System simulation from operational data *

Armin Wasicek
University of California,
Berkeley, CA
arminw@berkeley.edu

Edward A. Lee
University of California,
Berkeley, CA
eal@eecs.berkeley.edu

Hokeun Kim
University of California,
Berkeley, CA
hokeunkim@eecs.berkeley.edu

Lev Greenberg
IBM Research Labs
Haifa, Israel
levg@il.ibm.com

Akihito Iwai
DENSO International America
San Jose, CA
akihito_iwai@denso-diam.com

Ilge Akkaya
University of California,
Berkeley, CA
ilgea@eecs.berkeley.edu

ABSTRACT

System simulation is a valuable tool to unveil inefficiencies and to test new strategies when implementing and revising systems. Often, simulations are parameterized using offline data and heuristic knowledge. Operational data, i.e., data gained through experimentation and observation, can greatly improve the fidelity between the actual system and the simulation. In a traffic scenario, for example, different road conditions or vehicle types can impact the outcome of the simulation and have to be considered during the modeling stage. This paper proposes using machine learning techniques to generate high fidelity simulation models. A traffic simulation case study exemplifies this approach by generating a model for the SUMO traffic simulator from vehicular telemetry data.

Categories and Subject Descriptors

[**Modeling and simulation**]: Simulation support systems; [**Physical sciences and engineering**]: Engineering—*Computer-aided design*

Keywords

Model generation, Machine Learning, Traffic simulation

1. INTRODUCTION

The Internet-of-Things (IoT) and similar efforts push on the acquisition of large data sets. The IoT infrastructures continuously monitor and record the operation of systems and their environment. The availability of these data sets is a game changer not only for the operation of the systems themselves, but also for their design. By being specific to a certain system, the data can give insights on its inner working that a general textbook model would fail to provide. Moreover, the adaption of manually created models may be difficult, error-prone, or even not feasible. Automating modeling efforts based on data facilities adapting a model to reality.

Systems simulation is a major tool to approach various real-world tasks or processes. Simulation is used to gain insights into the nature of a problem or to validate potential solutions. Indeed, in many complex operational environments executing multiple field test runs could be expensive or dangerous. Moreover, in initial design phases there is no way to execute real physical tests, so system simulation is a frequently used technique. A simulation comprises models that capture behavior of relevant entities, rules that governs their interaction, and a context that assigns a semantics to the simulation. By running simulations, designers infer properties of their system realizations. A plurality of modeling formalism, styles, and strategies already exist and new ones are being researched.

Considering existing modeling and simulation approaches, some high-level challenges overarch different approaches. *Model fidelity* is concerned with how closely a model matches the system being modeled. Model fidelity determines the validity of any inference made about system being modeled, and it is always approximate. Methods to estimate model fidelity include reference measurements. Similarly, *model calibration* involves adjustment of model parameters to improve fidelity. Both steps require data acquisition and processing.

In this work, we introduce *data-driven design* as a means to analyze and solve system design problems using system simulation. We discuss *model generation* as an integral step to create simulation models out of operational data. Model generation aims to automate the system modeling process while maintaining model fidelity by using operational data. To this ends, machine learning and its ability to generate models and system parameters out of data is a key technique. We highlight some useful techniques by calibrating a traffic simulation using vehicular On-Board Diagnostics

*This work was supported in part by the TerraSwarm Research Center, one of six centers administered by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA; the iCyPhy Research Center (Industrial Cyber-Physical Systems, supported by IBM and United Technologies); and a Marie Curie IOF Action within the 7th Framework Programme under the funding ID PIOF-GA-2012-326604 (MODESEC).

DAC '15, Jun 07-11, 2015, San Francisco, CA, USA
ACM 978-1-4503-3520-1/15/06.
http://dx.doi.org/10.1145/2744769.2747944

(OBD) data. The general approach, however, is applicable to a much wider range of applications.

The organization of the paper is as follows: Section 2 introduces data-driven design as a method to generate system simulations out of measurement data. The next section discusses machine learning techniques to generate models out of operational data. In the case study in section 4, we illustrate our ideas in a mobility scenario. Sections 5 and 6 discuss related work and draw a conclusion, respectively.

