (expiry_timer < current_time && no reply) then
30:                    fwd_to_closest_suc_in_local_zone;
31:                   set expiry_timer;
32:                  if (expiry_timer < current_time && no reply) then
33:                     search surrounding mobile location servers in an
                        expanded ring search;
34:                    end if
35:                  end if
36:                end if
37:              end if
38:          else
39:            Go to step 30;
40:          end if
41:        end if
42:      end if
43:    end if
44:  end if
45:  send_waiting_data_packet (location);
```

Figure 17-2: UVLS Query Algorithm

The packet delivery rate of UVLS is further improved by two simple optimisation techniques. As previously outlined, vehicles cache location information that they have passively learned from buffered and forwarded packets. To avoid the situation where stale information is cached and returned to the querying vehicle, a vehicle considers this passive location information to only be valid for a maximum time of *t/s*, the time it takes a vehicle to exceed the radio range at maximum speed. Secondly, if a vehicle receives a location query packet and determines that it is the destination being queried, it directly sends a location reply to the source vehicle and drops the location query packet. These simple caching and opportunistic location resolution optimisation techniques improve location service QSR and reduce query path lengths.

17.5 Infrastructure Enhanced Geo-Routing Protocol (IEGRP)

UVLS can operate over any geo-routing protocol and is not in any way dependant on IEGRP to fulfil its location service function. IEGRP uses stateless greedy routing with store and forward buffering and predictive link stability, characteristics that have been identified as being best practice for vehicular routing [61]. However the distinguishing factor of IEGRP is that it exhibits a number of hybrid characteristics

- If a forwarding node is a RSU, it checks the target destination of the packet and routes to the closest RSU over the wired backbone network. In the simulation analysis this is referred to as IEGRPv1.
- The greedy routing strategy will prioritise a RSU out of its one hop neighbours, if it can make greater forward progress towards the destination to another RSU over the backbone network, provided the destination is not a one hop neighbour.
- It will advertise RSU neighbours in its beacons so that indirect neighbour vehicles may learn about infrastructure based two hop neighbours. It will then allow the greedy or store and forward rule to be overridden in order to reach a RSU but only if the RSU has a wired connection to a RSU that will gain closer physical distance to the destination. In the simulation analysis the afore-mentioned two features are referred to as IEGRPv2.

Therefore the IEGRP forwarding algorithm will not simply choose the neighbour that will achieve the greatest physical progress towards the destination but rather, may choose an RSU that achieves less distance towards the destination in one hop than the existing vehicle, if it can offload to another RSU that can gain greater physical distance in the subsequent hop. Secondly in the majority of existing delay tolerant vehicular routing protocols, if a vehicle is at the local maximum and does not have a neighbour that it can route greedily towards, it will buffer the packet until it comes within range of a neighbour vehicle that will geographically advance the packet towards its destination However IEGRP exploits topology knowledge acquired indirectly via beaconing, regarding two hop RSU neighbours, to select the best target forwarder that may actually achieve negative forward progress in the short term but will be able to route to another RSU that will achieve greater physical proximity to the target vehicle overall. As this additional beacon information simply contains the location of the two hop RSU, it does not contribute to excessive additional overhead. Figure 17-3 illustrates this with the blue vehicle forwarding the packet to the black vehicle in order to offload the packet to a RSU that can route over the wired network to a RSU in closer proximity to the target red vehicle. Pseudocode for the main component of the IEGRP routing algorithm is shown in Figure 17-4.

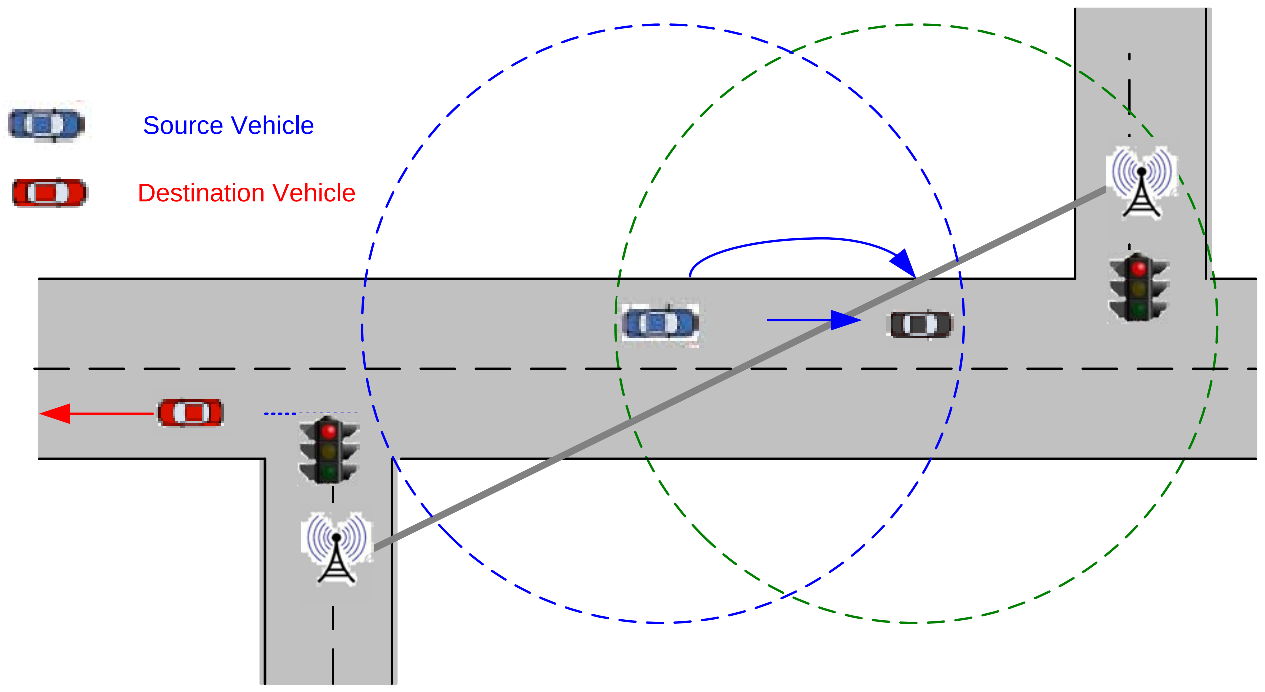


Figure 17-3: IEGRP Over-riding greedy routing algorithm to favour Road-Side Unit connectivity

```
Notations:
Input: Packet P                    Source: Vehicle S
Destination: Vehicle/RSU D         Radio Range: R
S' Neighbour Table: N(s)

 1: nextHop = S
 2: currentDistance = distance(S, D)
 3: if (currentDistance < R)
 4:   P is sent directly to D
 5: else
 6:   for all nbr in N(s)
 7:     if Ns == RSU
 8:       cRSU = closestRSU (Ns, D)
  9:      if (cRSU has wired connection to a RSU
          closer to D than currentDistance)
11:          nextHop = cRSU
12:          break to line 29
13:        end if
14:      else if (distance (nbr, D) < currentDistance)
15:     nextHop = nbr
17:   end for
18: end if
```

```
19: //No greedy Neighbour identified
20: if (nextHop == S)
21:   for all nbr in N(s)
22:     nbrCRSU = closestRSU (nbr(Ns))
23:     Repeat Line Numbers 9-13
24:   end for
25: if (nextHop == S)
26:   bufferPacket (P)
27:     break out of IEGRP
28: end If
29: sendPacket (P, nextHop)
```

Figure 17-4: IEGRP V2X Routing Algorithm

17.6 Vehicular and Network Simulation Modeling

The performance of UVLS and IEGRP is evaluated using *OPNET*, a commercial network simulator [62]. In order to achieve realistic vehicular mobility patterns, the microscopic traffic simulator, *SUMO* [63], is used with the obtained vehicular traces imported using an *OPNET* customised mobility model as the movement patterns of the wireless nodes. The mobility traces are based on a 600m2 Manhattan grid with uniform 200m street lengths. Vehicular traffic is generated according to traffic flow definitions files via the *SUMO DUAROUTER* module with densities to model sparse to busy yet free flowing conditions. Vehicles are maintained in the network for the duration of the simulation using the SUMO *rerouter* feature.

Custom process models were developed in *OPNET* for the proposed UVLS and IEGRP protocols as well as the comparative location service and routing protocols. A quantitative evaluation will be conducted of IEGRP against greedy, delay tolerant and QoS enabled routing protocols. Furthermore UVLS will be compared against schemes from each location service category: MBLS (hierarchical vehicular scheme) [51], ILS (flat DHT vehicular scheme) [54] and GHLS (flat MANET scheme) [41].

The radio propagation model deployed when simulating protocol performance can have a significant impact on successful packet reception. In this simulation study, an empirical propagation model is considered, based on measurements conducted in Cork City [64]. This considers path loss due to distance, obstacle interference via shadowing based on a log normal distribution and multipath fading via the Nakagami-m model. It yields a 100% delivery rate up to 35m after which the probability of successful delivery decreases accordingly up to a maximum radio range of 55m, where 100% packet loss is experienced.

The vehicular PHY and MAC layer are modelled according to the 802.11p standard with each vehicle transmitting a beacon to its one hop neighbours at 100ms intervals as per the WAVE specification assuming a dedicated control channel, with neighbour entry expiry after 150ms. All simulations are run for 1000s. As the precise characteristics of vehicular telematics and infotainment applications have not yet been specified, communication is modelled according to a Poisson process with inter-arrival times modelled according to an exponential distribution with a mean of five seconds.

As the availability and placement of RSU infrastructure will have an impact on the location service performance, in particular on its success rate, and in order to discount the possibility of biased evaluation as a consequence of the random placement of infrastructure, the evaluation in this paper presents the mean across a subset of permutations in the RSU placement. It is assumed RSUs are located where there is existing infrastructure such as traffic lights at intersections (maximum 12 positions in the discussed simulation analysis). As it was not computationally feasible to evaluate every possible permutation, a diverse subset was chosen to cover RSUs clustered in a particular part of the network, those distributed throughout the network and finally RSUs located around the perimeter of the network.

17.7 Simulation Analysis

The performance of UVLS, IEGRP and comparative protocols is evaluated with respect to the Packet Delivery rate (PDR). The PDR is the ratio between the data packets generated at the source vehicle to those successfully delivered to the destination vehicle. An idealised location service is used as it provides a benchmark for location service performance

This metric is the most important when highlighting the improved performance and robustness of the proposed location management framework and is evaluated over varying RSU and vehicle densities and source and destination vehicle distance ranges. Source-Destination range queries refer to a data packet destined for a target vehicle within a particular radius of the source vehicle in order to examine the impact of node proximity and in particular its impact on the QSR of the respective location services. A destination vehicle is chosen within a specified minimum and maximum range from the source. This distance is chosen at the time that packet is generated and may increase or decrease depending on the trajectories of the source and destination vehicles. Destination vehicles that are less than 55m from the source are not chosen as such queries are resolved in the source vehicle's one hop neighbour table. Therefore ranges starting at 55m-110m are considered, increasing in 55m increments.

The impact of these range queries on each location service across varying degrees of infrastructure is shown in Figure 17-5(a-b) which demonstrates the mean PDR derived across six variations in RSU placements for each given RSU density category and considering two varied source destination ranges. It can be noted that as source destination distance ranges are increased there is increased dependance on multihop routing. IEGRP makes better use of infrastructure, where available, to deliver packets. Negligible improvement can be observed for the 110m-165m distance range but IEGRP incurs much improved PDR for the increased distance range of 220m-275m with partial and full infrastructure.

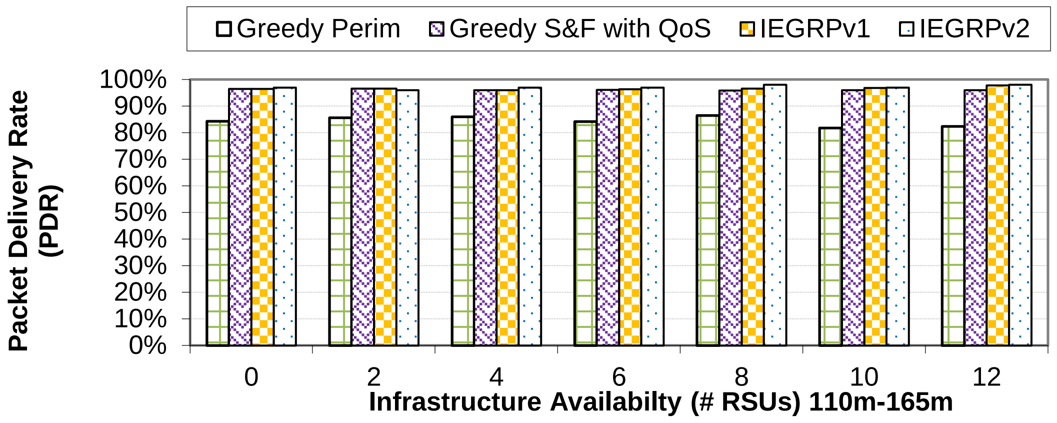


Figure 17-5(a): IEGRP V2X Routing Algorithm Performance over varied infrastructure densities with a source destination range of 110m-165m.

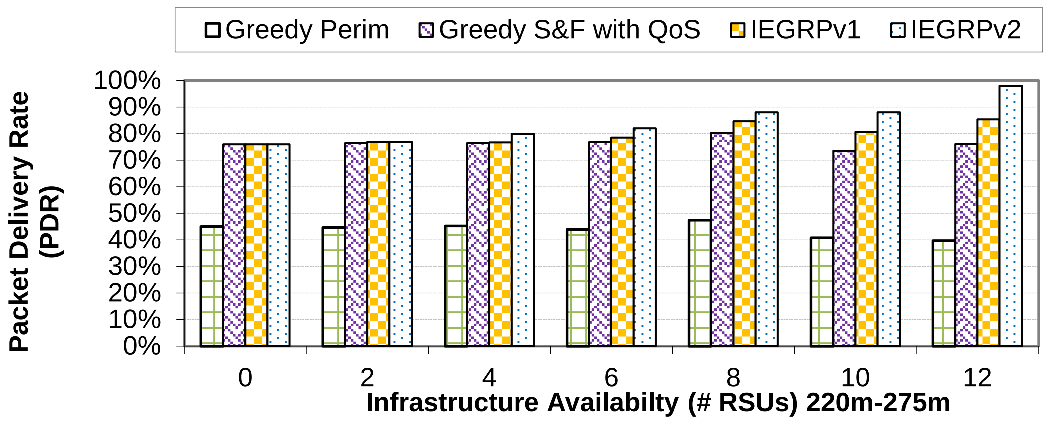


Figure 17-5(b): IEGRP V2X Routing Algorithm Performance over varied infrastructure densities with a source destination range of 220-275m.

The impact of these range queries on each location service is shown in Figure 17-6(a-b). UVLS out performs GHLS, ILS and MBLS in all scenarios. UVLS performs very closely to an idealised location service for local queries and also demonstrates excellent PDR for wide area queries with partial RSU coverage. This is dependent on VANET density. It can be noted that that as the distance between the source and destination vehicles increase, the PDR suffers degradation in performance, even with an idealised location service, as infrastructure is decreased.

It should be observed that GHLS performs poorly in all scenarios despite RSU availability or close proximity between source and destination vehicles. A GHLS scaling factor of 0.5 was used in the simulation analysis as this achieved the best performance. Factors of 0.25, 0.75 and 1.0 were also evaluated but are not shown for brevity. Poor GHLS performance is as a result of the lack of correlation between the location server placement strategy and the underlying topology causing particular hashed server coordinates to be permanently isolated. Furthermore the authors of GHLS state that a node's unique identifier is hashed into a location with the node closest to that location assuming the role of the location server. However given the frequently disconnected nature of vehicular networks, a forwarding node may believe it is the closest node to the hashed location server coordinates, though not within radio range, thereby storing a vehicle's location information and assuming the role of location server. As expected this has a negative impact when a vehicle queries the location server location with such queries only resolved if the query packet opportunistically encounters the rogue location servers. GHLS performance, though consistently poor throughout, is not affected by the range of queries.

MBLS provides similar delivery rates to UVLS when full infrastructure is available. However the performance of MBLS is heavily dependent on infrastructure availability with its performance decreasing sharply with partial infrastructure and also with decreased vehicular density. A number of factors contribute towards declined MBLS performance, even in the case of full infrastructure. Firstly, MBLS uses a static hierarchical partitioning scheme, independent of the underlying road topology, and thus does not consider empty order squares such as though on the outer edges of the network. Therefore as the MBLS update and query mechanism is based on a chain of servers located at intersections, empty order 1 squares result in the ultimate failure of update or query transactions. Furthermore, as infrastructure is decreased, non-infrastructure based lower order servers are responsible for forwarding updates and queries to higher order servers, relying solely on a connected path through the ad-hoc network. Finally, the update mechanism employed in MBLS has a negative impact on performance. MBLS proposes, in an effort to reduce the location service overhead that is typically associated with periodic updates, to update only when a vehicle passes a new intersection and when crossing a boundary between order squares. Also when a location server receives a query it replies with the predicted location of the vehicle based on its previously stored coordinates and speed as opposed to the actual cached location. The update expiry threshold is calculated as the time taken to travel between intersections in addition to the time taken to travel the radio range at the current speed. This approach decreases the accuracy of the returned location as it does not consider the distances between intersections or that a vehicle does not travel at a constant speed.

