3

Driver Modelling

Jens Klimke and Lutz Eckstein

Institute for Automotive Engineering, RWTH Aachen University,
Steinbachstraße 7, 52074 Aachen, Germany

3.1 Introduction

Traffic simulations become more and more relevant for the development of Advanced Driver Assistant Systems (ADAS) and algorithms for automated driving. They are used to evaluate the functions concerning important impact factors like safety, efficiency, mobility or costs. Therefore, the system is tested and evaluated as a component of the virtual vehicle in simulations. The factors manageability and acceptance of the users regarding the tested system are prospected and evaluated in driving simulators, where the real driver can be part of the virtual environment. Both, in traffic simulations and in simulators, the realistic behaviour of the surrounding virtual road users to the equipped vehicle is an important requirement for a suitable evaluation of the system because this behaviour influences the reaction of ADAS and driver significantly. Moreover, it is necessary, that the behaviour of the traffic can be adjusted systematically in order to generate defined traffic situations of relevant constellations and in different nuances of criticality. As in real traffic, small changes in the initial conditions can produce a large difference in the result. This phenomenon can only be reproduced in a simulation if the driving behaviour patterns reflect the human driver behaviour closely.

The basis of this *driver model* and its possible functionality or ability is the underlying simulation environment. To determine the risk of congestion for example, a traffic simulation environment with *macroscopic*, e.g., fluid dynamic based traffic behaviour, is suitable. The easiest macroscopic representation of virtual traffic could be an equation with the result of an average velocity dependent on the density of traffic. This might be a complex mathematical relation producing suitable results for some purposes but it is impossible to understand the specific inner traffic effects like congestion waves and traffic collapses. For such effects, the influences of the traffic elements on the driver models have to be understood.

These are basically the interactions between the driver-vehicle-units among each other and the reactions of the units to the traffic environment like traffic light systems or the road curvature. In this kind of traffic simulation, called *microscopic* traffic simulation, the desired controlling reaction of the driver or the automated function is calculated and implemented directly into the vehicle. This is done in form of a change of the dynamic state of the vehicle, e.g., a desired acceleration, which consequently results a change of velocity and position. The driver and the vehicle represent an inseparable unit, but entirely with a unit-specific behaviour. The behaviour might respect some dynamic restrictions of the vehicle and in some cases of the driver, but does not depict the driver-vehicle-interaction.

For the analysis of modern ADAS this kind of simulation is not suitable, as a driver has, e.g., to be able to override the system by using the control elements, like pedals, steering wheel or switches. An ACC for example can be switched off in critical situation by using the brake pedal or can be overridden by using the accelerator pedal to further increase or keep the acceleration. These effects can only be simulated if vehicle and driver are implemented as separate models and if the interfaces between driver model and vehicle model are used to implement the driver's wish to the vehicle. Thus, this concept can be called *sub-microscopic* or *nanoscopic*.

Another specific application for sub-microscopic traffic simulations is the exploration of detailed effects related to the vehicle, like fuel consumption analysis in specific traffic situations or environments. Within these analyses a very detailed vehicle model is needed. But it is not only the specific application which let us chose a higher level of traffic simulation. Obviously, the higher the level of detail, the more effects can be depicted with a single traffic simulation environment and model set-up but at the expense of computing time up to the loss of the real-time capability. Additionally, the effort of setting up the models increases due to the increase of model parameters. For the same reason the validation of the models is much more complex, too.