2. DATA IN DESIGN

Design Space Exploration (DSE) anticipates the impact of certain design decisions. As a part of a systems engineering approach, it aims to find feasible designs and make good architectural decisions prior to actually implementing a particular system [18]. DSE requires the availability of a model representation of the system under consideration and its environment (e.g., using SysML). Often, a good starting point is to reuse a previous design. Following a certain engineering process (e.g., the V-model), a campaign of experiments and simulations are then set out to refine these models and to gain insights on the quality of potential solutions. Possible actions include evaluating and interpreting data as well as adapting engineering models. Frequently, data generated during such a campaign is aggregated to optimally calibrate models, e.g., by fitting model parameters. Other data sources that are used in the design process are technical standards, databases, individual experiences, and 'rules-of-thumb.' We refer to a data set as *operational data* if it has been acquired through observation or experimentation like on a prototype, a legacy system, or a partially functioning system.

Incorporating information from operational data is a crucial step in modeling and can result in more specific simulation models than using general knowledge. For instance, a traffic simulation can use a general model of a passenger car. Distinguishing between different types of passenger cars, e.g., sports cars, minivans, SUVs, and using their specific properties might increase the fidelity of the simulation. As a matter of fact, the required modeling effort increases with system's degree of complexity. Operational data can help to mitigate this effect by automating the generation of models or partial models, thus, leading to more fine-grained simulations.

It is convenient to separate between system and environment, where the system is a designed solution for some business needs that operates in a given environment. *Environment models* are used to provide stimuli for system models and evaluate system performance under different conditions. Typically, users of the systems are also part of the environment. So, for example, in case of a new road design, the existing road networks, transportation demands, and driver-cars dynamic behavior are parts of the system environment.

While the prime goal of the design process is design of a system and its models, significant effort is also invested in developing environment models. The later are used to evaluate and validate the system design. One of the approaches for reducing the environment modeling effort is to record environment behavior and provide it as a stimulus to a system model during simulation. One of the main limitations

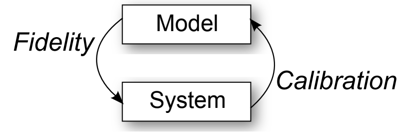


Figure 1: Relation between model fidelity and model calibration

of this approach is that a recorded environment can't react to system behavior, even though in many cases the system does significantly influence environment behavior. Another common approach is to use generic models of the environment, but these models have low fidelity, and the resulting simulation will also have low fidelity.

The approach described in this work (see section 3) uses machine learning techniques in the design process to generate high fidelity models from operational data. This technique significantly reduces the manual effort, while preserving the quality of the models. Moreover, an update of the models can be fully automated, so it can be used to generate models for multiple environment configurations, for example taking into account time-of-day dependencies. System models will be evaluated and validated for the selected environment configurations.

An additional advantage of the proposed approach is an automatic adaptation of models for environment changes over time. Indeed, the traffic demands and road network might change over time and these changes should be taken into account to optimize system performance. This can be done in an on-demand mode, where model calibration is triggered manually, for example, only if a significant change is detected in the environment behavior, or in an on-line mode, where models are continuously calibrated.

Environment models can be used at different phases of a product life cycle, starting from an initial design phase of a new system, where different design alternatives' performance should be compared, through system upgrades to validate correctness, to operational planning, where an optimal system configuration needs to be selected for current or planned scenarios. In all these cases, the proposed approach significantly reduces modeling effort and improves the validity of the analysis.

3. MODEL GENERATION USING MACHINE LEARNING

The model generation task systematically estimates and adjusts model parameters using operational data. The goal is to maximize the fidelity of the model for both the system and the environment [10]. The best operational data answers specific questions — when data are specific, taking action on them becomes easier. Consequently, we strive to generate a specific model out of observed data. Automating this process further increases its value: situation-specific simulations can be generated with little effort. For this automation, we rely on machine learning techniques to extract parameters from an operational data set.