ILS, as with MBLS performs well when infrastructure is available but quickly deteriorates as infrastructure is reduced. This is as a result of its necessity to send periodic maintenance messages between intersections to maintain the DHT overlay. If these messages cannot be delivered the overlay becomes inconsistent and permanently partitioned. Furthermore DHT successor and predecessor pointers become inaccurate. This scheme is only feasible with full infrastructure and if locality awareness is considered in construction of the overlay.

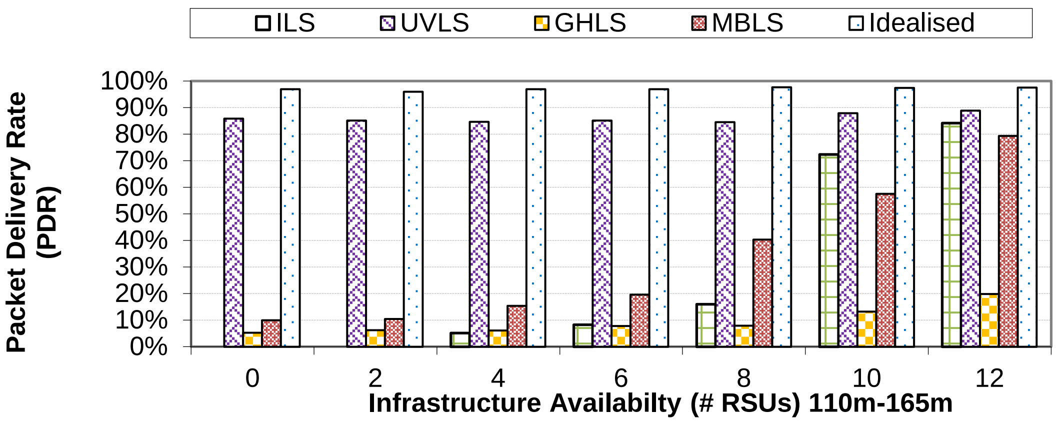


Figure 17-6(a): UVLS Comparative Location Service Performance over varied infrastructure densities with a source destination range of 110m-165m.

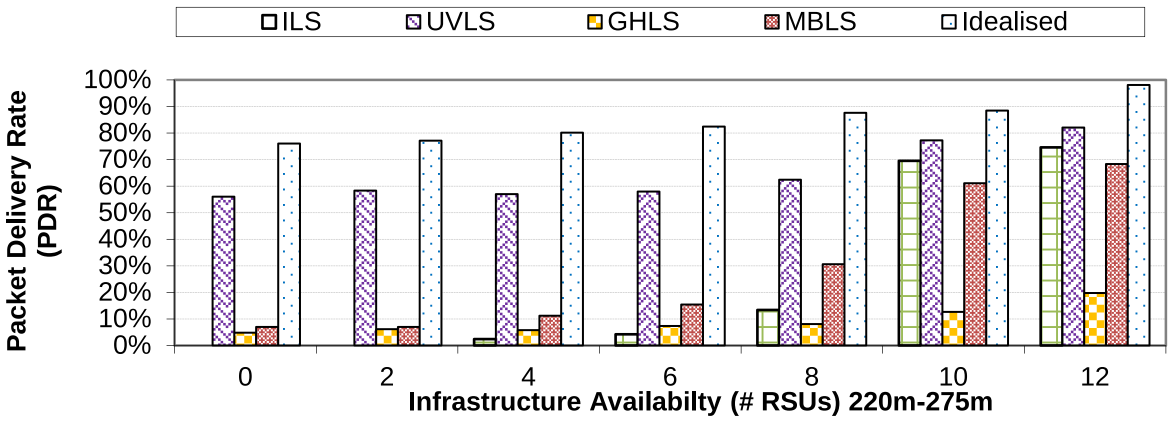


Figure 17-6(b): UVLS Comparative Location Service Performance over varied infrastructure densities with a source destination range of 220m-275m.

As expected, the impact of vehicular density plays a vital role in the delivery rate of any location service as infrastructure is decreased. The impact of this is shown in Table 17-1. Densities were chosen to represent sparse traffic with approximately 40 vehicles and a peak of 310 vehicles representing dense yet free flowing traffic conditions. Thus far, the results discussed in the preceding sections have assumed 310 vehicles in the network at all times. As expected, it can be noted that the performance of all schemes is susceptible to decreased vehicular density. With a decreased density and an increase source destination range it can be observed that even the idealised location service suffers a sharp decline in PDR as infrastructure is reduced. When the density is maintained at a peak, the decline in PDR is less severe as infrastructure is reduced. However UVLS is still less susceptible to performance degradation as a result of decreased density and infrastructure than comparative schemes.

Table 17-1: Impact of Vehicular Density on Location Service PDR(%)

12 RSUs

Range:	**55-110**		**110-165**		**165-220**		**220-275**	
Density:	90	310	90	310	90	310	90	310
GHLS	14.8	20.9	13.7	19.8	15.2	20.1	15.0	19.8
ILS	73,4	86.5	72.9	84.2	74.1	73.0	73.2	74.6
MBLS	62.6	81.3	61.7	79.4	64.0	70.1	62.9	68.2
Ideal	92.7	99.1	87.8	97.5	83.8	91.9	82.3	98.0
UVLS	77.9	95.5	76.5	88.9	75.4	82.9	75.3	82.0

6 RSUs

Range:	**55-110**		**110-165**		**165-220**		**220-275**	
Density:	90	310	90	310	90	310	90	310
GHLS	4.0	7.3	5.6	7.9	3.4	7.2	2.4	7.4
ILS	5.4	9.2	4.5	8.2	4.6	5.4	3.2	4.2
MBLS	13.7	16.4	13.4	19.6	12.3	19.3	6.4	15.4
Ideal	94.1	99.2	78.5	96.9	48.9	81.7	45.1	82.5
UVLS	74.3	94.9	70.3	85.1	42.2	58.6	0.4	57.9

0 RSUs

Range:	**55-110**		**110-165**		**165-220**		**220-275**	
Density:	90	310	90	310	90	310	90	310
GHLS	5.4	5.9	3.2	5.3	2.8	5.5	1.9	4.8
ILS	0	0	0	0	0	0	0	0
MBLS	5.0	10.5	5.1	10.0	4.8	8.6	0.8	6.9
Ideal	93.5	96.9	77.1	96.9	44.3	75.5	34.7	75.9
UVLS	77.4	93.4	69.6	85.9	34.5	56.3	32.3	55.9

17.8 Conclusion

In this paper a new location management framework for urban vehicular networks is described, comprised of an urban location service and an infrastructure enhanced routing protocol. The proposed IEGRP outperforms routing solutions that are not designed to exploit partial infrastructure availability thereby maximising packet delivery and UVLS outperforms current distributed location schemes by employing locality aware and robust query resolution, further enhanced via caching and opportunistic query resolution. While this paper focuses on vehicular location services as a vital prerequisite for geo-routing, this concept is easily transferable to content service discovery which would be useful for implementing publish/subscribe services and content sharing systems. Future work will consider the performance of UVLS and IEGRP based on a larger real map of Cork City in Southern Ireland and will consider the impact of a less pessimistic channel model that evaluates NLOS scenarios based on an obstacle model specific to the simulated map. V2I communications will also be considered with multiple categories of infotainment applications that require varied levels of QoS evaluated.

17.9 References

[1] Lochert. C, Hartenstein. H., Tian. J., Fußler. H., Hermann. D., Mauve. M., "A routing strategy for vehicular ad hoc networks in city environments," IEEE Intelligent Vehicles Symposium, vol., no., pp. 156-161, 9-11 June 2003.

[2] Lochert. C., Mauve. M, Fußler. H., Hartenstein. H., "Geographic routing in city scenarios," SIGMOBILE Computing. Communications. Review., vol. 9, no. 1, pp. 69–72, 2005.

[3] Lee. K.C., Haerri. J., Lee. U., Gerla. M., "Enhanced perimeter routing for geographic forwarding protocols in urban vehicular scenarios", Proceedings of IEEE 2007 Globecom Workshops, (pp. 1–10), Washington, USA, November 2007.

[4] Seet. B-C., Liu. G., Lee. B-S., Foh. C-H., Wong. K-J., Lee. K., "A-STAR: A Mobile Ad Hoc Routing Strategy for Metropolis Vehicular Communications." NETWORKING, 2004.

[5] Naumov. V., Gross. T-R., "Connectivity-Aware Routing (CAR) in Vehicular Ad-hoc Networks," 26th IEEE International Conference on Computer Communications (INFOCOM), vol., no., pp.1919-1927, 6-12 May, 2007.

[6] Schnaufer. S., Effelsberg. W., "Position-based unicast routing for city scenarios", Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, (WoWMoM 2008), (pp.1-8). Newport Beach, CA, USA, June 2008.

[7] Fußler. H., Hannes. H., Jorg. W., Mauve. M., Wolfgang. E., "Contention-Based Forwarding for Street Scenarios," Proceedings of the 1st International Workshop in Intelligent Transportation (WIT 2004), pages 155–160, Hamburg, Germany, March 2004.

[8] Lee. K-C., Lee. U., Gerla. M., "TO-GO: TOpology-assist Geo-Opportunistic routing in urban vehicular grids," Sixth International Conference on Wireless On-Demand Network Systems and Services (WONS), vol., no., pp.11-18, 2-4 Feb. 2009.

[9] Lee. K., Le. M, Haerri. J., Gerla. M., "Louvre: Landmark overlays for urban vehicular routing environments," Proceedings of IEEE WiVeC, 2008.

[10] Jerbi. M., Senouci. S.M., Meraihi. R., Ghamri-Doudane. Y., "An improved vehicular ad hoc routing protocol for city environments", Proceedings of the International Conference on Communications (ICC), pp.3972–3979), Glasgow, Scotland, June 2007.

[11] LeBrun. J., Chuah. C.N., Ghosal. D.,Zhang. M, "Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks", IEEE 61st Vehicular Technology Conference (VTC Spring), vol. 4, pp. 2289–2293, June 2005.

[12] Ahmed. S., Kanere. S.S., "Skvr: Scalable knowledge-based routing architecture for public transport networks", Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks (VANET), New York, NY, USA, pp. 92–93, ACM, 2006.

[13] Zhao. J., Cao. G., "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks," 25th IEEE Conference on Computer Communications (INFOCOM), vol., no., pp.1-12, April 2006.

[14] Leontiadis. I., Mascolo. C.,"GeOpps: Geographical Opportunistic Routing for Vehicular Networks," IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 2007.

[15] Cheng. P.C., Weng. J.T., Tung. L.C., Lee. K.C., Gerla. M., Harri. J., "GeoDTN+NAV: A Hybrid Geographic and DTN Routing with Navigation Assistance in Urban Vehicular Networks," Proceedings of the 1st International Symposium on Vehicular Computing Systems (ISVCS), Dublin, Ireland, July 2008.

[16] Mo. Z., Zhu. H., Makki. K., Pissinou. N., "MURU: A Multi-Hop Routing Protocol for Urban Vehicular Ad Hoc Networks", 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, July 2006.

[17] Namboodiri. V., Gao. L., "Prediction-based routing for vehicular ad hoc networks", IEEE Transactions on Vehicular Technology 56 (4), 2007.

[18] Borsetti, D., Gozalvez. J., "Infrastructure-Assisted Geo-Routing for Cooperative Vehicular Networks", IEEE Vehicular Network Conference, 2010.

[19] Frank. R., Giordano. E., Gerla. M., "TrafRoute: A Different Approach to Routing in Vehicular Networks", VECON, Niagara Falls, Canada, 2010.

[20] Yanlin. P., Abichar. Z., J. Chang., "Roadside-Aided Routing in Vehicular Networks", IEEE International Conference on Communications, 2006.

[21] Ding. Y., Xiao. L, "SADV: Static-Node-Assisted Adaptive Data Dissemination in Vehicular Networks", IEEE Transactions on Vehicular Technology, Vol. 59, No. 5, June 2010.

[22] Mershad. K., Artail. H., Gerla. M., "ROAMER: Roadside Units as message routers in VANETs", Elsevier Journal on Ad Hoc Networks (2011), doi: 10.1016/j.adhoc.2011.09.001

[23] Camp, T., Boleng, J., and Wilcox, L., "Location information services in mobile ad hoc networks", IEEE International Conference on Communications (ICC), 2001.

[24] Basagni, S., Chlamtac, I., Syrotiuk, V., and Woodward, B., "A Distance Routing Effect Algorithm for Mobility (DREAM)," ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), October 1998.

[25] Renault, E., Amar, E., Costantini, H., and Boumerdassi, S., "Semi-Flooding Location Service", IEEE International Conference on Vehicular Technology Fall (VTC), Canada, September 2010.

[26] Kasemann, M., Fubler, H., Hartenstein, H., and Mauve, M., "A Reactive Location Service for Mobile Ad-Hoc Networks", Technical Report TR-14-2002, Department of Computer Science, University of Mannheim, November 2002.

[27] Fan, X., Yang, X., Yu, W., Fu, X., "HLLS: A History Information Based Light Location Service for MANETs", IEEE International Conference on Communications (ICC), South Africa, May 2010.

[28] Stojmenovic, I., "A Scalable Quorum Based Location Update Scheme for Routing in Ad Hoc Wireless Networks", Technical Report TR-99-09, SITE, University of Ottawa, September 1999.

[29] Jiang, J-R., Ling, W-J., “SEEKER: An Adaptive and Scalable Location Service for Mobile Ad Hoc Networks”, ACM International Conference on Supercomputing (ICS), Italy, June 2006.

[30] Bae, I-H., “An Adaptive Location Service on The Basis of Diamond Quorum for MANETs”, IEEE International Conference on Natural Computation (ICNC), 2007.

[31] Abraham, I., Dolev, D., and Malkhi, D., “LLS: a locality aware location service for mobile ad hoc networks”, 2004 Joint Workshop on Foundations of Mobile Computing (DIALMPOM), 2004.

[32] G-Y. Chang., J-P. Sheu., “A Region-Based Hierarchical Location Service with Road Adapted Grids for Vehicular Networks”, 39th IEEE International Conference on Parallel Processing Workshops (ICPPW), San Diego, USA, September 2010.

[33] H. Saleet., O. Basir., R. Langar and R. Boutaba, “Region-BasedLocation-Service-Management Protocol for VANETs”, IEEE Trans. Veh. Technol., Vol. 59, No. 2, February 2010.

[34] Woo, H., Lee, M., “Mobile Group based Location Service Management for Vehicular Ad-hoc Networks”, IEEE International Conference on Communications (ICC), Japan, June 2011.

[35] Li, J., Jannotti, J., De Couto, D.S.J., Karger, D.R., and Morris, R., “A Scalable Location Service for Geographic Ad Hoc Routing” International Conference on Mobile Computing and Networking (MobiCom), USA, August 2000.

[36] Xue, Y., Li, B., Nahrstedt, K., “A Scalable Location Management Scheme in Mobile Ad-hoc Networks”, IEEE conference on local computer networks(LCN), November,2001.

[37] Kieß, W., Füßler, H., and Widmer, J., "Hierarchical Location Service for Mobile Ad-hoc Networks," ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 8, 2004.

[38] Yu, Y., Lu, G-H., Zhang, Z., “Enhancing Location Service Scalability with HIGH-GRADE”, IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, 2004.

[39] T. Cheng, C., Lemberg, H. L., Philip, S., Van den Berg, E., and Zhang, T., “SLALoM: A scalable location management scheme for large mobile ad-hoc networks”, IEEE WCNC, March 2002.

[40] Seet, B-C., Pan, Y., Hsu, W-J., and Lau, C-T., "Multi-home region location service for wireless ad hoc networks: an adaptive demand driven approach", WONS, 2005.

[41] S. M. Das, H. Pucha and Y. C. Hu, "On the Scalability of Rendezvous-Based Location Services for Geographic Wireless Ad-Hoc Routing", Elsevier Computer Networks, Vol. 51, No. 13, pp 3693-3714, September 2007.

[42] Woo, S-C., and Singh, S., "Scalable routing protocol for ad hoc networks", Wireless Networks, 2001.

[43] Wu. X., "VDPS: Virtual home region based distributed position service in mobile ad hoc networks" IEEE International Conference on Distributed Computing Systems (ICDCS), June 2005, USA.

[44] Cheng, H., Cao, J., Chen, H-H., Zhang, H., "GrLS: Group-Based Location Service in Mobile Ad Hoc Networks", IEEE Transactions on Vehicular Technologies, Vol. 57, No. 6, November 2008.