In the past decades many driver models where developed with special focuses on different specific elements of the driving task. Some try to show an optimal behaviour, without taking into account the physical and cognitive abilities and limitations of the human driver. Others focus on these restrictions or on the information process in the driver's brain and body and the capability of the driver to process different information in parallel. In literature many categories of driver models are published. Jürgensohn defines in [1] two basic categories of driver models, formal and non-formal models. *Formal* models have a fixed description but a changeable inner value. The result of formal models is reproducible, that means, the same conditions lead to the same output. *Non-formal* models are not described by those fixed dependencies (like equations or lingual definition) or they have a non-changeable (constant) character. Examples of formal models are *descriptive* models, which have a fixed description but have a character which is not defined by an input-output structure. In the European research project ASPECSS [2] and in Deliverable D3.1.1 [3] of the DESERVE project the definition is different. In these sources *descriptive* models are clearly defined (fixed, but not constant) and generate a numeric, quantitative output dependent on different numerical influences. This output is reproducible but can anyway contain stochastic elements. *Functional* models describe physical and psychological aspects of driving, like the information processes, the human structure of thinking and acting. They do not generate a numeric output but draw a picture of the elements of driving. The difference between functional and descriptive models in this definition is not unique and not complete; there are hybrid models and models which can't be matched to any of these categories. In this chapter, the distinction between *formal* and *functional* models is used to avoid the conflict of the two definitions of *descriptive* models.

In complex traffic simulations the usage of both kinds of models is needed to depict realistic traffic flow and driving behaviour. Formal models describe algorithms for a driver model how to reach its goal by setting defined reference values dependent on the input. Functional models can help to understand the driver's wishes and to create an eligible structure and decision algorithm.

In the DESERVE project, a rapid prototyping platform for the development of ADAS was created and a suitable tool-chain for the development process was outlined. The traffic simulation is an important tool in the development process of ADAS and thus is part of the DESERVE tool-chain. As described above, a realistic driver model is needed for the development and evaluation of modern ADAS. In the next sections, the way of modelling the driving behaviour is described, followed by the requirements for the DESERVE driver model. On the basis of the requirements the structure of a sophisticated driver model is developed and the used implementation techniques and strategies are explained. In the last section two different applications of the driver models are presented.

3.2 Driver Modelling

Driving is not just a single decision and a single action at once. It is rather a complex interoperation of different motivations, perceptions, decisions and states with continuous and discrete changes. To create a realistic driver model, a strict delimitation between these elements has to be done and it is helpful to create a suitable structure with a unique and logical naming of the elements and well-defined interfaces. To develop such a structure, driving has to be analysed on the basis of typical driving scenarios, manoeuvres and actions.

Besides the perception and the handling or action, the information processing is the most important part of driving. Within the information processing, the driver estimates desired values for different future vehicle states he wants to achieve, like a desired speed, a desired following distance, and distance to stop. These inner desired states are called driver-variables or briefly *variables*. Often a driver has multiple desired values for the same variable, generated by different motivations, between which a decision is needed. As an example the desired speed shall be used: The driver can have multiple causes of choosing a desired speed. For example the following three: First, to reach the destination as soon as possible. Second, the speed limits on the road. Third, the curvature of the road combined with the need for safety. For each motivation, a desired speed can be determined. The speed limit for the first mentioned motivation is the maximum speed the driver would choose on a free, straight road. If there are no further influences like other road-users or speed limits, the driver would travel with this speed. Situations, which do not allow travelling with this speed, do not imply that it is not the driver's wish (the driver wants to, but can't). For the second motivation, a speed in an interval around the speed limit, dependent on the law-abiding is desired. This can be higher or lower or exactly the speed limit. The third motivation results in a desired speed which allows the driver to pass a curve in a comfortable and safe manner.

All described motivations lead to different speeds, so the driver is in a dilemma: She/he has to decide for one speed to accelerate or decelerate to. The decision in this case is taken in a pragmatic way: The lowest speed wins, because on the one hand there is a comfort and safety limit, on the other hand there is a limit because the driver accepts the given speed limits or at least wants to avoid fees for driving too fast.

The described example shows two input types to the driving behaviour, the driver's character (here: need for safety, need for comfort and law-abiding) and the current situation described by the state of the own vehicle and other vehicles as well as the road and environmental structure. Moreover, not only the local situation influences driving. A good driver reacts before approaching to a discrete situation to reach the desired value in time. In the curve speed example above, a real driver would estimate the comfortable and safe speed based on the visual perception of the road's curvature before reaching the curve. On that perception, the driver decelerates with a rate which leads to the desired speed at the moment the curve is reached. Within the curve the driver corrects this estimation to satisfy the desired safety and comfort. The predictive behaviour is called *anticipatory* driving. The correction is called *compensatory* driving [4]. This phenomenon also has to be regarded in the development of driver models.