The field of machine learning [12] is concerned with building models from example inputs and using these models to infer underlying patterns in data and to predict future behavior. The learning goal is to infer a model from observations. Model generation subsumes model calibration [13]. Model calibration works to determine the model parameters, whereas model generation also aims to reveal and integrate structure in the resulting simulation model.

Model generation encompasses all steps that are required to bridge the gap between an operational data set and the input language of a domain-specific simulator. Broadly, the following problems have to be solved:

1. *Specify variable model properties:* Reduce the input language of the simulator to a subset that fulfills the pragmatics of the design problem at hand. Prior information on the required model properties should be explicitly incorporated, for instance, by creating a template.
2. *Find and select predictor variables:* The operational data should contain data points that can serve as a predictors for these variables.
3. *Select an appropriate learning technique:* It is an art to develop an algorithm that connects the predictor variables with the required model properties. Many general machine learning techniques exist that can be tailored to a specific problem instance.
4. *Assess model fidelity:* Check how well the generated model approximates reality. For example, compare simulation traces and observations.

The result of this process should be an operational model that is ready to be executed in a simulator to gain insights on a design problem.

3.1 Example: Change Point Detection

The key requirement on machine learning tools that infer models given operational data is the ability to capture abrupt changes in the data that suggest parameter calibration, or in extreme scenarios, a switch between learned models. Change Point Detection (CPD) is a paradigm that aims at identifying the changes in the generative parameters of operational time-series data.

Bayesian techniques are often used for CPD tasks, for which the machine learning model consists of a prior belief of the occurrence frequency and distribution of change points within the data. CPD then yields a posterior maximum-likelihood estimate of the change points in the time series.

Formally, the CPD problem operates on a sequence of observations $\mathcal{X} = \{x_1, x_2, ..., x_T\}$ and yields M non overlapping partitions ϕ_i of the data such that

$$\bigcup_{i=1}^{M} \{\phi_i\} = \mathcal{X}\,. \tag{1}$$

For each partition $\phi_i, i = 1, 2, ..., M$, the observation points within the partition are independent and identically distributed (i.i.d.) according to

$$x_k \sim P(x|\eta_i), \quad x_k \in \phi_i \tag{2}$$

where η_i are the partition-dependent parameters that are also i.i.d. The reader is referred to [1, 6] for further algorithmic details of the detection process.

4. TRAFFIC ANALYSIS USING VEHICLE DATA

To illustrate system simulation using operational data, we leverage the telemetry data collected from vehicles to model the commute scenario on San Pablo Dam Road in Orinda, California. We use SUMO (Simulation of Urban Mobility) [3] to simulate a traffic scenario. San Pablo Dam Road serves as an alternative route for commuters in the San Francisco East Bay heading southbound.

4.1 San Pablo Dam Road Scenario

San Pablo Dam road has been the subject of a previous study [24]. The original scenario has been developed to analyze flow oscillations and it has been included in the SUMO software distribution as a tutorial. The tutorial implements model calibration as an optimization problem. The result is an environment model of the San Pablo Dam Road that can be used to analyze the situation and to design a system like a traffic light controller optimizing throughput.

In the past study, vehicle arrival and passing times along the road have been manually collected. Eight researchers stood at different sections on the side of the road recording the time when a vehicle was passing. Thus, the raw data set contains sequences of timestamps at roadside locations. Prior knowledge has been introduced by using a map to determine distances between these locations as well as to abstract the road network. Based on the timestamps and the distances, the speed for a road segment can be computed. Most importantly, the stationing of the observers on the road has an impact on how the speed values in the segments look. Picking the observer positions might impact whether important spikes are observed, or may mitigate or amplify certain effects of the road like curves, slopes, or other traffic conditions. The authors of the original study decided to station the observers with increasing proximity towards one end of the road. San Pablo Dam road is a two-lane, bi-directional highway with a downstream traffic light that generates a bottleneck. There is very little vehicle overtaking and almost no side traffic. The tutorial calibrates a vehicle model to the collected traffic demand data by minimizing the error between the observed data to the simulation data. It uses a variable speed sign for constraining the outflow velocity so that the original (real world) network's outflow condition is preserved.