[45] Rao, R., Liang, S., You, J., "LBLS: A Locality Bounded Hashing-Based Location Service", Journal of Networks, Vol. 5, No. 1, January 2010.

[46] Geonet project: http://www.geonet-project.eu. Final Specification, Januaary 2010.

[47] Ahmed, S., Karmakar, G-C., Kamruzzaman, J., "Hierarchical Adaptive Location Service Protocol for Mobile Ad Hoc Networks", IEEE International Conference on Wireless Communications and Networking (WCNC), Hungary, April 2009.

[48] Flury, R., and Wattenhofer, R., "MLS: An efficient location service for mobile ad hoc networks", ACM International Symposium on Mobile Ad Hoc Networking and Computing, Florence, Italy, May 2006.

[49] Wang, W., Chinya, V., Ravishankar, V., "Hash-Based Virtual Hierarchies for Scalable Location Service in Mobile Ad-hoc Networks", Springer Science, January 2009.

[50] Bae, I-H., Kim, Y-J., "An Adaptive Location Service on the Basis of Fuzzy Logic for MANETs", IFSA World Congress, Theme: Theory and Applications of Fuzzy Logic and Soft Computing, Mexico, July 2007.

[51] Boussedjra, M., Mouzna, J., Bangera, P., Manohara Pai, M.M., "Map-Based Location Service for VANET", IEEE International Conference on Ultra Modern Telecommunications & Workshops (ICUMT), St. Petersburg, 2009.

[52] Brahmi, N., Boussedjra, M., Mouzna, J., Cornelio, A K.V., Manohara Pai, M.M., "An Improved Map-Based Location Service for Vehicular Ad Hoc Networks", IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiCOM), China, September 2010.

[53] Ashok, D.M., Manohara Pai, M.M., Mouzna, J., "Efficient Map-Based Location Service for VANETs", IEEE International Conference on ITS Telecommunications (ITST), Russia, August 2011.

[54] Chang. Y.C., Shih. T.L., "Intersection Location Service and Performance Comparison of Three Location Service Algorithms for Vehicular Ad Hoc Networks in City Environments", IEEE International Symposium on Wireless Pervasive Computing (ISWPC), Greece, May 2008.

[55] Xiang-yu. B., Xin-ming. Y, Jun. L., Hai. J, "VLS: A Map-Based Vehicle Location Service for City Environments", IEEE International Conference on Communications (ICC), 2009. ICC '09.

[56] Zhang, G., Chen, W., Hong, L., Mu, D., "A Novel Location Service for Urban Vehicular Ad Hoc Networks", IEEE International Symposium on Microwave, Antenna Propagation and EMC Technologies for Wireless Communications, 2009.

[57] Zaki, S., Ngadi, M.A., Abd Razak, S., "Location Service Management Protocol for Vehicular Ad Hoc Network Urban Environment", International Conference on Computer Science and Information Technology (CCSIT), Part II, LNICST 85, pp. 573–584, 2012.

[58] Hosseininezhad, S., Leung, V., "Reliability-based Server Selection for Heterogeneous VANETs", ICST Transactions on Mobile Communications and Applications, Vol. 11, Issues. 7-9, July-September 2011.

[59] Gerla, M., Lee, U., Zhou, B., Lee, Y., Marfia, G and Soldo, F, "Vehicular grid communications: the role of the internet infrastructure", ACM International Workshop on Wireless internet (WICON'06), USA, August 2006.

[60] Hosseininezhad, S., Leung, V., "Reliability-based Server Selection for Heterogeneous VANETs", ICST Transactions on Mobile Communications and Applications, Vol. 11, Issues. 7-9, July-September 2011.

[61] Cabrera, V., Ros, F.J. and Ruiz, P.M., “Simulation-based Study of Common Routing Issues in VANET Routing Protocols”, IEEE Vehicular Technology Conference (VTC) Spring, Barcelona, Spain, April 2009.

[62] OPNET network simulator. Available at: http://www.opnet.com

[63] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D., "SUMO - Simulation of Urban MObility: An Overview”, SIMUL 2011, The Third International Conference on Advances in System Simulation, 2011, 63-68

[64] G. P. Grau., D. Pusceddu., S. Rea., O. Brickley., M. Koubek., D. Pesch., “Vehicle-2-Vehicle Communication Channel Evaluation using the CVIS Platform”, 7th International Symposium on Communication Systems, Networks and Digital Signal Processing, UK, 2010.

18 Making Traffic Visualization Movies by Scripting a Graphical User Interface

Daniel Krajzewicz;
German Aerospace Center, Berlin, Germany

18.1 Abstract

This report presents a recent extension to the open source traffic simulation "SUMO", which allows to generate demonstration movies by scripting the user interface from an external application. The scripting was realised by extending an available on-line interface. For obtaining a final movie, further steps are necessary, which are described. Examples for using the extension are given and discussed.

Keywords: Road Traffic Simulation, Visualization, Application Programmer Interface.

18.2 Introduction

SUMO (Simulation of Urban Mobility, [1], [2]) is a microscopic road traffic simulation, developed at the Institute of Transportation Systems at the German Aerospace Center. SUMO's development started in 2001, with a first version released for open use in 2002. Since that time, SUMO evolved to a mature state and became a popular simulation tool, mainly used in academic context for research on vehicular communications (V2X), see also [3]. SUMO has a built-in graphical user interface (GUI), based on a 2D visualization using openGL [4] which is embedded in a FOX-Toolkit [5] window.

The open source nature of SUMO allows external parties to extend it for meeting their purpose. In 2008, staff members from the Technical University of Lübeck, Germany, supported a simulation scripting interface based on a socket connection, which allows an on-line interaction between a running simulation and an external application [6]. The interface is called "TraCI", an acronym for "Traffic Control Interface". In the recent past, this interface was reworked, extending it by further possibilities to access and manipulate objects that contribute to the simulation and splitting the access to different kinds of objects into separate modules for a cleaner implementation.

Within this report, a recent extension to the scripting interface is presented, namely the ability to script the simulator's GUI using an external script or program. The extension was motivated by the need to generate a demonstration movie for the PRE-DRIVE C2X project. A very common approach for generating movies, is to use a tool that periodically captures the contents of the computer screen – either completely or for a chosen area of the screen only. This method has two major disadvantages. At first, as the tool runs in parallel to the "recorded" application, it competes for resources, such as CPU time or access to the hard discs. This often yields in delays in executing one of the applications, visible in the resulting movies. The second is, that usually a movie contains complex movements of the camera and/or zoom changes. If a screen capture application would be used, then the user had to adapt the view to such a story plot manually and in real time. It is hardly possible to perform this in a perfect way.

As TraCI was undergoing a restructuring process at the same time, the idea to use it in combination with already available possibility to save screenshots via the SUMO GUI, was quite straightforward. “Atomic” actions for interacting with a “view”, the instance visualizing the simulation scenario, were recognized and implemented into a new TraCI module. Besides forcing sumo-gui to take snapshots of the visualized scenario and save them into image files, these “atomic” actions allow, for example, setting the camera’s zoom and position to a given value. As usually the camera is moving to a new position and the zoom is smoothly adapted to a new value, a higher-level API was built on top of these atomic actions. Further post-processing is used to embed decals and/or additional information in the generated images and combine them into a presentable movie.

The remaining part of this report is structured as following: at first, SUMO’s user interface and the on-line interaction API are presented, upon which the scripting interface was implemented. Then, the scripting interface itself is introduced, followed by a description of a higher-level interaction API which encapsulates the access to the scripting interface for an easier set-up of non-atomic operations. One possible post-processing solution is presented afterwards, followed by examples where the complete process was employed. Then, some further observations done while using this approach for generating movies are given. The report closes with a summarizing section including possible future extensions, and acknowledgements.

18.3 Graphical User Interface in SUMO

The applications belonging to the SUMO suite were originally set up as command line applications. Later, a graphical user interface was built upon the already existing simulation application for demonstration purposes. It is a multi-document window, which allows to open different views at the loaded scenario. The user may zoom into and move across the network using the mouse. The GUI supports different colouring schemes for evaluation of the current vehicle behaviour or of the state of the road network. In addition to roads, intersections, and vehicles, infrastructure objects, such as simulated sensors, bus stops, or traffic lights are rendered, as well as abstract shapes – polygons and points of interest (PoIs), which do not participate in the simulation process. Both, polygons and PoIs can be assigned to “layers” for ordering. In combination with network and shape importers included in the SUMO package, the GUI allows to present complex scenarios with a high range of detail, see Figure 18-1.

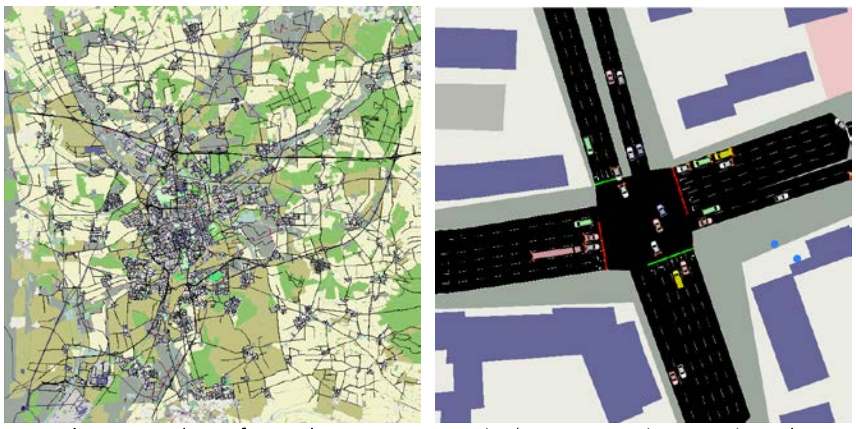


Figure 18-1: Two sample screenshots from the same scenario (top: scenario overview, bottom: zoomed at an intersection).

The GUI was implemented by deriving visualization classes from already existing simulation classes that represent artefacts the simulation model consists of, such as lanes, vehicles, bus stops, etc. Instantiation of the right class is done by using factory classes [7]. Figure 18-2 shows this approach by example for bus stops.

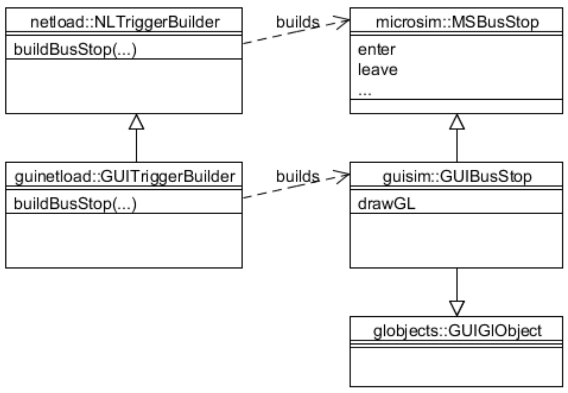


Figure 18-2: Building visualization classes derived from simulation classes using factories; this example shows the building process of simulated bus stops.

Besides implementing the drawing method, each visualization class is additionally derived from an abstract `GUIGlObject` class and implements this class' interfaces, such as:

- Retrieving the object's id and name (used to determine the object below the cursor);
- Retrieving the object's bounding box for centering the view around the object;
- Opening an interaction popup dialog;
- Opening a table which shows the object's fixed and changing parameter;
- Drawing the object's name.

SUMO was designed to handle large road networks, see [1] and [8] for examples. For increasing the rendering speed, the GUI takes advantage of an R-tree structure [9] [10] that calls the visualization classes' drawing methods for visible objects only. In parallel, the level-of-detail is increased when zooming into the road network. For example, bus stop names or vehicles are rendered only, if being at least one pixel in size. Vehicles, though visualized, are not stored in the R-tree directly, but are rendered by the roads they are currently placed at. This was done to avoid the need to update the vehicles' positions within the R-tree. The c++ R-tree implementation by Greg Douglas was used [11].

When traversed, the R-tree calls each found object's "`drawGL`" method, passing a structure named "`GUIVisualizationSettings`". This structure holds information about how objects shall be drawn, e.g., which values or methods shall be used for determining a lane's or a vehicle's colour. The visualization settings structure holds an enumeration value for the chosen colour scheme and the lane, when being drawn, calls the appropriate method to determine the value and chooses the colour representing this value from a user-defined look-up table.

Other possibilities exist to adapt the rendered objects' appearance to one's needs. Vehicles, points of interest, signs and detectors can be exaggerated in size. Their names can be displayed if wished, using custom colour and size. Additional "decal" images, usually photos taken from planes or satellites, can be loaded and displayed. After being edited in a dialog, the current visualization setting can be saved to a file and included in the simulation configuration. The file format is, as almost all files read and generated by SUMO applications, based on XML and described using a XML schema definition.

The GUI is capable to save screenshots of the currently displayed area in different bitmap and vector formats, among them .gif, .bmp, .tif, .svg, .ps, and, if compiled with the according additional libraries, .png and .jpg. Image writing capabilities are realized using FOX-Toolkit's native image support for bitmap graphics, whereas vector output is realized using the gl2ps library [12].

18.4 Online-Interaction Interface

The simulation application SUMO includes a socket based on-line interaction interface named "TraCI". When being started with the command line argument "--port-number", the simulation does not start immediately after loading the scenario, but opens a socket connection and starts to wait for a connecting client. After connection, the client is responsible for triggering simulation steps and for ending the simulation.

The client has the access to a large number of objects the simulation consists of, including vehicles, lanes, edges, intersections, traffic lights, simulated sensors, etc. Besides retrieving values from named objects, TraCI also allows to change their behaviour. For example, it is possible to change the current plan of a traffic light, change the speed of a vehicle, etc. The complete – still growing – list of accessible objects and values is described within SUMO's documentation at [2].

18.5 Basic GUI Interaction

The scripting interface for the GUI is built upon TraCI and the current message format. Each opened openGL view can be addressed in the same way as the simulated structures and allows to retrieve and change its settings. Table 1 shows which values of a view can be accessed.

Table 18-1: Parameter of a view that can be retrieved or set.

Value	Get	Set	Type
view ids	X		string list
zoom	X	X	double
offset	X	X	double*2
schema	X	X	string
boundary	X	X	double*4
track vehicle		X	string
screenshot		X	string

Here, "view ids" denotes the list of opened views, "zoom" an abstract factor by which the shown area is scaled, "offset" the offset of the view's centre from the loaded network's centre in meters, "schema" the name of the used visualization scheme, "boundary" the coordinates of the right upper and the left bottom corners of the visible part of the network. "Track vehicle" allows to centre the view on a named vehicle and the camera keeps the vehicle in the centre when running the simulation. "Screenshot" saves the current view under the given name locally on the computer the simulation is running on. The file format is defined by the given file name's extension.

Besides interacting with the view, it is also possible to add and to remove polygons and points-of-interest (PoIs) and to change their appearance. Additionally, the simulated vehicles' colours can be changed via the interface.

18.6 Higher Level Interaction

The presented on-line interface allows to interact with a single view by setting its values directly. But usually, a movie consists of complex camera movements, including fluid changes of the zoom and/or the position the camera looks at. The functionality realizing such actions was implemented in Python. A “script” consisting of entries with the fields “begin”, “end”, and “action” was processed. Sorted by “begin”, the “action” was executed as soon as the simulation reached the simulation step “begin”, and until it reached “end”. “begin” and ”end” are floating point numbers which should match the used simulation step size's granularity. Table 2 presents the implemented “actions”.

Table 18-2: Actions for interacting with a view.

Value	Parameter	Type
snapshot	<OUTPATH>	Saves a snapshot in each step between *begin* and *end* to <OUTPATH>. <OUTPATH> may contain the placeholder '%s', which is replaced by the current image number.
move_at	<VEHICLE_ID>	Interpolates "offset" between the position at *begin* and the position of the vehicle with the given id *end*
move_to	<POSITION>	Interpolates "offset" between the offset at *begin* and <POSITION> at *end*
track	<VEHICLE_ID>	Keep "offset" at the position of the vehicle with the given id. If an empty string occurs, "tracking" ends.
zoom_to	<DOUBLE>	Interpolates "zoom" between the value at *begin* and <DOUBLE> at *end*
show_ego_range		Obtains a measure for communication quality and uses it to colour vehicles around the ego vehicle.
show_per		Draws a circle around vehicles chosen as “equipped”.

Taking "snapshot" apart, the actions fall into two groups: simple interpolation between values, may them be positions or zoom levels, and complex scripts that are strongly dependent to data from the movie generating client. While the first group could be included in SUMO's API, there are no plans for realising a user-friendly solution for the second group of actions.

18.7 Post-Processing

As described, the GUI is capable to store single screenshots only. Using tools such as VirtualDub [13], available under the GPL, they can already be replayed as a movie, though often not very fluidly, as no compression is yet applied, so that both, reading the image files from the hard discs, and displaying them is more time consuming than reading a movie file. For generating movies, these images have to be merged into a single, usually compressed file. Also, it is often wished to have some additional decals, explanation text, or copyright information, see Figures 18-4 and 18-5. To realize this, we use the procedure depicted in Figure 18-3, which is described in the following.