Of course the driver has more responsibilities than the decision of the desired speed. According to Rasmussen [5], the driving task can be seen in three levels: The strategic level where the driver plans and creates strategic values like a route, the manoeuvring level, where the driver processes the decisions and determines desired values and value sequences, the strategy can be implemented with. This behaviour is conscious: The driver knows exactly how to solve the driving task and creates a strategy. The driver is able to reflect decisions and actions he/she took in this level. In the control level the driver implements these conscious values into the vehicle by using the steering wheel, the accelerator and brake pedal and other control elements of the vehicle. This operation is not done in a single step. Often the driver determines a subconsciously desired value, like a desired acceleration, which is then transferred into the actual vehicle input. This value is not reflected by an experienced driver. It is an automatism by the driver to reach the conscious desired value. The desired speed shall be used for an illustration: After the decision to move freely, because no other road-user is influencing the driver, the desired speed is detected, which is a conscious value. To reach this speed, the driver accelerates with the desired acceleration, which is a subconscious value because the driver cannot quantify this value and it is not part of the strategy. The final implementation is done by using the vehicle's controls to reach this acceleration. The advantage of using this subconscious step is that the regarded values can be set, manipulated and limited dependent on realistic driver's needs independent of the conscious behaviour. Often the desired acceleration and yaw rate or curvature is used as an output of macroscopic driver models. In this definition these variables represent subconscious variables. Thus, without the implementation by using steering wheel and pedals, the model can be seen as a macroscopic driver model.

3.3 Requirements for DESERVE

Before creating a driver model, an analysis of the requirements for this model based on the field of application has to be done. In DESERVE, a rapid prototyping platform and development process has been created. The details of the platform can be found in Chapter 2. This requirements section concentrates on the applications of the DESERVE platform. In the first year of the project, the needs for the driver model were analysed in D3.1.1 [3]. There are two kinds of driver models identified in the project: the virtual driver for the usage in traffic simulations like described above and the driver intention and distraction model, which is used as a component of an ADAS to detect the real driver's state.

The literature review, the analysis of existing driver model concepts and in particular the research work in the DESERVE project shows that it is not possible to create one holistic driver model to satisfy all scientific needs. Nevertheless it would be very attractive, if there was one basic structure combining the ideas of the previous research, in which the algorithms can be added as independent modules. The connections of all modules – with properly defined affiliation and interfaces and in conjunction with a suitable parameter set – will produce the expected results. For that reason, a generic module-based structure needs to be developed which is well-defined and flexible for amendments. Most of the integrated algorithms can be used for several applications while others are specific to one. The generic structure should fit to all applications of driver modelling in an open way.

Another important issue is the implementation. Many driver models are implemented in native programming languages. This fact has a significant disadvantage: It becomes very muddled due to the one dimensional structure of programming code. Often driver model structures are shown in a two dimensional representation with levels in the up-down dimension and sequence of the information processing in the left-right direction (time related). An implementation of the driver model in an analogous structure could be very helpful to create a clear and well-arranged model. Thus, a graphical implementation would be aspired. Furthermore, it should be possible to structure or capsulate the content properly as well as the definition of the interfaces to take the advantage of modern programming techniques like object oriented programming or code reuse to avoid redundancy. Next to the structural requirements, the system shall be able to hold values or states over one or more time steps to implement the memory of the driver. Another requirement is the possibility to connect the driver model to the traffic simulation environment. This can be done by communication interfaces or by the native integration of the compiled driver model, for example as a dynamic linked library or similar techniques.

