Evaluation of Drivers Interaction with Assistant Systems Using Criticality Driven Guided Simulation

Stefan Puch[1], Bertram Wortelen[2], Martin Fränzle[1], and Thomas Peikenkamp[2]

[1] Carl von Ossietzky University, 26129 Oldenburg, Germany
{stefan.puch,martin.fraenzle}@informatik.uni-oldenburg.de
[2] OFFIS - Institute for Information Technology, 26121 Oldenburg, Germany
{bertram.wortelen,thomas.peikenkamp}@offis.de

Abstract. Advanced Driver Assistance Systems (ADAS) operate more and more autonomously and take over essential parts of the driving task e.g. keeping safe distance or detecting hazards. Thereby they change the structure of the driver's task and thus induce a change in driver's behavior. Nevertheless it is still the driver who is ultimately responsible for the safe operation of the vehicle. Therefore it is necessary to ensure that the behavioral changes neither reduce the controllability of the vehicle nor the controllability of the hazardous events. We introduce the Threshold Uncertainty Tree Search (TUTS) algorithm as a simulation based approach to explore rare but critical driver behavior in interaction with an assistance system. We present first results obtained with a validated driver model in a simple driving scenario.

Keywords: Guided Co-Simulation, Driver Model, Hybrid Simulation, Risk Analysis, Monte Carlo.

1 Introduction

ADAS have a strong impact on the behavior of drivers and the controllability of vehicles and thus it needs to be demonstrated that the use of an ADAS does not reduce the controllability. The European Code of Practice for the Design and Evaluation of ADAS [4] proposes in more details, how to address these issues during the system development process. It recommends empirical experiments with human drivers as primary instrument for the evaluation process. Unfortunately these experiments are often time consuming and costly. Another option, which is proposed in this paper, is to use a model based approach which integrates executable models of environment, driver and ADAS into a co-simulation.

Although the driver model we use is only an abstraction and does not cover all aspects of human behavior, it still provides some advantages for speeding up the evaluation process. First of all, replacing the human driver by a driver model enables a fully automated simulation and thus a huge amount of different situations can be evaluated in a short amount of time. Such an approach can be useful during an early evaluation phase to identify scenarios which may deserve

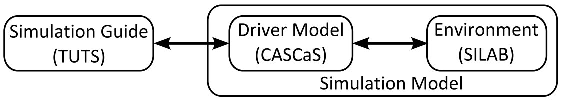


Fig. 1. Setup for the guided simulation as used for the present work

a more detailed analysis by doing experiments with human subjects. Another important issue addresses the risk assessment of the ADAS. Since risky situations are typically very infrequent and the number of tests in empirical experiments is limited, rare events are normally not observed. This issue can be addressed by doing large numbers of simulation as it increases the probability of observing rare events. But performing pure Monte Carlo simulations is not very efficient for this matter, because the largest amount of simulation samples show typically non-critical and mostly similar behavior. As risk assessment is not interested in the average behavior, we present in this paper a heuristic algorithm that guides the simulated driver behavior into critical situations. We furthermore present a first evaluation study of our guided simulation approach. It utilizes a driving simulator study that was conducted with 17 human drivers. Data of this experiment were used to create a cognitive driver model. Human participant and driver model have to interact with an in-vehicle system while driving. We used the TUTS algorithm to guide the driver model behavior into critical situations, where it may hits the pillar of a bridge.

2 Simulation Infrastructure

We propose to use closed-loop simulations of the driver-vehicle-environment system to support the analysis of newly introduced in-vehicle systems. A high-level view on the connection of the driver model, the driving environment and the simulation guide based on the TUTS algorithm is shown in Figure 1. The guide is application independent and is thus not aware of what kind of simulation model it is guiding. In the setup used in the present work, it only guides the behavior of the driver model and not the environment. The driver model, which we employ in our experiments is created using the Cognitive Architecture for Safety Critical Task Simulation (CASCaS). Cognitive architectures are a means to simulate human behavior in a psychologically plausible way. The objective of CASCaS is to supplement the development process of new assistance systems with the possibility of a virtual-human-in-the loop simulation in early design phases. To account for the variations in human behavior, a lot of processes involved in CASCaS contain probabilistic elements (PEs). The TUTS algorithm adjusts the PEs in order to simulate rare behavior. We will give a brief overview on CASCaS and some of its most important probabilistic elements. For more detailed information see [6].