Our goal is to repeat the scenario using operational data from the vehicles. We leverage telemetry data that has been collected on San Pablo Dam road during different times of the day. We perform CPD on the speed data to segment the road in pieces of constant velocity.

4.1.1 The Connected Car

The Connected Car augments its fundamental functions to transport people and goods with increased connectivity and data processing capabilities. It uses wireless communication interfaces (e.g., WiFi, LTE) to connect to devices both inside and outside the vehicle, enhancing the in-car experi-

Table 1: Some driver-vehicle model parameters

Parameter	**Description**	**Unit**
Acceleration	The acceleration ability of vehicles of this type	m/s^2
Deceleration	The deceleration ability of vehicles of this type	m/s^2
Tau	A driver's reaction time	s
chosenSpeed = $\mathcal{N}$*(speedFactor, speedDev)*	SpeedFactor and speedDev describe a normal distribution's μ and σ parameters. Each time a vehicle is instantiated, the simulator samples this distribution to generate a *chosenSpeed* specific to this vehicle.	km/h

ence. The rate at which data is flowing from a Connected Car is growing dramatically. An IHS study [4] estimates that about 30 terabytes of data would be collected each day from the 152 million Connected Cars on the road in 2020, enabling a truly big data challenge. The ability to perform effective data analytics and the newly acquired communication capabilities trigger the transition of the automobile from an isolated system to a service-oriented architecture. Thus, the Connected Car is a key enabler for an era of smart mobility [14].

As many new cars are already being delivered as Connected Cars, older ones can be turned into Connected Cars by leveraging their On-Board Diagnostics (OBD) interfaces. The OBD-II standard specifies the type of diagnostic connector and its pinout, the electrical signaling protocols available, and the messaging format. Many different types of dongles are commercially available that expose a vehicle's OBD interface via USB, WiFi, or Bluetooth. In combination with a smartphone or an M2M modem, the dongle can upload telemetry data to the cloud. Besides user-centric applications like enriching the driving experience, reducing fuel consumption and emissions, or novel insurance models, the data from the Connected Car could also be useful for traffic authorities for real-time traffic reporting and to predict congestion.

For the purpose of the case study we collected speed profiles and GPS data while driving on the road. We use a commercially available OBD-II dongle and connect it via Bluetooth to an Android smartphone app that periodically polls the vehicle's statistics and forwards them via LTE to the cloud. We use this operational data to build models in the scope of a traffic optimization problem.

4.1.2 *SUMO - Simulation of Urban Mobility*

SUMO is a free and open sophisticated microscopic traffic simulator that is extensively used in the transportation studies community [3]. A SUMO model consists of vehicle types, road networks, routes that vehicles follow, and a demand model that determines how frequently certain route-vehicle combinations are activated. The model is represented in several XML files.

The driver-vehicle model in SUMO is an agent-based approach to conceptualizing driver behavior. It encompasses the behavior that a driver exhibits in prevailing traffic circumstances. The driver can make decisions regarding acceleration, deceleration, lane changes, using exits, etc. In the Hierarchical Control model of Michon [17] [19], the SUMO vehicle model would be in between the Strategy Level that is concerned with general trip planning and the Operational Level, which encompasses the car controlling process, and the Tactical level where maneuvering takes place. A driver-vehicle model captures how a driver will and can react to current traffic situations. It is defined by several parameters (see Table 1). SUMO's manual states:

> *"The maximum acceleration for example is not the car's maximum acceleration possibility but rather the maximum acceleration a driver chooses - even if you have a Jaguar, you probably are not trying to go to 100km/h in 5s when driving through a city."*

So what are good values to choose? Rather than estimating these values, operational data can be used to derive appropriate values. The tutorial uses an empirical data set as ground truth and fits the parameters of a driver vehicle model such that the error between the simulation model and the data is minimized. More specifically, the tutorial adjusts the behavior of the driver-vehicle model until it meets the empirically measured speed distributions of the last segment of the road network. Because of that, the road network in the tutorial has been simplified to two segments.