Single screenshots are generated as described before. Afterwards, we use a second script for embedding decals and/or other additional information into the original images, one by one. This script is also responsible for cropping the images to their final size. Our script, implemented in Python, uses the "convert" application from the open source image manipulation library ImageMagick [14] for image processing. The last step, merging of the images into a compressed movie, is done using VirtualDub, which is also responsible for compressing the movie.

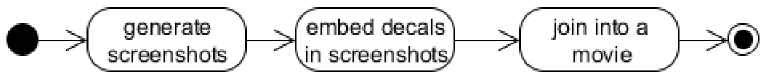


Figure 18-3: Steps for obtaining a movie.

It should be noted that the post-processing steps described here are surely not matching any state-of-the-art approaches. Sophisticated tools designed for such purposes only exist. Such tools are probably faster and more user friendly than performing the post-processing steps by scripting. The method described here is given to deliver a complete description about generating movies.

18.8 Examples

The visualization scripting API was already used to prepare presentation material within the projects iTETRIS and PRE-DRIVE C2X, both co-funded by the European Commission and both working on vehicular communication (V2X). Figure 18-4 shows two screenshots from a movie, which introduces the vehicular communication model developed in PRE-DRIVE C2X, described in [15]. The movie starts with a single "ego" vehicle (Figure 18-4a) and ends with an overview of the V2X coverage of the road network (Figure 18-4b). The penetration rate of vehicles equipped with a V2X device is 10%. The transmission ranges on the right side are visualized using polygons.

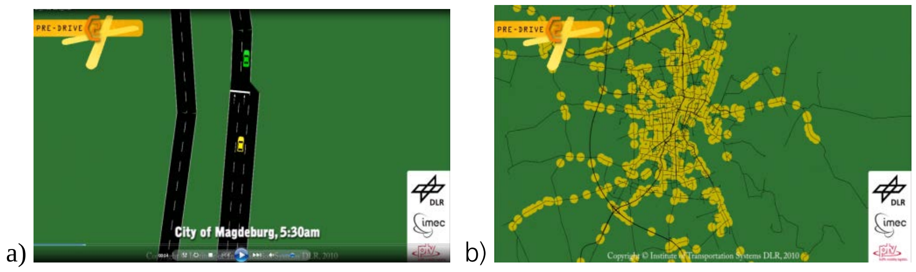

a)
PRE-DRIVE
City of Magdeburg, 5:30am
DLR
imec
b)
PRE-DRIVE
DLR
imec
Copyright © Institute of Transportation Systems DLR, 2010

Figure 18-4: Screenshots from a PRE-DRIVE C2X movie.

The "ego" vehicle was coloured in yellow, while other equipped vehicles were coloured in dependence on the probability of receiving a message from "ego". Non-equipped vehicles were coloured in grey. For visualizing the coverage, each equipped vehicle was coloured in yellow and additionally a circle-shaped polygon with the radius of communication range (here: 300m) was put below it.

Figure 18-5 shows screenshots from two movies from the iTETRIS project which demonstrate traffic management applications based on V2X. Image 5a visualizes speed collection using cooperative awareness messages (CAMs), where each dot, realized using added PoIs represents one collected value and the colour of the dot represents the speed ranging from 0m/s (red) to 13.9m/s (green).

Image 5b puts two simulations together. In both cases, an emergency vehicle is simulated, once without (left) and once with a V2X-controlled traffic light adaptation to incoming emergency vehicles (right). The emergency light was rendered using the AnoP library [16].

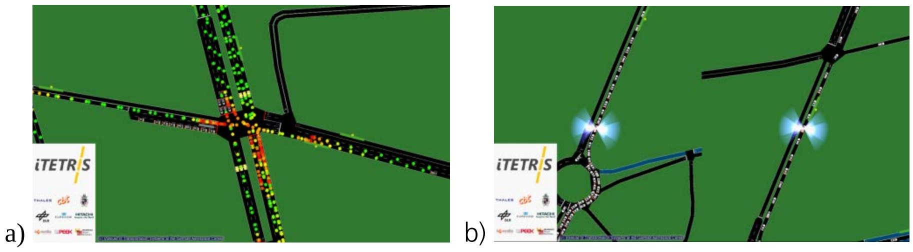


Figure 18-5: Screenshots from two iTETRIS demonstration movies.

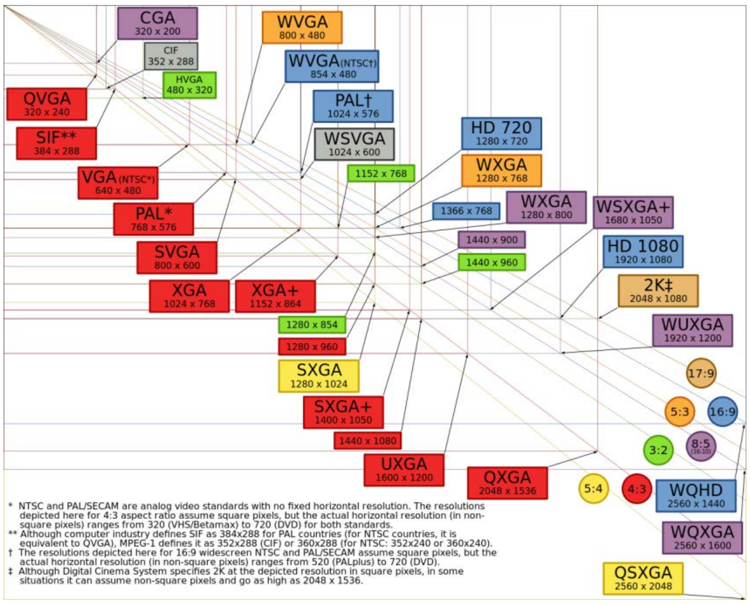
In both cases, adding text, logos and copyright information was done by post-processing single images as described earlier.

18.9 Observations and Advices

While working on the presented examples, some experiences have been made, which are shared in the following.

18.9.1 Resolution

Some standard resolutions exist [17], partially shown in Figure 18-6. The screenshots taken by the API – as all screenshots from sumo-gui – have exactly the size of the display area. As the display area is surrounded by borders, it is quite difficult to set the size of the display area to a wished resolution of the images to generate. The size of the borders must be added to the desired resolution, and, as no other method is available by now, the window size would have to be set to the resulting dimensions manually. In addition, the window cannot be larger than the screen size, limiting the resolution of the generated movie to the screen size minus window border sizes.



* NTSC and PAL/SECAM are analog video standards with no fixed horizontal resolution. The resolutions depicted here for 4:3 aspect ratio assume square pixels, but the actual horizontal resolution (in non-square pixels) ranges from 320 (VHS/Betamax) to 720 (DVD) for both standards.
** Although computer industry defines SIF as 384x288 for PAL countries (for NTSC countries, it is equivalent to QVGA), MPEG-1 defines it as 352x288 (CIF) or 360x288 (for NTSC: 352x240 or 360x240).
† The resolutions depicted here for 16:9 widescreen NTSC and PAL/SECAM assume square pixels, but the actual horizontal resolution (in non-square pixels) ranges from 520 (PALplus) to 720 (DVD).
‡ Although Digital Cinema System specifies 2K at the depicted resolution in square pixels, in some situations it can assume non-square pixels and go as high as 2048 x 1536.

Figure 18-6: Common Video Standards Resolutions [18].

One could think of possible extensions, such as opening a full-screen openGL-view or exporting images in a higher resolution by generating them via an off-screen buffer. Both methods are not available, currently. The only possibility to obtain images in a desired size was to generate larger images, and pruning them during the post-processing step.

18.9.2 Frame Rate

The frame rate [19] of a movie describes how many images ("frames" in this context) are shown per second. 24 or 25 fps (frames per second) are common. In the context of the purpose presented here, the frame rate should be put against the simulation step size, as running the simulation with a step size of 1 s and generating a movie with 24 fps yields in a very fast, hard to follow visualization of the scenario, especially if the camera is moved.

As long as the camera is focussing on a single position on the network, or tracking a single vehicle, like in both iTETRIS scenarios, using simulation step sizes of one second and a frame rate of 10 fps brought satisfying results. The PRE-DRIVE C2X movie, including different kinds of camera movements, was generated with a frame rate of 24 fps and a simulation step size of .2 seconds.

18.9.3 Technical Parts of Story Telling

The shown examples differ in story telling. Within the iTETRIS movies, the camera was fixed – either to an intersection, or to a vehicle. But the PRE-DRIVE C2X example was meant to contain a more complicated plot, as described earlier. The realisation of this story into pictures turned out to be more challenging than the implementation of the APIs, due to the following, non-functional issues.

9. In the first transition from a complete scenario view towards the "ego" vehicle, the camera was moved at a high zoom (low altitude) along the scenario. It was found that this confuses the person watching the movie, as streets pass by at a high speed without any remarkable context. Using a lower velocity yielded in a too long "prelude", so that several – time consuming – attempts had to be done to get an acceptable combination of zoom and movement.
10. If the complete communication range, e.g. 300 m, of a vehicle shall cover the complete visualized area (1280 x 720 pixels) then a vehicle is about 2 x 4 pixels in size. This is too small to recognize the colour of other passing, V2X-equipped vehicles, which shall be coloured by the probability to receive a message from "ego". In contrary, when zooming more towards the "ego" vehicle, the development of the connectivity is incomplete. In addition, the other vehicles pass too fast if running into the opposite direction than the "ego" vehicle.
11. It should be noted, that a demonstration movie is usually presented to a bigger audience, and it is hardly predictable which parts of the shown scene get into a single viewer's focus, especially if the movie is shown more often than once. As a result, the maker should be aware that any erroneous or strange behaviour will be visible.

To summarize, further traditional movie making skills – those of a director or a director of photography – are supposed to be of help.

18.9.4 Overall Performance

In comparison to the number of iterations needed to adjust the movement of the camera, the duration of generating the images and post-processing them into a movie was not found to be critical.

The major bottleneck was posed by the size and number of generated images. In all presented cases, it was decided to use .bmp images as output format, as they are lossless and offer a colour resolution of 24 bpp (bits per pixel). Such an image is about 3 MB in size. For a movie of one minute length with a frame rate of 24 fps, a hard disc storage of more than 4 GB is needed.

18.10 Summary and Outlook

The visualization scripting interface has already proved to be a usable tool for generating presentation material and we assume some of the methods described herein to be valuable for the developers of other simulation packages as well. It is worth to mention that all the steps yielding in a movie could be performed using open source applications only.

Currently, the simulation supports the possibility to connect only to a single client. As the simulation is often used in conjunction with other clients than the described movie-making script, it is currently necessary to embed the movie-making script into the client which realizes the desired simulation behaviour. This need can hopefully be solved in the future by allowing more than one client to connect to the application.

In the future, some work is assumed to be put into a 3D scene representation. First steps towards a 3D view were already done, incorporating the OSG library embedded directly into the same window. Figure 18-7 shows a screenshot of the currently implemented 3D view. Releasing the 3D view as open source is under discussion.

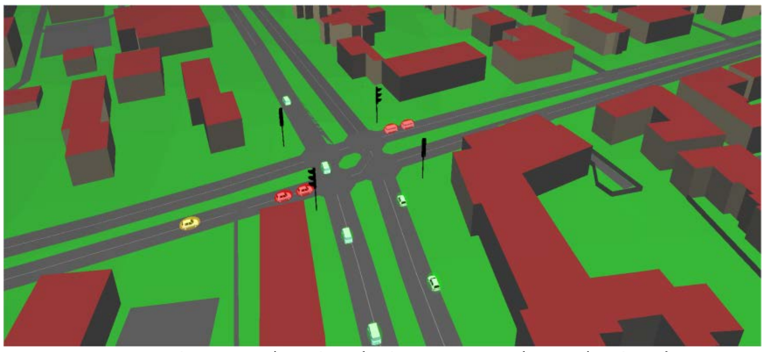


Figure 18-7: A 3D view on the simulation, currently under implementation.

18.11 Acknowledgements

We want to thank the European Commission which co-founded two projects where the developed extensions were needed, namely PRE-DRIVE C2X and iTETRIS. We also want to thank the authors and contributors of the open source and/or free projects which were used to obtain the complete movies.

18.12 References

[1] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent Development and Applications of SUMO - Simulation of Urban MObility. In: International Journal on Advances in Systems and Measurements, 5 (3&4), pp.128-138. ISSN 1942-261x (2012)

[2] DLR and contributors: SUMO homepage. http://sumo.sourceforge.net/ (2013)

[3] Krajzewicz, D.: Summary on Publications citing SUMO, 2002-2012. In: Proceedings of the SUMO2013 Conference (2013)

[4] Khronos Consortium: openGL homepage. http://www.opengl.org/, last visited on 08.04.2013

[5] Van der Zijp, J.: FOX-toolkit homepage. http://www.fox-toolkit.org/, last visited on 08.04.2013

[6] Wegener, A., Piórkowski, M., Raya, M., Hellbrück, H., Fischer, S., Hubaux, J.-P.: TraCI: An Interface for Coupling Road Traffic and Network Simulators. In: 11th Communications and Networking Simulation Symposium (2008)

[7] Gamma, E.: Design Patterns. Addison-Wesley Publishing Company, ISBN 0-201-63361-2 (1995)

[8] Krajzewicz, D., Bonert, M., Wagner, P.: The Open Source Traffic Simulation Package SUMO. In: RoboCup 2006 (2006)

[9] Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84. pp. 47. doi:10.1145/602259.602266. ISBN 0897911288 (1984)

[10] Wikipedia: R-Tree. http://en.wikipedia.org/wiki/R-tree, last visited on 08.04.2013

[11] Douglas, G.: different projects. http://www.superliminal.com/sources/sources.htm, last visited on 08.04.2013

[12] Geuzaine, C.: gl2ps homepage. http://geuz.org/gl2ps/, last visited on 08.04.2013

[13] unknown author: VirtualDub homepage. http://www.virtualdub.org/, last visited on 08.04.2013

[14] ImageMagick Studio LLC. ImageMagick homepage. http://www.imagemagick.org/, last visited on 08.04.2013

[15] Bieker, L., Krajzewicz, D., Röckl, M., Capelle, H.: Derivation of a fast, approximating 802.11p simulation model. In: Intelligent Transport Systems Telecommunications (ITST2010), Japan (2010)

[16] Krajzewicz, D.: AnoPlib homepage. http://www.krajzewicz.de/small_vision/anop.php, last visited on 08.04.2013

[17] Wikipedia: "Display resolution". http://en.wikipedia.org/wiki/Display_resolution, last visited on 08.04.2013

[18] “Jedi787plus”: Wikimedia Commons, File:Vector Video Standards4.svg. http://commons.wikimedia.org/wiki/File:Vector_Video_Standards4.svg, last visited on 08.04.2013 (2009)

19 Basic driving dynamics of cyclists

Erik Andresen, Armin Seyfried and Felix Huber;
University of Wuppertal, Faculty D – Dep. Civil Engineering, Wuppertal, Germany
Mohcine Chraibi and Armin Seyfried;
Central Institute for Applied Mathematics, Research Centre Juelich, Juelich, Germany

19.1 Abstract

In this work we introduce the Necessary-Deceleration-Model (NDM) which is a car-following-model developed to investigate driving behavior of bicycles. For this purpose the derivation of the mathematical description of the NDM is investigated. For the sake of calibration and validation of the model, several experiments are performed. The results of the experiments are presented and examined. Finally, the limits and possibilities of the Necessary-Deceleration-Model are discussed.

Keywords: Driving behaviour, driving dynamics, bicycles, car-following-model

19.2 Introduction

Recently people's awareness of the environment changed due to the global warming and lack of resources. Many people use their bicycle not only as a leisure activity but also in order to reach their work site [8]. Therefore it is not surprising that the popularity of cycling increased in the last years and is still increasing. More and more tracks for cyclists are built or still in the planning. Furthermore, the use of e-bikes and pedelecs increased, such that special routes for fast bicycles are in consideration.

In some situations, where special routes for bicycles are missing, cyclists must use together with cars the same route. However, neither traffic systems in which only cyclists are involved nor heterogeneous traffic systems with cars, motorcycles, bicycles, e-bikes and pedestrians are well investigated [7], [11], [12].

A long-range goal is to find and determine characteristics which are necessary for planning and designing roadway facilities and for controlling and regulating traffic flow [10].

With help of the Necessary-Deceleration-Model scrutinized in the following sections we investigate driving dynamics of cyclists and driving dynamics of traffic systems in which more than one type of vehicle are involved.

19.3 The Necessary-Deceleration-Model

19.3.1 Miscellaneous

The Necessary-Deceleration-Model (NDM) is a uni-dimensional car-following-model continuous in space especially designed to simulate the driving behavior of bicycles.