CASCaS is a hybrid architecture consisting of a set of components which are related to different aspects of the human behavior. Like most other CAs, CASCaS simulates goal oriented human behavior. Multitasking is achieved by switching between a set of task goals. A scheduling algorithm dynamically selects in a probabilistic manner from a set of weighted goals the current goal to be executed. The main knowledge processing unit of CASCaS is based on a rule engine, which probabilistically selects from a set of weighted rules in a similar manner as the goal selection algorithm. The rules describe how the cognitive model solves its tasks and achieves its goals. Beyond the selection between finitely many discrete choices, CASCaS also makes use of continuous probability distributions. For a realistic timing of different processes like hand and eye movements or fixation durations, CASCaS adds noise to the calculated durations. The noise is drawn from continuous probability distributions.

3 Threshold Uncertainty Tree Search

In safety-critical environments critical situations are often associated with extreme behavior, which shows up very rarely. Exploring the model behaviour for these rare situations using a pure MC approach is very inefficient. Hence Puch, et al. [3] introduced a concept of an algorithm, that probabilistically guides the simulation into critical situations. We applied this algorithm to the driver model use case presented in section 4. Unfortunately the search speed was very unsatisfying. We elaborated on this aspect and present now as a result the Threshold Uncertainty Tree Search (TUTS) algorithm. TUTS is Monte Carlo-based, meaning that it repeatedly simulates a scenario starting from some initial state, while taking different probabilistic choices. Let S be the set of reachable states and $S_0, S_t \subseteq S$ sets of initial states respectively terminal states which all satisfy a user defined initial condition respectively termination condition. Each simulation run starts in an initial state and stops once a terminal state is reached.

The simulation model (in this case the CASCaS driver model) has to make several choices for all probabilistic elements, that occur during simulation, e.g. the probabilistic selection of a goal. PEs can be divided into two classes: discrete probabilistic elements (DPE) and continuous probabilistic elements (CPE). A DPE e_d is associated with a categorical distribution for a finite set of options denoted by O_{e_d}. A CPE e_c is associated with a continuous probability density function (PDF) denoted by f_{e_c}. We initially assume, that the simulation model only contains DPEs and no CPE. Given an initial state $s \in S_0$ the model behavior b can be characterized by the sequence of probabilistic events $b = o_1, o_2, \ldots, o_{n_b}$, with o_i being the option taken at the i-th event and n_b being the number of events that occur until a terminating state is reached, which might vary for different behaviors. By starting in the same initial state and taking the same sequence of options in different simulation runs, we obtain deterministic behaviors. We call a guidance algorithm with this property a deterministic guidance.

The left part of Figure 2 illustrates an event tree, that was derived by taking successively different available options during multiple simulation runs. Because CASCaS contains many different processes with probabilistic elements, each behavior typically contains a great number of options. This results in very large event trees, because the tree size increases exponentially with its depth.

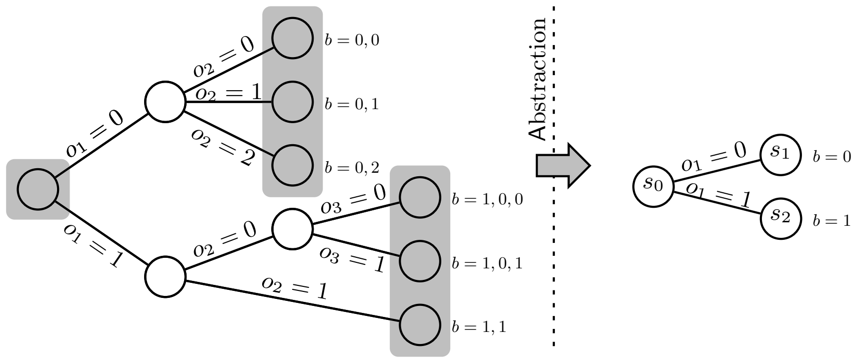


Fig. 2. Only considering events at the gray nodes strongly reduces the event tree

The size of the event tree can be reduced drastically by only considering a subset of important PEs. Let us consider that we are only interested in the events of some PEs that occur in the gray-marked states in Figure 2. The guide will not consider the events of other PEs, which leads to a smaller graph shown in the right part of the figure. This however introduces the problem, that the same sequence of options on two different simulation runs results in the same path of the event tree although the model behavior might differ. Thus the guide is no longer deterministic. If the ignored PEs contribute only marginally to the behavior of the agent, we expect that the observed behaviors in one node are at least similar. Thus a node of the event tree represents a class of similar driving behavior. In section 4 we analyze the similarity of driving behavior within nodes.