The driver model (virtual driver) in DESERVE shall be used in different traffic simulation environments for testing and evaluating ADAS functions in the process of the development. Within the project, the driver model shall be implemented and tested for a control function which is designed to show the advantages and benefits of the DESERVE platform. Therefore, an Advanced Cruise Control system (ACC) is combined with a Heading Control (HC). The system shall assist the driver on inter-urban road scenarios and increase the safety within the full speed range (WP 4.2, [6, 7]). The decision for demonstrating the system for the inter-urban area is made, because this area is a very important research field for the usage of ADAS functions of the next generation; especially those who reach the next level of driving automation (cf. SAE automation level 2 – partial automation, [8]). Also the evaluation of ADAS for the increase of safety is important in the inter-urban area. Therefore, detailed driver models are needed with the claim to be valid for the intended purpose. In particular, the modelling of realistic human behaviour on intersections and junctions is one of the most important developments for today's traffic simulations in order to develop ADAS with the goal to reduce the high number of accidents on intersections.

Analysing the application in DESERVE, the driver model requirements can be briefly defined:

- Inter-urban driving behaviour including safe-passing of slow, right-moving vehicles has to be implemented.
- The driver model needs the capability of route-following within multi-lane roads and complex but flexible transport networks.
- Full intersection and traffic light behaviour has to be implemented.
- Anticipatory driving behaviour, like early speed adaption needs to be reflected.
- Re-use of validated driving behaviour algorithms and driver model approaches is required.

The driver model is implemented and connected to the simulation environment PELOPS [9]. The inter-urban ACC and HC developed in DESERVE is tested in virtual traffic scenarios containing units controlled by the here described driver model. These scenarios include straight and curvy multi-lane roads, complex intersections with traffic lights and right-of-the-way controls by signs and structure, different speed limits, rare and dense traffic with different parameterisations and slow moving vehicles (e.g. mopeds). This testing set-up leads to a set of manoeuvres and primary driving tasks which have to be implemented:

Longitudinal	**Lateral**
Free moving	Lane keeping
Approaching/Following	Curve cutting
Braking in critical situations	

Figure 3.1 Primary driving tasks which are implemented in the driver model within the DESERVE project separated by longitudinal and lateral control.

Longitudinal	**Lateral**
Stopping, Standing and starting	
Turning on intersections	
Lane change	
Safe passing	

Figure 3.2 Manoeuvres which are implemented in the driver model within the DESERVE project.

There are several other manoeuvres which can be implemented like U-turning or stopping on the road side. These manoeuvres are not implemented within DESERVE. Nevertheless, the structure of the model shall offer the possibility to enhance the functionality.

3.4 Generic Structure

In this chapter the ika driver model is introduced. Within the DESERVE project, a suitable and generic driver model structure was developed and implemented which fulfils the requirements from the previous section. The interfaces and driver parameters are defined and described in this chapter.

3.4.1 Model Structure

From literature review, two generic structures can be identified: The three levels of driving by Rasmussen and the three blocks of perception, information processing and action, which can be found in several formal and non-formal model approaches (e.g. [10]). This leads to a matrix-form model shown in Figure 3.3. The modules (blue boxes) in the matrix represent model implementations or parts of those. The arrows, in different shades of grey, describe the information flow between the blocks and represent the internal