Like car-following-models in general [1], [2], the NDM describes the driving behavior of a vehicle from its own perspective while it is in a traffic system. That means the state variables of each vehicle are updated at each time step and determined by mathematical regularities. (see Figure 19-1)

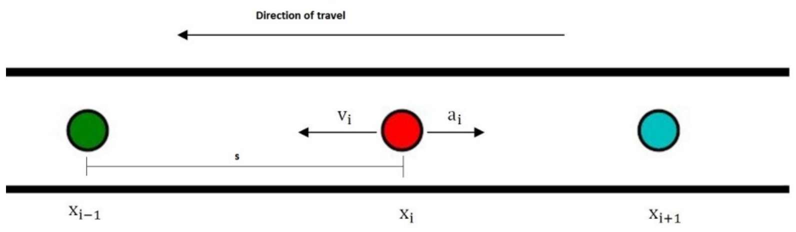


Figure 19-1: Spatial coordinate, velocity and acceleration of a virtual vehicle in a uni-dimensional car-following-model while it is in a breaking progress.

A driver, regardless of the type of vehicle he uses, has a desired speed which he tries to reach if there is no slower vehicle or obstacle in front of him.

In case there is a slower moving predecessor, the driver will decrease his speed by decelerating until his speed is aligned to the speed of the predecessor. The amount of the deceleration depends on the current headway s [1], [2], [3] and the current difference of the velocities Δv $(= v_{i-1} - v_i)$ between the considered driver and his predecessor [1], [3], [4], [9]. Considering s and Δv in every time step the deceleration has to be high enough to avoid collisions [1], [4], [9].

As all vehicles can only decrease their speed with a certain maximum deceleration, the driver has to maintain a safety distance $d(v)$ depending on his own speed [1], [3].

19.3.2 Derivation of the NDM

The NDM consists of three components acc, dec_1 and dec_2 representing the acceleration and the deceleration of a vehicle. As the NDM is discrete in time, the velocity and the spatial coordinate of a driver in the next time step are numerically calculated by means of an adequate numerical solver and considering the superposition of the acceleration and deceleration terms as expressed in the following equations:

$$x(t + \Delta t) = x(t) + v(t) \cdot \Delta t, \qquad (\mathbf{1})$$

$$v(t + \Delta t) = v(t) + (acc + \min(dec_1 + dec_2, b_{max})) \cdot \Delta t. \qquad (\mathbf{2})$$

Acceleration

The first term *acc* is representing the free acceleration of a vehicle until it reaches its desired speed v_0 . For this purpose the following expression [5] is used:

$$acc = \frac{v_0 - v}{\tau}. \qquad (\mathbf{3})$$

The relaxation time τ regulates how fast a vehicle can accelerate to its desired speed.

If a driver falls below the safety distance $d(v)$ it is not necessary and furthermore not useful for him to continue accelerating. That means the term acc is only supposed to be effective if the vehicle's current distance s is bigger than the safety distance $d(v)$:

$$acc = \begin{cases} 0, & s \leq d(v) \\ \dfrac{v_0 - v}{\tau}, & s > d(v) \end{cases} \qquad (\mathbf{4})$$

For simplicity the safety distance $d(v)$ is assumed to be linearly velocity-dependent:

$$d(v) = s_0 + l + T \cdot v, \tag{5}$$

with:

s_0 = distance between two standing vehicles (see Figure 19-2),

T = constant of proportionality,

l = length of the considered vehicle.

Deceleration

As mentioned earlier a driver has to decrease his speed with a deceleration that is high enough to avoid a collision with an obstacle in front of him. For this purpose we introduce the fundamental physical equation

$$s_{nec} = \frac{(\Delta v)^2}{2b}. \tag{6}$$

In this case s_{nec} is representing the necessary braking distance of a vehicle to avoid a collision. It depends on the square of the relative speed if the vehicle decreases its speed with a certain deceleration b. If we rearrange Equation (**6**) to

$$b_{nec} = \frac{(\Delta v)^2}{2s} \tag{7}$$

we obtain the necessary deceleration b_{nec} to avoid a collision depending on the current difference of the speeds of the considered vehicle and his predecessor and the current headway s.

However, s describes the distance between the centers of two vehicles. As vehicles of all type have a physical length, we need to increase the braking distance, so that the foremost point of a vehicle does not touch the tail of the front vehicle. Furthermore it is common to preserve some security distance between the vehicles even if all of them had come to a standstill. Taking these considerations into account we obtain from Equation (**7**)

$$b_{nec} = \frac{(\Delta v)^2}{2\left(s - 2 \cdot \frac{l}{2} - s_0\right)} \tag{8}$$

as the necessary deceleration for a vehicle to avoid a collision and to keep a certain distance to the front vehicle after the braking process (see Figure 19-2). Hereby we assume that all vehicles have the same length l.

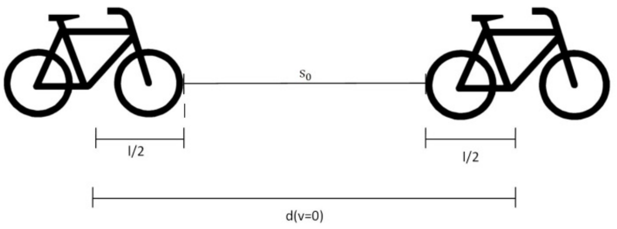


Figure 19-2: Aspired distance d(v) between two standing vehicles.

As the necessary deceleration b_{nec} is expected to be high enough to avoid headways below $s_0 + l$, there is for now no need to consider the case $s - s_0 - l \leq 0$.

Since no type of vehicle is able to slow down with a deceleration higher than a certain maximum, we have to consider the limitation of the necessary deceleration b_{nec} to a maximum b_{max} . We obtain:

$$b_{nec} = \min\left(\frac{(\Delta v)^2}{2(s-l-s_0)}, b_{max}\right). \qquad (\mathbf{9})$$

Because of the limitation of b_{nec} to the maximum b_{max} the minimal distance $s_0 + l$ can be undercut in dangerous braking situations. However these undercuts are located in the range of less than ten centimeters. Hence, they can be tolerated.

To distinguish b_{nec} from further, later introduced, deceleration terms we define:

$$dec_1 := b_{nec} \qquad (\mathbf{10})$$

The NDM is already up to now able to simulate plausible driving behaviour of vehicles in the three fundamental traffic situations (free accelerating, moving in a group, approaching an obstacle) of the longitudinal dynamic [1], except for the following situation:

As a driver approaches his predecessor, he decreases his speed until it is aligned to the front vehicle's speed. Assuming the considered driver has to decelerate with a high deceleration, so that he undercuts the safety distance, there is no deceleration which let him fall back to maintain the safety distance again. Because dec_1 = 0 if the velocities of two considered vehicles are aligned, an additional deceleration term must be considered, namely:

$$dec_2 = \frac{b_{max}}{(l-d(v))^2} \cdot \big(s - d(v)\big)^2. \qquad (\mathbf{11})$$

The second deceleration term dec_2 is only effective if $s \leq d(v)$.

The fact that dec_2 vanishes if $s = d(v)$ guarantees a continuous deceleration in every situation.

Since the distance between two drivers quickly increases if the front one is vastly faster, it is only necessary for the second deceleration term to be effective if the difference between the velocities is lower than a constant parameter ϵ. We obtain:

$$dec_2 = \begin{cases} \dfrac{b_{max}}{(l-d(v))^2} \cdot \big(s-d(v)\big)^2, & s \leq d(v);\ \Delta v \leq \epsilon \\ 0, & \text{otherwise} \end{cases}. \qquad (\mathbf{12})$$

As dec_1 and dec_2 can be effective simultaneously in certain situations we have to limit their summation to the maximum possible deceleration b_{max} due to the previously mentioned fact, that a vehicle can only decrease its speed with a deceleration not more than a certain maximum.

19.4 Calibration and Validation of the NDM

19.4.1 Miscellaneous

For Calibration and Validation of the NDM we evaluate the bicycle experiments dated on May, 6th 2012 in Wuppertal, Germany performed by the University of Wuppertal in cooperation with Juelich Research Center. About 30 participants of all ages were involved. Two of the participants used an e-bike.

Using the videos filmed by two cameras that overlooked the whole area of the car park (see Figure 19-3) we were able to extract the trajectories of the cyclists. As part of the test runs two different types of experiments were carried out.

The first experimental run called *Single Experiments* was performed to investigate the individual behaviour of cyclists while accelerating. For this purpose participants have to increase their speed from zero until the desired speed is reached.

We carried out the second experimental run called *Group Experiments* to investigate the collective driving behaviour of cyclists while moving in a group. There for a settled number of cyclists are supposed to drive through an oval track simultaneously without being allowed to overtake (see Figure 19-3). The participants were told to drive normally without haste.

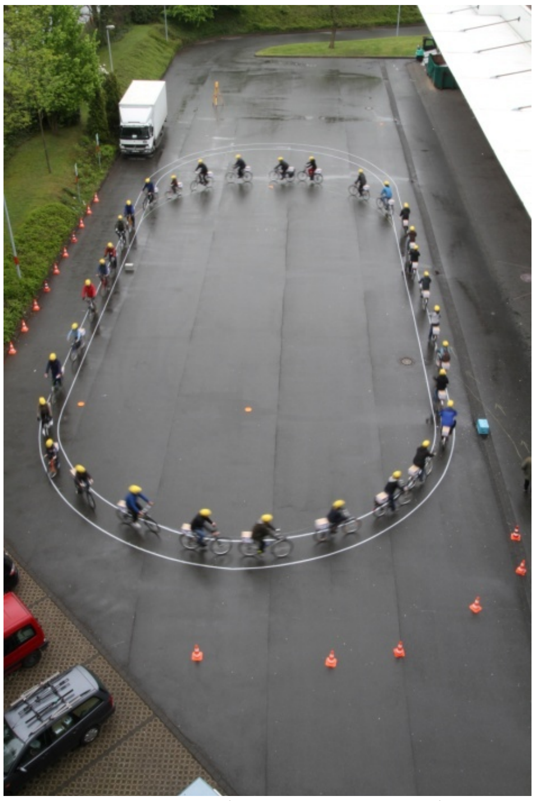


Figure 19-3: Experimental set-up (Group Experiment) with 33 participants.

19.4.2 Driving behaviour while accelerating

The average desired speed of the participants is about 4,3 m/s (corresponds 15,5 km/h) (see Figure 19-4). On average it takes 20-25 meters to accelerate to the desired speed and the duration of the acceleration phase is about 7 seconds.

As shown in Figure 19-4 the exponential acceleration that emerges from Eq. (3) is in good agreement with the empirical acceleration process of the participants. Therefore, the model parameters v_0 and τ are set to 4,3 m/s and 1,4 s respectively.

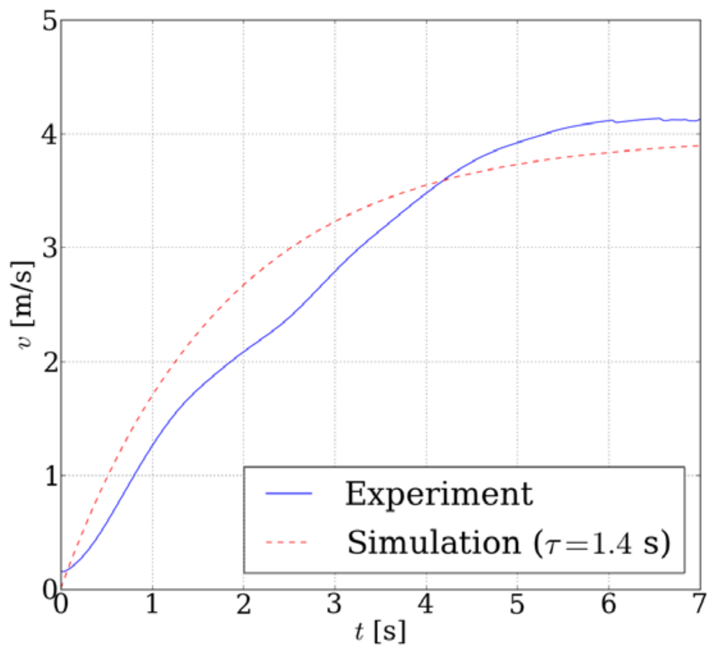


Figure 19-4: Free flow acceleration of real cyclists and cyclists simulated using the NDM.

19.4.3 Driving behaviour while moving in a group

Analysis of the gathered experimental data leads to calibrated values of NDM-parameter. See Table 19-1.

Table 19-1: Adapted model parameters of the NDM calibrated by using the measurements of the bicycle experiments.

Model parameter	Value
v_0	Gauss-distributed (mean-value: 15,5 km/h; standard deviation: 2 km/h)
τ	1,4 s
T	0,72 s
s_0	0,2 m
l	1,73 m
b_{max}	-5 m/s
ϵ	1,8 km/h

By analyzing the results of several experimental runs of the *Group Experiment* with various numbers of participants the fundamental diagrams can be set up. For this purpose experimental runs with 5, 7, 10, 15, 18, 20 and 33 participants have been performed. The length of the circuit was set to 86 m.

The following characteristics of driving behaviour can be extracted from Figures 19-5 and 19-6.

The maximum flow of ca. 0.8 s^{-1} is located at the density of ~0.25 m^{-1}. By density regions below this density, the mean speed of drivers is mainly determined by the speed of the slowest one, since he is the only one who can freely accelerate, respectively, moving with his desired speed. Above the density 0.25 m^{-1} the slowest driver is hindered by his predecessor as well, that is, the speed of the system is no more determined by the desired speed of the slowest vehicle.

Considering the calibrated model parameters of the NDM (Table 19-1) the relation of velocity, density and flow can be realistically reproduced by the NDM (see Figures 19-5 and 19-6).

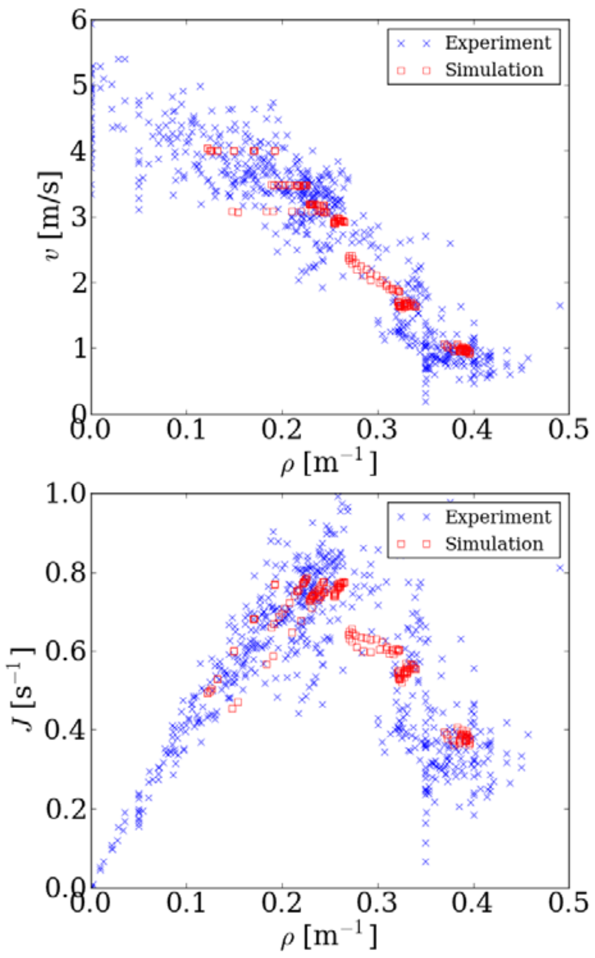


Figure 19-5: Comparison of the fundamental diagrams of the experimental runs and the simulation runs performed by the NDM using the model parameter listed in Table 19-1 and the numerical integration interval Δt=0.1 s.