In order to deal with CPEs a further source of variance is introduced. Each CPE e_c is discretized by splitting f_{e_c} into a set of q PDFs with equal probability density. The guide selects one of these PDFs and draws a random value, which introduced additional variance. See [3] for details.

In order to drive the model into interesting situations, the TUTS algorithms requires a user defined function, which returns a numerical measure of the criticality $\tilde{c}$ for each simulation run, e.g., the minimum time-to-crash value observed during the simulation run. A user defined threshold τ is used to define the level that separates the acceptable and unacceptable critical situations. The TUTS algorithms attempts to guide the simulation into a region close to the threshold of acceptable and unacceptable situations. Let $C(s)$ be the set of criticality values, that have been observed in all simulation runs, which passed the node s. Consider the event tree on the right side in Figure 2. Being in node s_0 the guide should choose an option ($o_1 = 0$ or 1), that most likely results in a criticality close to τ. We measure the closeness to τ in standard deviations according to the distribution of $C(s)$, by calculating the z-score of τ:

$$z(s) = \frac{\tau - \mu(C(s))}{\sigma(C(s))}$$

To increase the likelihood of observing a criticality of τ when passing s, the guide should prefer options which lead to small absolute z-scores. This is done in a probabilistic way by weighting all options, that the guide can select. A weight $w(s)$ is defined for each node s. With S'_s the set of child nodes of s are denoted. These are the nodes that are reached when selecting an option in state s. The function $t : S, O \longrightarrow S$ defines the parent-child relationship. $t(s, o)$ gives the node, that is reached when option o is selected while in node s. The guide uses the weights of the nodes to probabilistically select from the current set of options $\tilde{O}$. The probability of selecting option $o \in \tilde{O}$ if the current node is s:

$$\mathrm{P}(o) = \frac{w(t(s,o))}{\sum\limits_{p \in \tilde{O}} w(t(s,p))} \tag{1}$$

This means that options that leads to highly weighted states are selected with higher probability. Therefore the states with low z-values should have high weights. The weights are defined by:

$$w(s) = \frac{1}{(z+1)^{f(s_p)}} \tag{2}$$

Unless s has been visited twice, $\sigma(C(s))$ does not exist and z is undefined. Therefore, if any selectable child node has not yet been visited twice, the guide selects one of these randomly. In this way the guide explores each branch at least two times. The $f(s_p)$ exponent is used to adjust the weights the more confidence is gained about the distribution of criticality values in $C(s)$. Let s_p be the parent node of s. Thus each sibling uses the same exponent.

Especially for nodes at the top of the event tree the variance of criticality values $\sigma(C(s))$ is high and sibling nodes often have similar mean values $\mu(C(s))$. These nodes are at the beginning of the simulations. Many subsequent decisions influence the criticality of a simulation. This results in high variances and in uncertain mean values for early nodes. In order to take the uncertainty about the z-values into account the function f is used. This function should rise with the certainty of the z-values. This leads to a spreading of weights, the more confident the z-values are. For our use case scenario we used a simple definition of f that creates identical values for the set $S_{\tilde{O}} = \{t(s,o) | o \in \tilde{O}\}$ of all sibling nodes, that can be reached with the current set of options $\tilde{O}$. Let $n_{min} = \min(n_{s'} | s' \in S_{\tilde{O}})$ be the minimum number of visits of any node in $S_{\tilde{O}}$, then we used the following definition for f, with free parameters a and b to adjust the search speed:

$$f(s) = a + b \cdot n_{min}, \quad \text{with } a = 0.5, \quad b = 0.5 \tag{3}$$

4 Evaluation

In order to evaluate the TUTS algorithm we utilize the setup of a recent driving simulator study [6,7]. Aim of this study was to investigate drivers attention distribution when driving on a curvy road while interacting with a secondary in-vehicle task. We briefly describe aspects of the study that are relevant for the present work. For more details see [6,7].