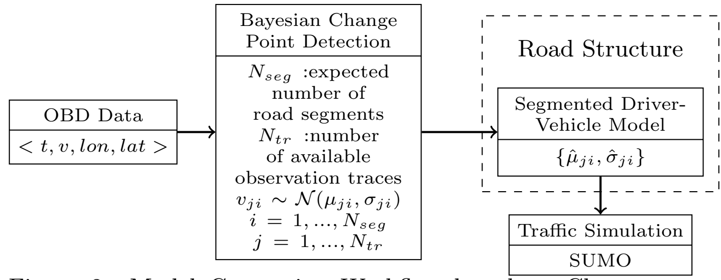


Figure 2: Model Generation Workflow based on Change Point Detection

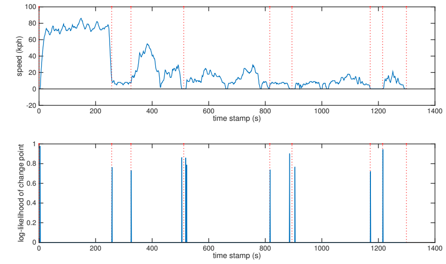
By using the telemetry data from the vehicles, we are able to get more fine-grained data and better speed measurements. In this case study, we use this data to improve modeling the San Pablo Dam Road situation in two ways:

- Finding an optimal segmentation of the road with respect to the collected speed profiles.
- Computing a representative speed distribution for a typical vehicle for each segment.

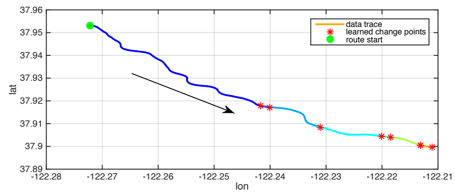
As more Connected Cars get on our roads, the data becoming available will be truly a bonanza for traffic authorities and researchers. Having more realistic simulation models will surely increase the quality of their decisions.

4.2 Change Point Detection on Speed Traces

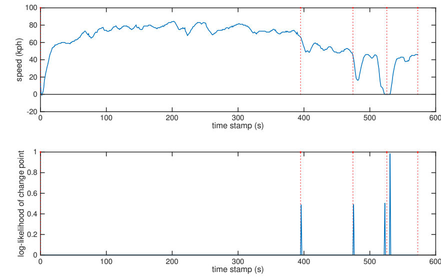
In this case study, we perform a Bayesian CPD on real-time speed traces from cars to infer a maximum-likelihood segmentation of the road (see Figure 2). Additionally, we



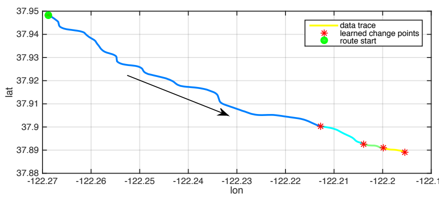
(a) Change Point Detection



(b) Trip map with change points
Figure 3: Rush-Hour Road Segmentation



(a) Change Point Detection



(b) Trip map with change points
Figure 4: Off-peak Hours Road Segmentation

get a characterization of the driver-vehicle model for each segment that is generated by the hidden parameters.

Adopting the CPD method (see section 3.1), where for each segment ϕ_i, the parameter set is characterized as $\eta_i = \{\mu_i, \sigma_i\}$ and μ_i and σ_i denote the mean and standard deviation of a univariate Gaussian distribution. Each sample observation $x_k \in \phi_i$ is assumed to be *i.i.d.* according to

$$x_k \sim \mathcal{N}(\mu_i, \sigma_i), \quad x_k \in \phi_i, \quad i \in \{1, ..., N_{seg}\}. \tag{3}$$

It is important to point out that the resultant model captures a composition of the environment components, as well as the structure of the system itself. Figure 3, for instance, depicts a speed trace collected during the rush-hour. The specific congestion pattern at that time leads to a significantly different final segmentation than a segmentation performed on a data set collected during off-peak hours on the same route (see Figure 4). Table 2 summarizes the number of segments and their respective speed distribution.