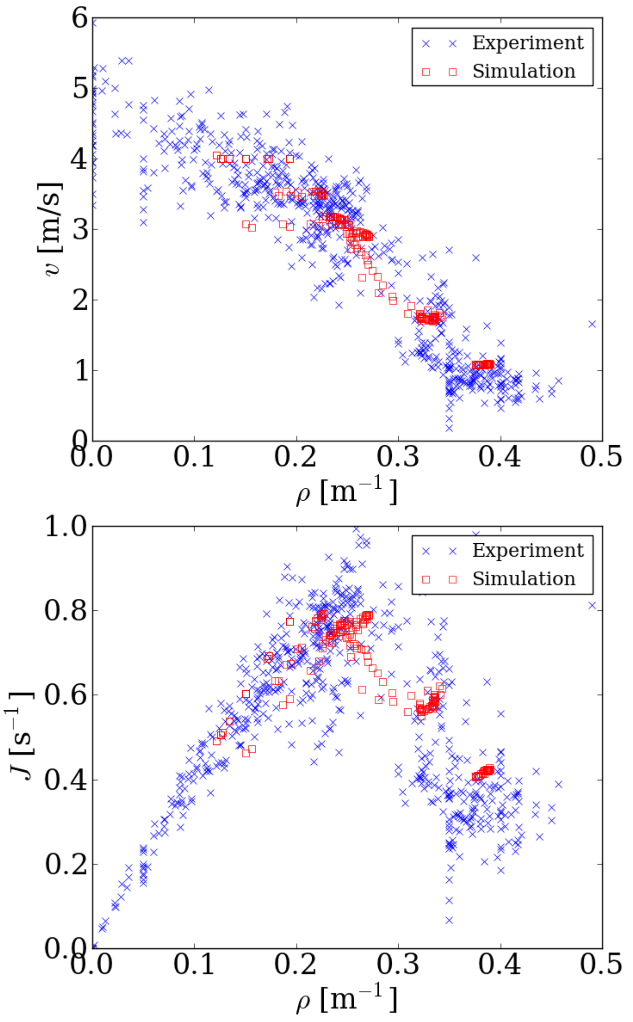


Figure 19-6: Comparison of the fundamental diagrams of the experimental runs and the simulation runs performed by the NDM using the model parameter listed in Table 19-1 and the numerical integration interval Δt=0.01 s.

The model is implemented in the programming language Python. It is possible to set the numerical integration interval Δt up to 0.1 s. without any appearance of relevant changes of the driving behaviour compared to a simulation performed using $\Delta t = 0.01$ (compare Figures 19-5 and 19-6).

For $\Delta t =$ 0.1 s, 10 minutes of driving behavior in a traffic system with 500 vehicles can be calculated in 30 seconds.

19.5 Conclusion, Limits and possibilities of the NDM

Since the NDM is developed by using fundamental physical relationships, the model parameters reflect physical characteristics of the driver or the vehicle. The NDM can model plausible driving behaviour in the three traffic situations of the longitudinal dynamic (free accelerating, moving in a group, approaching an obstacle).

By using the calibrated parameters, collisions do not occur. The driving dynamics responds only slightly to changes in the model parameters. Figures 19-4, 19-5 and 19-6 show that the model provides reasonable results when plausible values for the model parameters are used. Especially the comparison of the fundamental diagrams (see Figures 19-5 and 19-6) shows that the NDM can model realistic driving behaviour of cyclists moving in a group. Furthermore, as seen in Figure 19-4 the NDM is able to replicate the free acceleration progress of cyclists.

Using 0.1 s as the numerical integration interval, realistic driving behaviour in traffic systems can be efficiently simulated by the NDM, so that the model can be used in traffic-simulations.

Although originally the NDM has been developed to model the dynamics of cyclists in a traffic system, it is technically possible to simulate the dynamics of cars or shared-used routes as well. However, a thoroughly investigation of the NDM to investigate the exact dynamics of the mentioned systems is scheduled for future works.

19.6 Acknowledgements

We would like to thank Wolfgang Mehner and Maik Boltes working at Juelich Research Center for filming and extracting the trajectories of the bicycle experiments dated on 6th May 2012. Furthermore we would like to thank all the participants and helpers who were involved at the bicycle experiments and remain disciplined, so that excellent results of the experimental runs could be achieved although it was a cold and rainy day.

19.7 References

[1] M. Treiber, A.Kesting (2010) Verkehrsdynamik und -simulation, Springer Lehrbuch.

[2] Bando, M., Hasebe, K., Nakayama, A.,Shibata, A.,Sugiyama, Y. (1995) Dynamical model of traffic congestion and numerical simulation, Phys. Rev. E 51, 1035-1042.

[3] Gipps, P.G. (1981) A behavourial car-following model for computer simulation, Transp. Res. B Methodol. 15, 105-111.

[4] Treiber, M., Hennecke, A., Helbing, D. (2000) Congested traffic states in empirical observations and microscopic simulations., Phys. Rev. E 62, 1805-1824

[5] Pipes, L.A. An Operational Analysis of Traffic Dynamics, J. Appl. Phys. 24, 274 (1953)

[6] J. Zhang, W. Klingsch, A. Schadschneider, A. Seyfried (2011) Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions, Journal of Statistical Mechanics: Theory and Experiment, P06004 (2011)

[7] Ardeshir Faghri, Erika Egyhaziova (1999) Development of a Computer Simulation Model of Mixed Motor Vehicle and Bicycle Traffic on an Urban Road Network, Transportation Research Record 1674

[8] Gregory Gould and Alex Karner (2009) Modeling Bicycle Facility Operation, Transportation Research Board of the National Academies, Washington, D.C., 2009, pp. 157–164.

[9] Rui Jiang, Qingsong Wu and Zuojin Zhu (2001) Full velocity difference model for a car-following theory, Physical Review E, Volume 64, 017101

[10] Francis P.D. Navin (1994) Bicycle Traffic Flow Characteristics: Experimental Results and Comparisons, ITE Journal

[11] V. Thamizh Arasan and Reebu Zachariah Koshy (2005) Methodology for Modeling Highly Heterogeneous Traffic Flow, Journal of Transportation Engineering (July 2005)

[12] Chu Cong Minh, Kazushi Sano and Shoji Matsumoto (2005) The Speed, Flow and Headway Analyses of Motorcycle Traffic, Journal of the Eastern Asia Society for Transportation Studies, Vol. 6, pp. 1496 - 1508, 2005

20 Extending SUMO to support tailored driving styles

Joel Gonçalves, Rosaldo J. F. Rossetti
Artificial Intelligence and Computer Science Laboratory (LIACC), Department of Informatics Engineering (DEI), Faculty of Engineering, University of Porto (FEUP), Porto, Portugal

20.1 Abstract

Driving behaviour plays a fundamental role in transportation systems, where each single driver has a unique behaviour. In this paper we propose a methodology for eliciting driving behaviour from actual drivers, extract patterns, and generate populations of drivers based on the extracted driving styles. We show some preliminary results from driving behaviour speed. With this work we intent to provide the means for SUMO users to easily generate not only vehicle but also driving style custom populations.

Keywords: SUMO, Traffic Simulators, Driving Behaviour, Behaviour Elicitation.

20.2 Introduction

Nowadays, Transportation System research is concerned with several important issues related to rapid development of traffic network, especially in urban areas. This phenomenon has introduced dramatic changes in citizens' mobility and quality of life. Furthermore, it has proved to be a difficult challenge to cope with by researchers, decision makers, and practitioners [1]. While an adequate transportation system enables a good experience for the users, the contrary may be the source of important economic, social, and environmental issues.

By its nature, a transport system can easily become far too complex to be modelled with traditional mathematical approaches [2]. The elements composing the system (e.g. pedestrians, vehicles, road network layout, signalling layout, control systems, and so forth), the various interactions between them, and the solution space for solving a specific problem can be overwhelming. Under such conditions, simulation emerges as a natural approach for handling this complexity. Using simulation, one can model the desired transport system, explore applicable actions to the system, and predict the overcoming results of each action [3, 4]. This approach gives the advantage of covering a vast space of solutions in short time and without disrupting the real system. However, we cannot ignore the significant challenges for modelling the system in a simulation project, especially when the problem to be solved may be influenced by multiple entities with their specific interactions and dependencies [2].

The use of Multi-Agent Systems (MASs) as a paradigm for modelling the Transportation Systems rapidly emerged [5]–[7]. Specifically, microscopic traffic simulators have the ability to represent each individual vehicle in the transportation system. Each of these vehicles represents a driver, with a pre-defined starting point and destination point. Depending on the network, the drivers may choose their own path according to some decision-making process; they also may change lanes to take over other slower vehicles. Albeit these simulators give a coherent solution for analysing some problems, most of the criticism to this approach is focused on the validation of the tools. In this paper, we intent to contribute with a methodology for evolving the rigid and predictable vehicle behaviour within traffic simulators with behaviours that mimic real driver intentions underlying their decision making.

Our main hypothesis is that if in our simulations we create a virtual population of drivers where each of them resembles a set of extracted driving behaviour patterns, then our simulations will inherit driving behaviour validation and our predictions will be more accurate than traditional driving behaviour approaches when the number of drivers is not very high. In this situation, normal distributions used to play the randomness of driving behaviour are not appropriate. Furthermore, with the driving behaviour extension it would open new possibilities of application for these simulators, e.g. compose a distributed simulation where traffic simulators would manage the non-player drivers in a driving simulation.

In Section 20.3 we present the methodology, and then in Section 20.4 we propose the architecture for implementing a SUMO-based solution. After that, we show our preliminary elicited behaviour results in Section 20.5. Finally, we discuss our contributions and draw conclusions from this work in the last section.

20.3 Methodology

Our proposed methodology is composed by four core features: (i) driving behaviour modelling, (ii) behaviour elicitation and extraction, (iii) virtual population generation, and (iv) validation and calibration. An overview of this methodology is presented in Figure 20-1.

The blue elements correspond to the driving behaviour modelling features. The main process behind this phase is to maximize the usage of current driving models already used (e.g. car-following models), and fine tune their parameters in order to resemble the desired driving style. An important step is the mapping between metrics to model parameters since driving performance measurements may not be easily mapped. In those cases the model may be discarded or suffer significant changes. After identifying the proper driving performance metrics we can then design experiments to force users provide the performance metrics in the relevant driving contexts. In this phase we believe that a validation of the overall experiment design, along with the identified metrics, should be reviewed by experts in driving behaviour modelling in order to ensure the mapped metrics were validated.

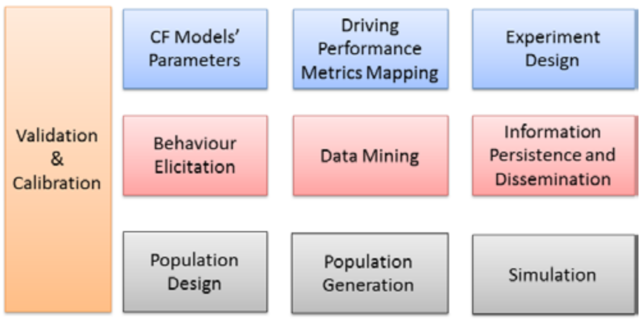


Figure 20-1: Proposed methodology for embedding driving styles in traffic simulation.

Next, the driving behaviour elicitation phase is where users participate in the designed experiment and data is collected. When data collection is concluded, multiple data mining related tasks can be performed in order to extract relevant information. It is expectable that tasks such as clustering, stereotype extraction, and even classification are useful to keep the information organized. The creation of clusters (groups with similar behaviour) can be very convenient (i) they provide general overview of driving behaviour population, (ii) as multiples drivers can belong to a single group, the set of driving behaviour groups does not scale linearly with the number of participants, and (iii) the compression of elicited behaviour in a set of groups, properly characterized, eases the dissemination of that information. In this phase, a calibration procedure can be performed at the individual driver level by measuring the differences between the output of the extracted profile and the actual driving performance in a similar scenario. At group level, careful assessment should be made by analysing how efficient the clustering is by a mathematical perspective, while from a driving behaviour perspective the meaning of the identified group should be representative of a relevant class of driving style.

In the final phase, we proceed to generate the virtual population of drivers to our traffic simulations. However, it may be relevant the frequency with which a given behaviour should appear during the simulation. Thus, researchers should design the population behaviour distribution that is more appropriate to a given problem. This should be validated by taking a sample of drivers from the traffic system simulated in order to assess the assigned distribution. After the population generation, the parameters should be interpreted by the traffic simulator and the simulation can then begin. A final validation step should be made to compare the obtained results with the actual system (if possible).

20.4 Software Architecture

The software architecture proposed is composed of two subsystems in a distributed framework as presented in Figure 20-2.

In the Remote Domain (RD), there will be a remote server whose major tasks are serving as data repository and performing driving behaviour dissemination. So, this defines the boundaries between researchers who provide driving behaviour data content and those who just need to obtain a set of driving behaviours. Also, there would be implementations of driving behaviours since more naïve models may not be adequate to represent complex patterns. Thus users could retrieve the driving behaviour from the latest identified clusters (along with their parameters values) and their respective implementations.

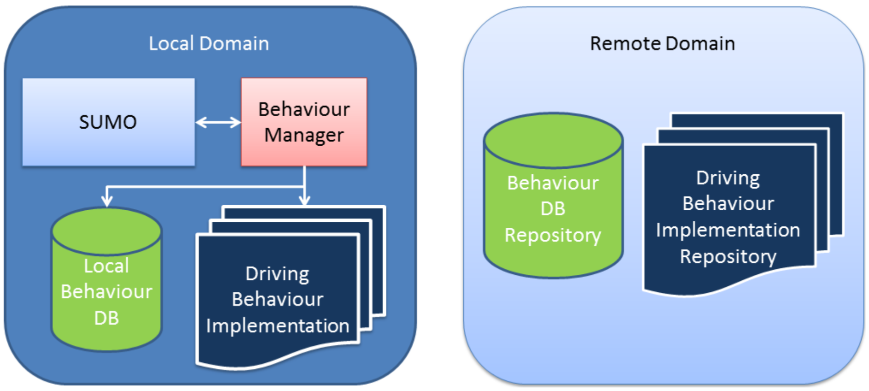


Figure 20-2: Architecture overview.

In the Local Domain (LD), there are two modules similar to the RD. In essence, these are subsets of the RD content, which were retrieved from the RD and only contain the user's desired content. A major component in this architecture is the Behaviour Manager which is responsible for interacting with the SUMO micro simulator in order to read and update the simulation values.

In practice, the Behaviour Manager works as a standard application connected to SUMO through TraCI and acts as a broker between the simulation engine and the agents that implement real driving behaviours. These agents have a set of vehicles to manage, they perceive the simulation through the Behaviour Manager, assigns updated values to the vehicles in order to simulate the driving behaviour and the Behaviour Manager finally commits the values in the simulation.

With such an approach we may sacrifice a bit of the simulation performance so as to get a flexible solution that will not require major changes to the SUMO microscopic simulator. An obvious bottleneck is the Behaviour Manager since after broadcasting the simulation state to the hosted agents it must wait for the responses and apply the changes to the simulator before requesting the engine to continue to the next step. Also agent coordination and communication with the Behaviour Manager may cause significant overhead in communication channels, especially in case of thousands of vehicles.

The main advantage is that researchers are free to develop their own agent communities' implementations along with their coordination techniques. This is valid as long as the Behaviour Manager is able to read the simulation state and update vehicle states in the SUMO engine. This flexibility gives the possibility to have a generic interface for integrating agents based on the Peer-Agent-Design [10] concept, which require more research since they are not yet fully developed.

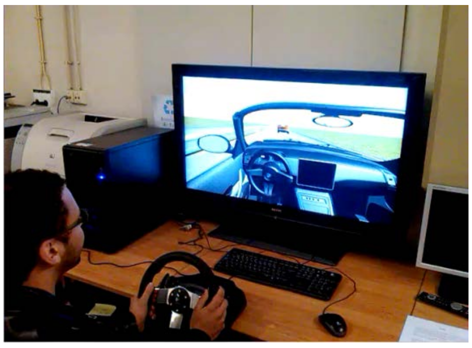
20.5 Preliminary velocity behaviour elicitation experiment

Despite the early stage of this project development, we already conducted some preliminary experiments to elicit behaviour from human drivers. More specifically, we collected data regarding the extraction of velocity clusters which will represent class of drivers with similar speed management while driving. Note that this experiment does not consider other vehicles; hence no interaction with headway vehicles or side vehicles in crossroads is considered. For this purpose we used a low-cost driving simulator we already developed in [8], as depicted in Figure 20-3. The hardware setup is composed by a Logitech G27 steering wheel, a 40-inch-screen TV, and the Serious Game software.

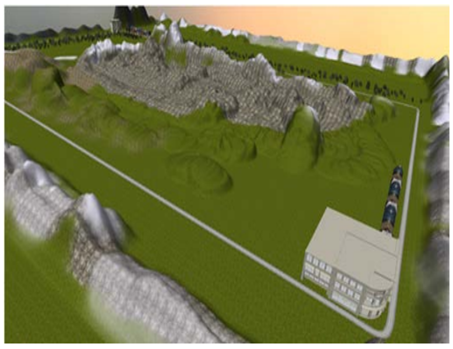
For this experiment we used a total of 9 participants that performed a 10-minute's period training for adapting to the driving simulator, and then they had to complete three laps from the map presented in Figure 20-3b. The participants were instructed to respect traffic rules, in particular the road markings and a velocity limit of 120 Km/h.

We defined as dependant variables the vehicle's position and its instant velocity; hence we can capture vehicle's position and velocity. As a result, we obtain two time series, with particular importance given to the speed time series.

After performing some preliminary analysis we identified two major behaviour changes: straight lines and curves. Hence we proceed to differentiate such situations by segmenting data according to the vehicle's position on the map so in the end we aim at obtaining the straight line velocity behaviour cluster (Svel) and the curve behaviour clusters (Cvel). Also, each driver's velocity time series were merged in order to compensate some eventual errors that could occur during the experiment, leading to more balanced driving behaviour patterns.