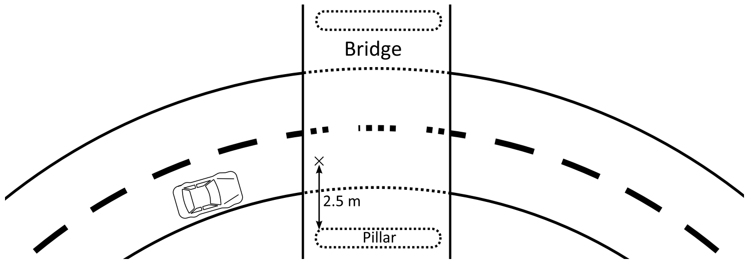


Fig. 3. Distance of bridge pillar to the center of the lane in the driving scenario

4.1 Scenario

The driving scenario consists of a winding road with curve radii between 375 m and 750 m. The drivers were instructed to focus on three goals: (1) Keep the car in the middle of the lane. (2) Keep a constant speed of 100 km/h as closely as possible. (3) Solve some tasks displayed on an in-vehicle display in varying time intervals as soon as possible. This third task is used representatively for the interaction with an infotainment or assistance system displayed in the center console. To meet the requirements of all goals the attention has to be switched between this three tasks and their respective areas of interest (road, speedometer, in-vehicle display).The structure of this scenario is simple. There is no complex task for the driver, no surrounding traffic, road signs or intersections. The most crucial aspect is how drivers distribute their attention among the three goals. If too few attention is paid to keeping the car within the lane, the driver might cross the lane border. Due to this simple structure, we used this scenario as a first use case for the TUTS algorithms, to identify driver model behaviors, where the interaction with the in-vehicle task leads to critical situations.

To demonstrate the functioning of our algorithm, we introduced a single critical point in the scenario. This was done by placing a bridge over the road, which was not present in the original scenario (see Figure 3). The minimum distance from the center of the car to the right bridge pillar was used as a measure for the consequence $\tilde{c}$ of the driving behavior. The pillar of the bridge is placed 2.5 m away from the center of the left lane.

4.2 Simulation

Wortelen, et al. [6,7] developed a driver model using CASCaS, which was able to drive this scenario. The driving behavior and the visual scanning behavior of the driver model has been validated against the behavior of 17 human drivers.

The simulation of the driver model is divided in two phases. In the beginning of each simulation run the driver model accelerates to 100 km/h. Reaching 100 km/h defines the initial states S_0. A simulation run ends 20 m after the car passed the pillar or if a predefined time limit is reached. This defines the terminal states S_t. The time span between initial and terminal state is typically around

8 seconds with small variations. The smallest distance to the pillar is reached approximately 7 seconds after the initial state.

As mentioned CASCaS contains several PEs. It can be configured which classes of PEs the simulation guide considers and which not. The most critical aspect in this scenario is the way in which the driver model switches attention between the three tasks [7]. We therefore assume that the probabilistic element in the selection process for task goals is most relevant for the guiding algorithm. The way in which the driver model executes the tasks is in great parts described by a set of rules, which are selected and executed in a probabilistic manner similar to the goal selection process.

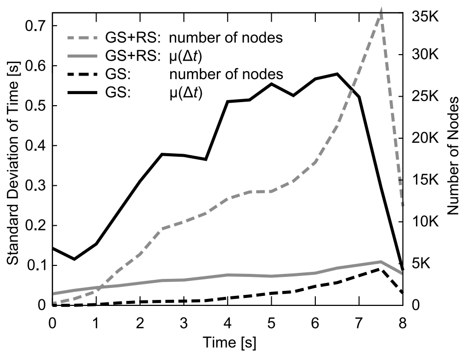
In order to compare the results of our guided simulation we performed three sets of simulations of this scenario. First we did 10,000 simulation runs in a Monte Carlo way without any guidance (MC). Then we used the simulation guide and did 10,000 runs considering only the goal selection DPEs (GS) and 10,000 runs while additionally considering rule selection DPEs (GS+RS). All other PEs are drawn randomly according to their distribution.

4.3 Results

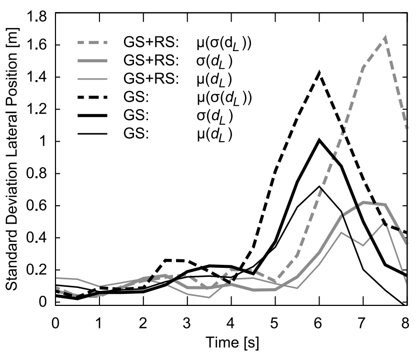
In the algorithm description we assumed that each node of the event tree represents similar behaviors. The PEs not considered by the simulation guide introduce behavioral variance within each node. The variance should be smaller the more PEs the guide considers. But at the same time this increases the size of the event tree. This effect is illustrated in Figures 4. The figures should be read as follows. Every time a node is entered during a simulation the current time distance (Δt) to the point in time when the initial state was reached, and the current lateral position d_L is recorded. The lateral position is measured in meter and is 0 m, if the car is in the center of the lane and increases, when the car drifts to the right lane border and decreases if it drifts to the left lane border. At the end of each 10,000 simulations we calculated for each node the mean values for Δt and d_L and their standard deviation. In the following we only consider nodes that have been visited at least twice and thus have a standard deviation. The x-axis is discretized in 500 ms steps. All nodes with a mean time within each 500 ms window are aggregated.