Segment	Scenario			
	Rush Hour		Off-Peak	
	μ	σ	μ	σ
1	69.64	15.81	71.57	12.50
2	7.38	1.28	52.72	4.11
3	25.78	14.61	29.97	16.14
4	12.64	7.26	33.58	15.79
5	5.83	2.52		
6	7.59	4.36		
7	0.01	0.07		
8	8.81	5.40		

Table 2: Learned Driver-Vehicle Model Parameters

Note that the CPD algorithm can work on several traces to mitigate the impact of an anomalous trace.

4.3 Comparison to Original SPD Scenario

We want to compare the results from our approach to the tutorial with respect to three aspects:

Segmentation: After running the segmentation algorithm, it became evident that both the simulation model from the tutorial [24] and ours have a similar segmentation. The length of single segments decreases towards the bottleneck. Assuming that the stationing of the transportation experts has been done strategically, we can conclude that our approach yields a similar model.

Data acquisition: Our data acquisition effort took much less time than the one of the tutorial. We were able to generate a similar model out much much less data. The tutorial was using eight observers plus one driver on two days for $2.5h$ summing up to $45h$. Our data acquisition takes as little time as a driver need to maneuver through the morning traffic which is roughly $23min$ times the number of vehicles to be measured.

Computational effort: Generating the model out of the observation data takes $8min$. This method uses the `fmin_cobyla` from the Python scipy package. Our custom CPD algorithm which is also implemented in Python runs for $2min$ when using 1300 traces.

5. RELATED WORK

Our proposed approach is closely related to microscopic traffic simulation models like SUMO [3], CORSIM [8] and VISSIM [7]. Microscopic models focus on driving behaviors of individual vehicles. For example, Shen *et al.* [23] address a traffic signal timing optimization problem. Hidas introduces SITRAS [9] which employs a microscopic traffic simulation for lane changing and merging on roads.

Parameterizing SUMO with vehicular data from a virtual reality driving simulator has been investigated in [15]. The iTETRIS simulation platform [22] uses vehicle position information retrieved from SUMO for large scale Intelligent Transportation Systems (ITS). Pan *et al.* [20] develop efficient re-routing strategies for avoiding traffic congestion.

For better reliability of simulation results, model calibration techniques for traffic simulation have been studied. Different types of procedures are proposed in [5] and [11] for calibration and validation of microscopic traffic simulation models. Liu *et al.* [16] carry out a feasibility study for automatic model calibration of traffic data using sensor and geography networking technology. Balakrishna *et al.* [2] and Park *et al.* [21] introduce application and case studies of model calibration in microscopic traffic simulations.

6. CONCLUSION

In this paper, we highlight model generation as a means to bridge the gap between an operational data set and its potential use in a system simulation. We propose to use machine learning to automate this task. Finally, we exemplify this by repeating a historic traffic simulation scenario using vehicle telemetry data. Traditionally, a model is often built to serve as an oracle to answer a specific problem. Our suggestion is to use specific models to enable designers to explore a problem and to find innovative solutions. With the ongoing rollout of Internet-of-Things technologies a huge amount of operational data will is becoming available. This data will radically change the way system simulations are built and used.

7. REFERENCES

[1] R. P. Adams and D. J. MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[2] R. Balakrishna, C. Antoniou, M. Ben-Akiva, H. Koutsopoulos, and Y. Wen. Calibration of Microscopic Traffic Simulation Models: Methods and Application. *Transportation Research Record: Journal of the Transportation Research Board*, 2007.

[3] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. SUMO - Simulation of Urban MObility - an Overview. In *The Third International Conference on Advances in System Simulation (SIMUL)*, 2011.

[4] M. Boyadjis. Emerging Technologies: Big Data in the Connected Car. Report, November 2013.