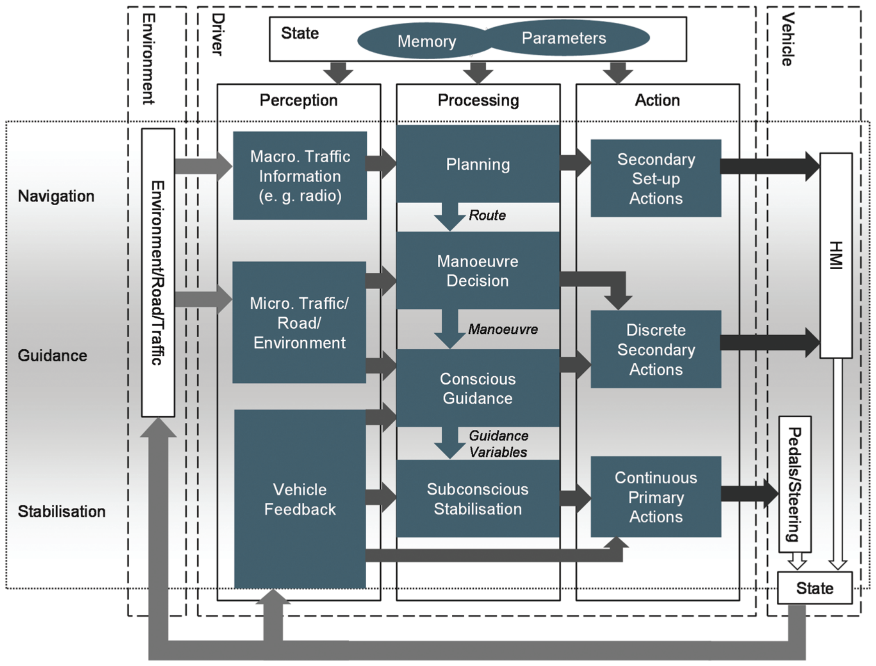

Figure 3.3 Driver model structure in the context of environment and vehicle: the structure includes perception, processing and action blocks including its functional modules and the regarded dynamic information flow.

interfaces. The blue arrows show the information flow through the three levels and represent the needed information (variables) for the driving tasks and manoeuvres. A central functional block of the model is the *State* block, where the driver-specific values are stored. The *Memory* module represents the driver's knowledge about the current situation, the manoeuvre states, the destination or route, etc. The memory is used to keep information for the following time steps, during the manoeuvre or for the whole simulation cycle. This information can be extrapolated to estimate current states of the ego-vehicle or other road-user even if the driver model does not sense the regarded information at the current time step. Thus, the memory has an interface to the *Perception* block and constitutes an input of this block besides the inputs of the environment and the vehicle. Current manoeuvre states and important values, which have to be known in the next time step, are also saved in the memory and are passed by the interface between the *State* and the *Processing* block where the driving calculation is implemented. The parameters represent the driver's character and are defined in two layers: qualitative and physical parameters (see Subsection 3.4.2). The *Parameters* block serves its values to all blocks of the driver model, for example by manipulating the handling time delay (reaction time). The *Action* block controls the handling or the conversion of the driver's wish into physical actions like the manipulation of the pedals, the steering wheel, shifting and using the HMI control elements.

As it can be seen in Figure 3.3, a strict assignment of all modules to a unique level is not possible. In the following the modules shall be explained in detail.

In the *Planning* module, route specific calculations are executed. In general, the units have a fixed route calculated or set in the initialisation of the simulation. In reality a driver changes the route under circumstances, e.g., traffic jams or road blocks. If such functionalities are needed, appropriate algorithms can be implemented in the Planning module. In the current implementation the route is stored in the memory. The Planning module calculates a value for each lane in the environment around the unit, which gives a quantitative value of how far the lane and its successors can be followed on the given route. Thus, the Manoeuvre Decision module can decide which lane the driver wants to take. The *Manoeuvre Decision* module processes all discrete manoeuvres and discrete decisions. That means on the one hand to decide for a manoeuvre and on the other hand to control the manoeuvre but not to calculate the related Guidance Values. The Decision module returns different states within the manoeuvre and process variables, which can be used by the following modules to perform the manoeuvre (in the figure briefly named *Manoeuvre*). An example is described in Section 3.5. Another output of the module is a set of *Discrete Secondary Actions* which are needed or desired at the beginning or during the manoeuvre. This can be for example switching the turning indicators in case of turning or lane changes. On the basis of the decision with its states and process values, a local strategy to perform these manoeuvres and continuous driving tasks is calculated in the *Conscious Guidance* module. Continuous driving tasks are performed during the whole simulation time without the need of a discrete decision. Of course, the output values of these tasks can be overridden by other results. An example is the motivation to keep the lane: This task is continuous because the driver always wants to stay in the lane but can be forced to leave the lane during an overtaking manoeuvre. Within the Conscious Guidance module the *Guidance Variables* are filled with values (guidance values), which the driver wants to reach. An example was given in Section 3.2 (desired speed during free moving). Several guidance values are calculated and passed to the *Subconscious Stabilisation* module. Within this module, desired stabilisation values are calculated.