The solid lines in Figure 4(a) show the mean standard deviation for Δt of all nodes in a 500 ms window. It can indeed be seen that the standard deviation for the GS+RS simulation are much smaller than for the GS simulation, showing that the range of behaviors represented by a node is narrower when goal and rule selection are considered. But at the same time the size of the tree growths much faster in the GS+RS configuration. This can be seen by the dashed lines, which show the number of nodes which have been aggregated in each window.

Data for the lateral position has been aggregated in the same way. In Figure 4(b) the mean standard deviation for d_L within each time window is shown by the thick solid lines. The dashed lines show the total standard deviation, which is calculated over all observed d_L values from all nodes within each time window. If the assumption, that taking the same paths in the event tree leads



(a) Number of nodes and variance in time



(b) Lateral position

Fig. 4. $\mu(x)$ is the mean over all observations of all nodes with a time interval an $\sigma(x)$ is the standard deviation. $\mu(\sigma(x))$ calculates the standard deviation within a node and averages it over all nodes within a time interval. All values are aggregated over all nodes within a time interval of 0.5 s.

to similar behavior holds, then the standard deviation in each node should be much lower than the total standard deviation. For the GS configuration it can be seen that especially for the last four seconds the mean standard deviation is indeed smaller than the total deviation. However the difference is not very big. By also considering the rule selection DPEs in the GS+RS configuration this difference can be increased. It can be seen that for both configurations during the first 4 seconds the standard deviation does not change much. The same holds for the mean lateral position as indicated by the thin solid lines. Afterwards the mean lateral position of the car in the GS and in the GS+RS configurations drifts towards the bridge pillar. This happens earlier in the GS configuration, where the highest lateral deviation is reached approximately at $\Delta t = 6$ seconds, while the pillar is reached at around $\Delta t = 7$ seconds. In contrast the GS+RS configuration reaches the highest lateral deviation at that time. In future work we will investigate on these differences. A possible reason is, that the guide in the GS configuration controls less aspects of the driver model and is thus not able to guide it as precisely.

The main objective is to explore the driver model behavior and to simulate valid behaviors that show rare and critical consequences far more often than pure Monte Carlo simulations would. In Figure 5 the results for all three configurations are compared. Shown are the frequency distributions of the consequence values $\tilde{c}$ discretized in 0.1 m steps each for 10,000 simulation runs. The distribution for the Monte Carlo simulation is very narrow. In fact 7,272 of the 10,000 runs show a consequence value of around 2.4 m. Nearly all deviations are within the lane boundaries, even though the driver model is interacting with the secondary task.

This is different for the GS and GS+RS configurations. The distributions are strongly biased toward low $\tilde{c}$ values. Especially the GS+RS configuration shows a high number of simulations close to $\tilde{c} = 0$. For the guided simulation

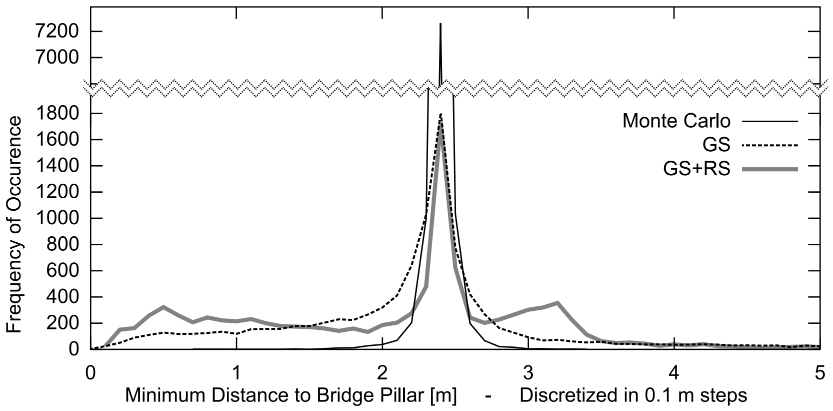


Fig. 5. Frequency distributions of criticality values for all three configurations

configurations also high $\tilde{c}$ values are observed more often. The reason for this is that the guide has to explore the behavior of the model, which might also lead to higher $\tilde{c}$ values. Another reason is, that many options selected by the guide destabilize the lateral control of the driver model, which not only leads to drifts to the right side, but also to the left side.