a) User taking experiments.



b) IC-DEEP map used.

Figure 20-3: Using IC-DEEP as a behaviour elicitation tool.

Table 20-1: Descriptive statistics of each straight line velocity cluster (units in Km/h).

Cluster	Max	Mean	Med	Min	Mode	STD	Var	Users
S_{vel_1}	118,19	108,50	110,46	90,38	109,77	6,08	36,93	U_1, U_5, U_9
S_{vel_2}	116,62	103,60	105,64	80,52	107,52	8,18	66,97	U_3, U_4, U_7, U_8
S_{vel_3}	113,90	99,98	100,49	85,33	100.39	6,99	48,81	U_6
S_{vel_4}	91,34	80,88	81,22	69,33	80,84	4,98	24,82	U_2

Concerning the data mining tasks, we conducted a computation of a cost distance matrix between all time series using the Dynamic Time Warping algorithm [9]. Once the matrix was finished we then proceeded to group users using Matlab's hierarchical clustering tool using the Ward method. In the end we obtain the logic groups based on their time series frequency pattern, meaning that even if they two series are desynchronized the method tolerates the difference as long as they have similar frequency pattern.

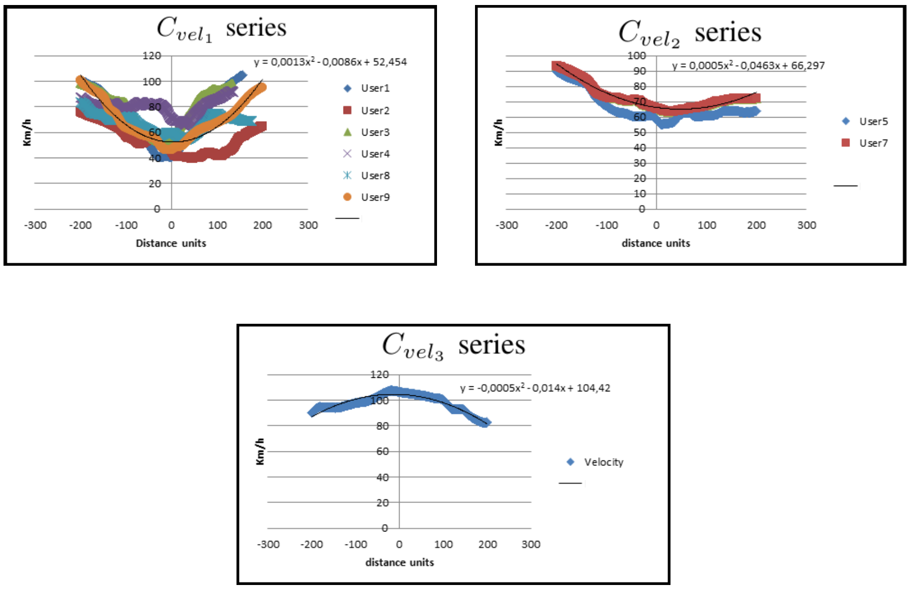


Figure 20-4: Graphical representation of velocity, as drivers handles curves, for each cluster.

In Table 20-1 we present the identified clusters for the straight line velocity. As we can see, even for a small sample it was possible to identify four different groups. We can observe the "spectrum" from a top group that could be considered more aggressive driving due to their highest velocity and standard deviation values, while in the bottom we have a more presumably defensive approach to velocity in straight lines.

As for the curve approach and leaving, we identified three different clusters presented in Figure 20-4. We interpret the results similarly to the first one as a normal approach since that was already expected by simple observation of people driving. Concerning the second, we observed an approach similar to the first one but the leaving phase is much smoother. And finally the third group can be interpreted as a very sportive driving style since the velocity is kept under high values.

20.6 Conclusions

In this paper we present a methodology to extend SUMO for enabling more complex driving behaviour models than those currently implemented in some microscopic simulators. Our aim is to ultimately equip SUMO with the necessary tools for representing individual vehicles not just as a set of vehicles performance parameters with a shared driving behaviour, but rather as a simulation population that has a set of vehicle types and a set of driving behaviour models assigned to each vehicle.

We propose that the software architecture support sharing and disseminating elicited data, since it is expectable the elicitation phases will be time consuming and computationally expensive. While on the one hand a remote domain system stores information from driving behaviour elicitation and disseminates the extracted clusters with other researchers, on the other hand we have a local domain with the subset that is interesting to a concrete SUMO experiment. In order to abstract concrete driving behaviour implementations we defined a broker element, the Behaviour Manager, to extend the SUMO simulation and to enable researchers to use their desired tools for creating driver communities.

Overall, our methodology is based on eliciting behaviour from actual drivers in order to generate a virtual society of drivers that represent real-world counterparts in transportation systems. We present results from elicited speed behaviour management using a low-cost simulator on a sample of drivers. As expected the results detected different speed behaviour patterns from different drivers under the same conditions. This allows us to identify generic behaviour patterns which can be used to characterize individual drivers in traffic network simulation settings. We believe this approach will improve the overall realism of traffic simulations by ensuring the behaviour of vehicles mimic real drivers at an individual scale.

20.7 References

[1] G. Dimitrakopoulos, Intelligent Transportation Systems. IEEE Vehicular Technology Magazine, vol. 5, issue 1, pp 77-84, 2010.

[2] Z. Kokkinogenis, L. Passos, R. Rossetti and J. Gabriel, Towards the next-generation traffic simulation tools: a first evaluation. Doctoral Symposium on Informatics Engineering , 2011.

[3] B.C. Silva, A. Bazzan, G.K. Andriotti, F. Lopes, D. Oliveira, Itsumo: An intelligent transportation system for urban mobility. LNCS Springer, no. 3473, pp 224-235, 2006.

[4] S. A. Boxill, L. Yu, An evaluation of traffic simulation models for supporting ITS development. Technical, Transportation Training and Research, Texas Southern University, USA, 2000.

[5] F. Zhang, J. Li, Q. Zhao Single-lane traffic simulation with multi-agent system. IEEE Conference on Intelligent Transportation Systems, pp. 56-60, 2005.

[6] B. Chen, H.H. Cheng A review of the applications of agent technology in traffic and transportation systems. Trans. Intell. Transport. Sys. vol. 11(2), pp. 485-497, 2010.

[7] B. Burmeister, A. Haddadi, G. Matylis, Application of multi-agent systems in traffic and transportation Software Engineering. IEE Proceedings, vol.144, no.1, pp.51-60, Feb 1997.

[8] J. Gonçalves, C. Olaverri-Monreal, R. Rossetti. IC-DEEP: A serious games based application to assess the ergonomics of In-Vehicle Information Systems, Proceedings of the 15th Intelligent Transportation Systems Conference, Anchorage, AK, USA, 16-19 Sep. 2012.

[9] L. Matias, J. Gama, J. Mendes-Moreira, and J. Sousa, Validation of both number and coverage of bus Schedules using AVL data, 13th International IEEE Annual Conference on Intelligent Transportation Systems, Madeira, Portugal, 2010.

[10] R. Lin, S. Kraus, Y. Oshrat, Y. Gal. Facilitating the evaluation of automated negotiators using peer designed agents. Proc. of The 24th Association for the Advancement of Artificial Intelligence (AAAI-2010).

21 Road Intersection Model in SUMO

Daniel Krajzewicz , Jakob Erdmann
German Aerospace Center, Berlin, Germany

21.1 Abstract

Besides basic models for longitudinal and lateral movement, a traffic simulation needs also models and algorithms for right-of-way rules. This publication describes how passing an intersection is modeled within SUMO, including a description of an earlier and the currently used model.

Keywords: Road Traffic Simulation, Intersection Model.

21.2 Introduction

SUMO [1][2] is an open source road traffic simulation package developed at the Institute of Transportation Systems at the German Aerospace Center. Being a microscopic road traffic simulation – each vehicle is modeled explicitly – it uses a car-following and a lane changing model for computation of a vehicle's longitudinal and lateral movement, respectively. The models used in SUMO were initially described in [3]. For simulation of real-life networks, further models are necessary. This paper describes the current implementation of the intersection control model used in SUMO. An earlier model which is also described was tied to a fixed simulation time step length of 1 second. The current model was implemented in order to enable variable simulation step lengths.

The rest of this document is structured as following: At first, the original and the currently used model for intersection control are described. Then, the model used for simulating the speed determination of a vehicle at an intersection without the right of way is presented.

21.3 Intersection Model

Generally, road networks are represented as graphs in SUMO. An intersection ("node") consists of incoming and outgoing edges, where an "edge" represents a road with one or more lanes. Each lane has a unique id which is derived from the edge id and a number representing the lane index starting with 0 at the rightmost lane. Within an intersection, lie so called "internal lanes" which connect the incoming lanes with the outgoing lanes. Vehicle movements across an intersection proceed along these internal lanes just as they would on regular lanes.

A lane may have more than one successor lanes. The connectivity among lanes is defined with "links". In older versions of SUMO, before the introduction of internal lanes, there was a link between an incoming lane and an outgoing lane. Since the introduction of internal lanes there is a link between an incoming lane and an internal lane (called an "entry link" and another link between the internal lane and an outgoing lane (called an "exit link") as shown in Figure 21-1. The entry links of an intersection are numbered from 0 to n. Since there is exactly one exit link for each entry link, the link index uniquely defines a connection across the intersection from an incoming lane to an outgoing lane. The link indices are computed using the following scheme: first, the incoming edges are sorted in clockwise fashion. Then, the lanes, starting at the top-most are traversed. The links outgoing from a lane are then iterated, starting with the right-most destination, relative to the incoming edge. These link indices are used when discussing right of way computations.

At most intersections, vehicles wait at the stop line at the border of the intersection until they may cross conflicting streams of traffic. However, on some types of intersections, left-turning vehicles are allowed to wait in the middle of the intersection. This is modelled in SUMO by splitting internal lanes at the halting position and introducing an "internal intersection" that lies within the original intersection. Vehicles using these internal lanes always pass the entry link to the intersection and then wait at the internal intersection instead. The right-of-way computation for internal intersections follows the same principles as that of regular intersections.

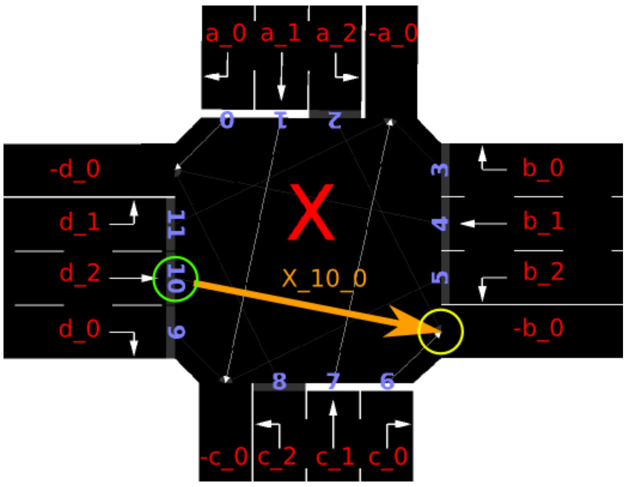


Figure 21-1: Intersection model terminology in SUMO. The intersection has id *X* and features the incoming roads *a,b,c,d* and the outgoing edges, *-a,-b,-c,-d* The connection from incoming lane *d_2* to outgoing lane *–b_0* crosses *X* on the internal lane *X_10_0*. The entry link with index 10 is circled in green. The exit link is circled in yellow.

21.3.1 Earlier Model

The right-of-way model that was implemented in the initial release of SUMO is a strong simplification of real world behaviour. When approaching an intersection a vehicle at first set the information about its approach to the intersection. After this has been done for all vehicles, the intersection "decides" which vehicles are allowed to pass without braking and which vehicles have to yield. This is done using a right-of-way matrix. This matrix describes which connections cross each other and which one has the right of way in case of crossing connections. This concept is illustrated in the following using an example.

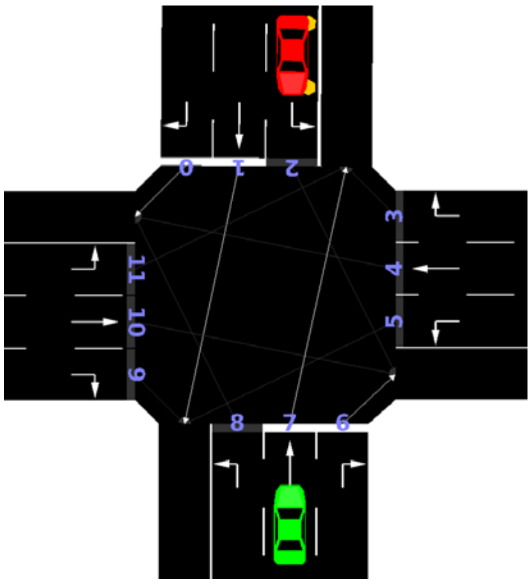


Figure 21-2: Intersection model terminology in SUMO. The intersection has id *X* and features the incoming roads *a,b,c,d* and the outgoing edges, *-a,-b,-c,-d* The connection from incoming lane *d_2* to outgoing lane *–b_0* crosses *X* on the internal lane *X_10_0*. The entry link with index 10 is circled in green. The exit link is circled in yellow.

Figure 21-2 shows an intersection which is approached by a red car on link 2 and a green car on link 7. Since the paths of both vehicles intersect and both wish to cross the intersection in overlapping time intervals, a right of way computation is performed. In Figure 21-3 the right-of-way matrix for this intersection is shown, emphasizing the discussed links. The matrix cell with row *i* and column *j* defines the right of way for a vehicle on link *i* in regard to a vehicle from link *j*. According to the colors (white/yellow/red) a vehicle on link *i* (ignores/has priority over/yields to) a vehicle on link *j*. In the example, the red car (link 2) yields to the green car (link 7) because of the red box in cell (2,7) which agrees with the common rules of traffic for left-turning vehicles. So, we see that the vehicle at link 2 has to wait for the vehicle at link 7.

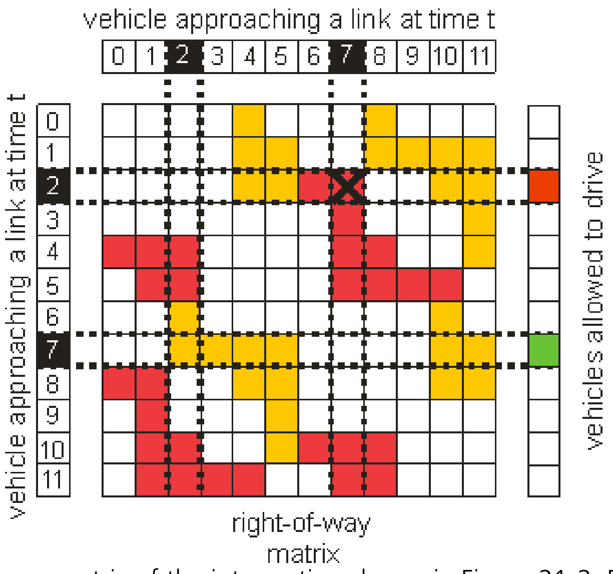

vehicle approaching a link at time t
0 1 2 3 4 5 6 7 8 9 10 11
vehicle approaching a link at time t
0 1 2 3 4 5 6 7 8 9 10 11
vehicles allowed to drive
right-of-way matrix

Figure 21-3: The right-of-way matrix of the intersection shown in Figure 21-2. Row i corresponds to the crossing/priority relation for link i. Link 7 crosses links 2,3,4,5,10,11 but has the right of way (yellow boxes). Link 2 crosses links 4,5,6,7,10,11 but must yield to 6 and 7 as indicated by red boxes. Since a vehicle approaches on link 7 (in the relevant time interval) the vehicle on link 2 has to brake.

The matrix itself is static and computed during the network import/generation. Traffic lights were modelled by removing the information about approaching the intersection for all vehicles that run at links that have a red light. Ignoring these vehicles during the right-of-way computation prevents them from blocking other vehicles and has the side-effect of not giving them the permission to pass the intersection.

Even though this model works well for simulation steps of one second, it caused problems when implementing sub-second time steps. Because the decision about letting a vehicle pass the intersection is performed in each time step, vehicles must not drive faster than their maximum braking ability multiplied with the step size time when being in front of the intersection. This is necessary to ensure that the vehicle can still brake if another vehicle with higher priority suddenly approaches. When decreasing the duration of simulation steps, this velocity is decreasing by the same factor, too, as depicted in Figure 21-4.