[5] L. Chu, H. Liu, J.-S. Oh, and W. Recker. A calibration procedure for microscopic traffic simulation. In *IEEE Intelligent Transportation Systems*, volume 2, pages 1574–1579, 2003.

[6] P. Fearnhead. Exact and efficient bayesian inference for multiple changepoint problems. *Statistics and computing*, 16(2):203–213, 2006.

[7] M. Fellendorf and P. Vortisch. Microscopic traffic flow simulator VISSIM. In *Fundamentals of Traffic Simulation*, number 145 in International Series in Operations Research & Management Science, pages 63–93. Springer New York, 2010.

[8] A. Halati, H. Lieu, and S. Walker. CORSIM – Corridor traffic simulation model. In *Traffic Congestion and Traffic Safety in the 21st Century: Challenges, Innovations, and Opportunities*, pages 570–576, New York, 1997.

[9] P. Hidas. Modelling lane changing and merging in microscopic traffic simulation. *Transportation Research Part C: Emerging Technologies*, 10(5–6):351–371, Oct. 2002.

[10] M. Hofman. On the Complexity of Parameter Calibration in Simulation Models. *JDMS*, 2(4):217–226, October 2005.

[11] J. Hourdakis, P. Michalopoulos, and J. Kottommannil. Practical procedure for calibrating microscopic traffic simulation models. *Transportation Research Record: Journal of the Transportation Research Board*, 1852:130–139, Jan. 2003.

[12] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*. Springer, 4th edition, 2013.

[13] P. Janssen and P. Heuberger. Calibration of process-oriented models. *Ecological Modelling*, 83(1):55–66, 1995. Modelling Water, Carbon and Nutrient Cycles in Forests.

[14] T. Koslowski. Forget the Internet of Things: Here Comes the 'Internet of Cars'. Article, Wired Magazine, April 2013.

[15] F. Laquai, C. Gusmini, M. Tonnis, G. Rigoll, and G. Klinker. A multi lane car following model for cooperative adas. In *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, pages 1579–1586, Oct 2013.

[16] H. Liu, Q. Yu, W. Ding, D. Ni, H. Wang, and S. Shannon. Feasibility study for automatic calibration of transportation simulation models. In *Proceedings of the 44th Annual Simulation Symposium*, ANSS '11, pages 87–94, 2011.

[17] M. Lützenberger and S. Albayrak. Current frontiers in reproducing human driver behavior. In *Proceedings of the 46th Summer Computer Simulation Conference, Monterey, CA, USA*, pages 514–521, 2014.

[18] M. Masin, L. Limonad, A. Sela, D. Boaz, L. Greenberg, N. Mashkif, and R. Rinat. Pluggable analysis viewpoints for design space exploration. In *CSER*, pages 226–235, 2013.

[19] J. A. Michon. A critical view of driver behavior models: what do we know, what should we do? In *Human behavior and traffic safety*, pages 485–524. Springer, 1985.

[20] J. Pan, M. Khan, I. Popa, K. Zeitouni, and C. Borcea. Proactive Vehicle Re-routing Strategies for Congestion Avoidance. In *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 265–272, May 2012.

[21] B. Park, J. Won, and I. Yun. Application of microscopic simulation model calibration and validation procedure: Case study of coordinated actuated signal system. *Transportation Research Record: Journal of the Transportation Research Board*, 1978:113–122, Jan. 2006.

[22] M. Rondinone et al. iTETRIS: A modular simulation platform for the large scale evaluation of cooperative ITS applications. *Simulation Modelling Practice and Theory*, 34:99–125, May 2013.

[23] Z. Shen, K. Wang, and F. Zhu. Agent-based traffic simulation and traffic signal timing optimization with GPU. In *14th International IEEE Conference on Intelligent Transportation Systems*, 2011.

[24] K. Smilowitz, C. Daganzo, M. Cassidy, and R. Bertini. Some Observations Of Highway Traffic In Long Queues. Research Report UCB-ITS-RR-98-6, Institute of Transportation Studies University of California at Berkeley, 1998.