5 Related Work

A similar approach to ours is found in Hu's work [1]. He presents an efficient model-based simulation engine for risk assessment of complex systems consisting of software, hardware and human elements. Furthermore, he developed a guided simulation process to avoid the slow coverage of Monte Carlo methods. The difference to our approach is that the engineering knowledge about the system is used prior to the simulation to generate a plan as a high level guide. The plan itself contains a list with scenarios of interests and is used as a map for exploration during simulation. Our approach is similar, but does not need a predefined event tree, which is used by Hu. Our tree structure is automatically derived during the simulation. The level of detail of the event tree can be adjusted, ranging from a small and abstract tree up to a fully deterministic one.

The basic idea of the TUTS algorithm was introduced on a conceptual level by Puch, et al.[3]. After first tests with the initial implementation we observed a slow search speed and elaborated on this aspect. The resulting algorithm presented in section 3 is similar to some game theoretic approaches. It shares important ideas of the UCT algorithm (Upper Confidence Bounds applied to Trees) [2], which is a game theoretic planning approach based on rollout Monte Carlo search techniques [5]. Like the UTC algorithm TUTS implements a weighted search within the event tree, that adjusts the search weights according to the confidence of the results achieved during the previous simulations. Though the search criterion of TUTS is different in its nature. This will be described in section 3.

6 Discussion

In this paper we have presented results from a first study to efficiently explore and simulate rare and critical model behavior in a cosimulation of environment, driver and ADAS. We therefore demonstrated the TUTS algorithm and pointed out promising achievements. However the use case was very simple and in future work we aim at more complex scenarios and driver tasks. Currently a driver model is under development which is intended to simulate a number of different highway scenarios like car following, overtaking and merging into traffic flows. An advanced highway assistance system is introduced into the simulation, instead of using an artificial in-vehicle task. Data from simulator experiments with human drivers already have been performed and will be used to validate the driver model. Furthermore we want to provide a concept for the analysis of the resulting event tree. The most relevant paths in the tree should be identified and presented to the system developer in order to reveal critical interaction sequences.

Acknowledgement. This research has been performed with support from the Ministry of Science and Culture of Lower Saxony within the project IMoST II.

References

1. Hu, Y.: A Guided Simulation Methodology for Dynamic Probabilistic Risk Assessment of Complex Systems. Ph.D. thesis, University of Maryland (2005)
2. Kocsis, L., Szepesvári, C.: Bandit Based Monte-Carlo Planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
3. Puch, S., Wortelen, B., Fränzle, M., Peikenkamp, T.: Using Guided Simulation to Improve a Model-Based Design Process of Complex Human Machine Systems. In: Klumpp, M. (ed.) Proceedings of the 2012 European Simulation and Modelling Conference, pp. 159–164. EUROSIS-ETI (2012)
4. Servel, A., Knapp, A., Jung, C., Donner, E., Dilger, E., Tango, F., Mihm, J., Schwarz, J., Wood, K., Ojeda, L., Neumann, M., Brockmann, M., Meyer, M., Kiss, M., Flament, M., Kompfner, P., Walz, R., Cotter, S., Winkle, T., Janssen, W.: Code of Practice for the Design and Evaluation of ADAS (2006), `http://www.prevent-ip.org`
5. Tesauro, G., Galperin, G.R.: On-line Policy Improvement using Monte-Carlo Search. In: Proceeding of Advances in Neural Information Processing Systems 9 (NIPS), Denver, CO, USA, December 2-5 (1996)
6. Wortelen, B., Baumann, M., Lüdtke, A.: Dynamic Simulation and Prediction of Drivers' Attention Distribution. Transportation Research Part F: Traffic Psychology and Behaviour (submitted)
7. Wortelen, B., Lüdtke, A., Baumann, M.: Integrated Simulation of Attention Distribution and Driving Behavior. In: Proceedings of the 22nd Annual Conference on Behavior Representation in Modeling and Simulation (in press)