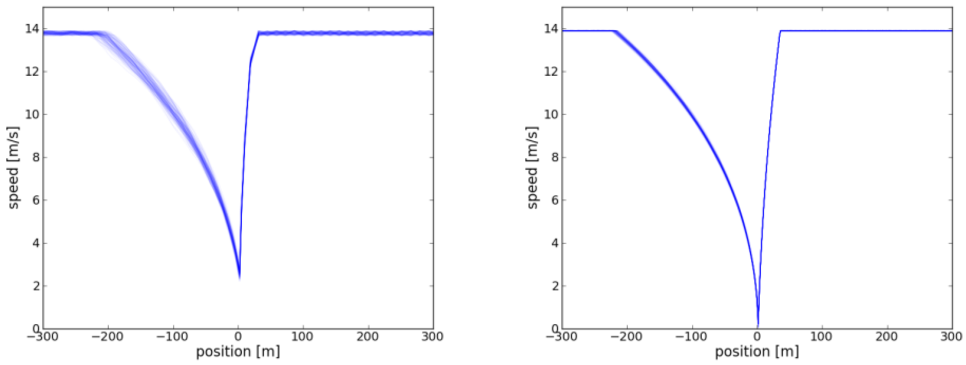


Figure 21-4: Vehicle speed when approaching an intersection in the old model; a) simulation steps of 1s, b) simulation steps of 0.1s.

This false behaviour for lower step times was the motivation to change the intersection control algorithm. Another motivation was the need to model the interaction between vehicles which occupy the intersection simultaneously. This became necessary after the introduction of internal lanes which allow the full range of longitudinal vehicle behaviour specifically unpredictable decelerations. At the end, transferring the logic for passing an intersection from the intersection model into the driver model is assumed to be a development step into the right direction, allowing further work on driver behaviour modeling.

21.3.2 Requirements for an improved Model

The goal for an improved intersection control model was to support all types of intersection typically found on European roads and to allow for realistic simulation dynamics. The following intersection types are deemed necessary:

- Intersections without prioritization (right-before-left),
- Prioritized intersections with
    - Different directions of the prioritized road (straight, turning),
    - Unprioritized lanes with yield or stop signs,
- Intersections controlled by traffic lights.

Important aspects of realistic intersection dynamics are the following:

- No deadlocks,
- No collisions,
- Efficient use of the intersection,
- Realistic acceptance gaps,
- Approaching unprioritized links without stopping,
- Qualitative dynamics independent of the simulation step length.

At the time of this writing all these goals have been met except the implementation of stop signs which is planned for a future release.

21.3.3 Current Model

The key to correct the deficiencies of the original model described in Section 21.3.1 was to not only consider the current time step, but give the right of way based on information about oncoming vehicles including an extrapolation of their time of arrival at the intersection into the future. To do so, each vehicle informs the entry link about its approach. In contrary to the initial model, not only the approach as such is stored, but also the expected time of arrival at the intersection and the speed of this arrival. Using this information, the time within which a certain link over the intersection is occupied can be computed. Each entry link also stores information about its "foe links". This corresponds to the red boxes in one row of the right-of-way matrix shown in Figure 21-3. When approaching an intersection (an entry link), a vehicle computes how long it will occupy the intersection and then checks against all approaching vehicles in all foe links of its entry link. If the requested time slot is separated from all approaching foe time slots by a suitable safety gap the vehicle is allowed to pass the entry link and thus enter the intersection. The size of the safety gap depends on the speed difference between the vehicle and its approaching foes and is set to a minimum value of 1 second.

A vehicle informs the entry links to the next few intersections on its current route (up to a distance of about 3000m) about its approach. Due to the advance knowledge of approaching foe vehicles, a vehicle approaching on an unprioritized link cannot be “surprised” by the sudden appearance of a foe. This allows decoupling the approach speed from the simulation step size. Instead, vehicles decelerate until they are one second away from the intersection. If braking is not necessary at this point they can safely accelerate and cross the intersection. The value of 1 second models an intersection with average visibility.

Once a vehicle enters an intersection by passing the entry link, this link is no longer informed. Since vehicles follow normal movement rules while on the intersection they may brake while on the intersection or even come to a stop. Therefore, other vehicles require an additional mechanism for keeping track of vehicles currently on the intersection and to avoid collisions. The goal is to use the existing functionality for letting vehicles follow each other at safe speeds. Normally, this functionality is active for vehicles which move on identical or subsequent lanes. At an intersection however, vehicles are on different lanes which cross somewhere on the intersection or merge into the same outgoing lane.

To be able to use the car following functions two things are required

1. A vehicle needs to know the lead vehicle;
2. There must be a well defined distance between the follower and the lead vehicle.

The first point is accomplished by declaring the first vehicle of any two vehicles to enter the intersection as the leader. This is particularly important, because several vehicles may be driving within the space of the intersection at the same time and there must be an antisymmetric, transitive and irreflexive leader relation among them to avoid deadlocks. The second point is accomplished by virtually superimposing both internal lanes up to the crossing point. If both internal lanes merge into the same outgoing lane, the crossing point is naturally the beginning of the outgoing lane. Let follower-vehicle *F* have distance *d_F* from the crossing point and let its leading vehicle *L* have distance *d_L* from the crossing point then the virtual gap g between both vehicles is defined as

$$g := d\_F - d\_L - length(L) - minGap(F)$$

where *length(L)* is the physical length of vehicle *L* and *minGap(F)* is the minimum gap that vehicle *F* intends to keep to its leader at all times. Note that *g* may be negative which causes vehicle *F* to stop.

In the current implementation each exit link maintains a list of "foe internal lanes". These are the lanes which correspond to the yellow and red boxes in one row of the right-of-way matrix in Figure 21-3. In other words, these are the internal lanes which intersect with the internal lane the approaching vehicle intends to use.

A vehicle that wishes to pass an exit link on its route asks this link for any additional vehicles to which it must adapt its speed. These vehicles are called link leaders. The link checks all of its foe internal lanes for occupancy, computes the virtual gap and returns each found vehicle as a potential link leader to be followed.

Figure 21-5 shows the same intersection as Figure 21-1 with three vehicles green (G), blue (B) and red (R). Vehicle R wishes to pass the exit link that belongs to link 11. Both vehicles G and B are on the same internal lane which is a foe internal lane for link 11. On the left side of Figure 21-5, vehicles G and B are potential link leader for R. Since G and B have entered the intersection before R, they will both be followed. In this case only the speed adaption to B is relevant since B is already following G. On the right side of Figure 21-5 the situation is slightly different. Vehicle R has already entered the intersection before vehicle B and therefore, R only follows G.

In the current implementation each vehicle maintains a list of link leaders being followed for each exit link. This list is used when maintaining the antisymmetric link leader relation among vehicles (vehicle R only sets vehicle B as its link leader if B does not already have R as its link leader).

The link based model of detecting conflicting approach information coupled with the handling of link leaders allows for full vehicle dynamics on the intersection together with efficient use of the intersection as a natural extension of car following.

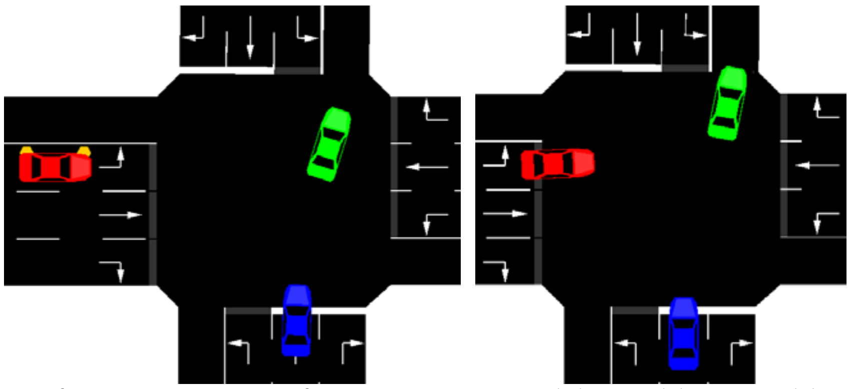


Figure 21-5: Examples of link leader relations for the vehicles green (G), blue (B) and red (R). In the left figure G is the leader of B and both are the leaders of R. In the right figure G is the leader of R and R is the leader of B because R entered the junction before B.

21.4 Approaching Links

The behaviour when approaching an intersection without having the right-of-way was consolidated for different time step sizes in the following way.

When the ego vehicle approaches an intersection where other vehicles may have the right of way, forcing ego stop, it decelerates to a velocity which allows braking in front of the intersection. One second before reaching the intersection, the ego vehicle decides whether the intersection may be crossed or not. If crossing is possible at this time, the vehicle may accelerate again, otherwise it decelerates to a velocity of zero.

Figure 21-6 shows that using this definition assures the similar behaviour for different simulation step sizes. The velocity used for approaching the intersection is the vehicle's deceleration capability multiplied with 1 s. For the standard Krauß parameters it is equal to 16.2 km/h, what was found to be empirically valid when compared to measures obtained from test drives with DLR test vehicles. Within the current model, the vehicle's maximum deceleration ability is used for all intersections and all directions of driving across them. Because in reality, this speed is mainly dictated by the possibility to look into foe lanes for determining whether the intersection may be crossed, further extensions of the model, in means of differing between approach velocities promise to improve the model's quality. It should be also noted that the simulated time line of deceleration and acceleration is not yet matching the reality.

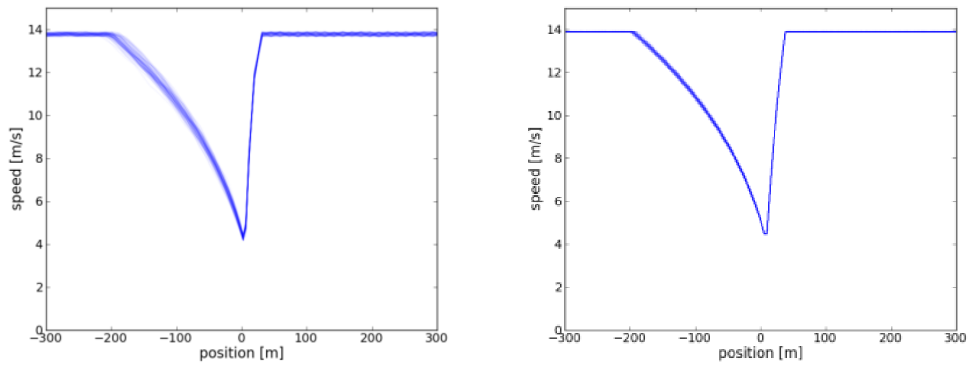


Figure 21-6: Vehicle speed when approaching an intersection in the new model. a) simulation steps of 1 s, b) simulation steps of 0.1 s.

21.5 Estimation of Link Entry/Exit Times

In the following, the estimation of times and speeds of arrival and leaving an intersection is discussed. Figure 21-7 shows the deviations of the estimated speeds and times over time for a major (high prioritized) link. These vehicles do not have to break. "deviation" denotes here the difference value obtained by subtracting the real from the estimated value in the following Figures. One may see that the times of arrival and leaving are both estimated too low and only increasingly move towards the correct value. This is due to the random "dawdling" behaviour of SUMO's default car-following model, see [4]. If the dawdling is disabled, the estimation is correct from the very begin on (not shown, here). The deviations in time are due to the same reason. They are straight, as in each time step, the estimation is based on the perfect speed (50 km/h in the shown example) and the dawdling is performed by the model afterwards. It should be noted, that the estimation could be more correct, if the dawdling, regarding its stochastic nature, would be taken into account during the computation of the times/speeds.

The additional error in estimating the leave time is probably due to taking into account the distance to the leader in jam/when standing (SUMO's "minGap" attribute of a vehicle type), which was set to 2m in the shown example; 2 m divided by 13.89 m/s gives the offset shown here, which is about 0.14 s.

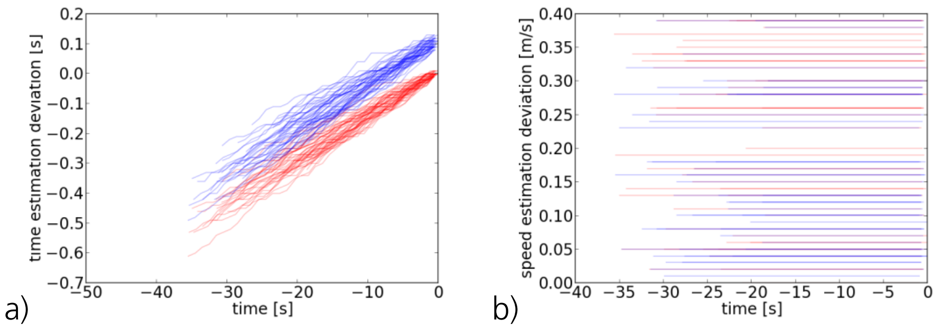


Figure 21-7: Deviations of the estimated time (left) and speeds (right) from their final counterpart for arrival at (red) and leaving an intersection (blue). High prioritized vehicles

The difference in starting times is due to using a random position for the place the vehicle departs at. This was done for adding randomness into the possible co-occurences of vehicles at high and at low prioritized roads. The behaviour of vehicles on prioritized roads is straightforward and can be easily explained, see above. But the behaviour of vehicles that have to react to crossing traffic are more complicated. Shown in Figure 21-8, the shape of time estimation development has three peculiarities. The first are large overestimations of the arrival and the leave time by about 260 s. The second can be seen better when focussing on the majority of traces, as done in Figures 21-8b and 21-8d. They show that the speed is – in addition to the continuous progress towards a correct value – oscillating with an amplitude of 2 s. The reason could be the dawdling, as discussed for vehicles approaching a major intersection. But, when looking at the same run with a dawdling value set to zero, as visualised in Figures 21-8c and 21-8d, some oscillations are still visible. The third peculiarity is an overestimation shortly before the link is reached.

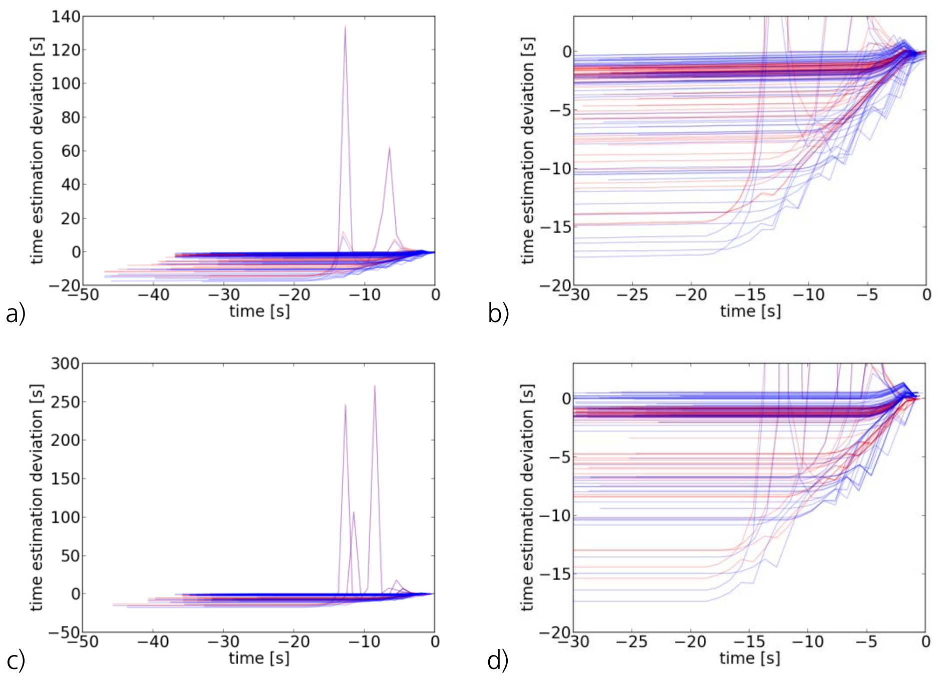


Figure 21-8: Deviations of the estimated time from their final counterpart for arrival at (red) and leaving an intersection (blue). Low prioritized vehicles. Top: with default dawdling, bottom: with no dawdling, left: the complete figure, right: focus on the majority of approaches

At the current time, these effects cannot be explained.

21.6 Summary

The currently implemented model for right-of-way rules at intersections was presented. It was shown that it is capable to work with different time steps. Additionally, some initial evaluations of the approaching behaviour were given. Besides additional explanations, missing at the current time, comparisons against real-world trajectories should be performed.

21.7 References

[1] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO – Simulation of Urban MObility: An Overview. In: SIMUL 2011, The Third International Conference on Advances in System Simulation (2011)

[2] DLR and contributors: SUMO homepage. http://sumo.sourceforge.net/ (2013)

[3] Krajzewicz, D.: Traffic Simulation with SUMO – Simulation of Urban Mobility. In: Barceló, Jaume (Ed.): Fundamentals of Traffic Simulation, Series: International Series in Operations Research & Management Science, Vol. 145, Springer, ISBN 978-1-4419-6141-9 (2010)

[4] Krauß, S., Wagner, P., Gawron, C.: Metastable states in a microscopic model of traffic flow. Phys. Rev. E, American Physical Society, 1997, 55, pp. 5597-5602, (1997)

22 Authors Index

ISSN 1866-721X