In general, these values are the desired acceleration and the desired yaw rate for the longitudinal and lateral control respectively. Based on all motivations the stabilisation value with the highest benefit for the driver is taken. Besides the desired values, some real physical values, which are states of the vehicle, can be directly sensed by the driver. Thus, the driver is able to implement these values subconsciously by using the vehicle control elements (pedals and steering wheel). This implementation is done in the *Continuous Primary Actions* module.

To define the interfaces between the modules it is helpful to create a manoeuvre and driving task table. For the DESERVE implementation the following tables (Figure 3.4 and Figure 3.5) were developed, derived from Figure 3.1 and Figure 3.2.

In the motivation of *free moving*, the desired velocity of the driver is calculated. This velocity depends on the speed limit, the curvature of the road ahead and the maximum desired velocity of the driver. To reach the velocity, the driver model accelerates (subconsciously) dependent on the current velocity and the desired velocity. A suitable model approach is part of the Intelligent Driver Model (IDM) by Treiber, Hennecke and Helbing in [11]. An adaption of that approach for the usage in complex driving simulations is published in [12]. The *following* motivation is mainly influenced by a desired following distance which bases on a driver specific following time gap. To reach this distance the driver needs to accelerate or decelerate. The *lane-keeping* is performed by the usage of fix-points based on the Two-Point Visual Control Model published in [13]. This model can be adapted, so that the fix-points cause a yaw rate, which the driver wants to implement. The adaption is published in [14]. The yaw rate is chosen as the desired subconscious stabilisation value because it physically implies both, the curvature and the velocity. During standing, the driver model maintains a brake pedal value which results in a vehicle that does not move. This means that the pedal value is a subconscious value, different to the other longitudinal tasks.

Motivation	**Conscious**	**Subconscious**	**Action**
Free moving	Velocity	Acceleration	Pedal value
Following	Distance	Acceleration	Pedal value
Lane keeping	Fix-points	Yaw rate	Steering wh. angle
Standing	-	Pedal value	-

Figure 3.4 Process variables for the four basic driving motivations *free moving*, *following*, *lane keeping* and *standing*.

Manoeuvre	**Conscious**	**Subconscious**	**Action**
Lane change (Two-phase lc.)	1. Lat. offset 2. Fix points	1. Lat. velocity 2. Yaw rate	Steering wheel angle
Stopping	Stop point	Acceleration	Pedal value
Safe passing	Fix-points	Yaw rate	Steering wheel angle

Figure 3.5 Process variables for the three manoeuvres *lane change*, *stopping* and *Safe Passing*.

In Figure 3.5, the manoeuvre *turning* is missing. In this model turning is implemented in the decision module, at least to control the turning indicators, but does not require a process implementation due to the given features: A lateral and longitudinal turning manoeuvre can be seen as a 'normal' street following motivation if the turning path is known and a turning speed is calculated by the given curvature. In the case of conflicts with 'right of way' road-users (e.g. at left turns), the driver model stops with the manoeuvre *stopping*. If the conflict is resolved, the stop manoeuvre is aborted, so the driver model switches to *free moving* or *following*.

The perception is partly done in the simulation environment: All perceived information is transformed to the driver's coordinate system by the simulation environment. The driver model adapts the information with driver specific perception errors, like perception limits, continuous noise, sporadic disturbances or fluctuations and accuracy limits.

3.4.2 Parameter Structure

In many driver model approaches, *physical parameters* are used to influence the driver behaviour and generate heterogeneous or driver specific results like in the IDM [11]. Examples of physical parameters are the maximum comfortable acceleration and deceleration or a constant following time gap to the leading vehicle. These parameters are well measurable for a single driver or a group of drivers, represent a direct input to the model approaches and are mostly independent of each other. To describe the character of a driver, a big set of physical parameters has to be defined. In other driver models *humanised parameters* on a higher level are used which are not directly measurable. These parameters have a meaning which can be described as a characteristic or a constant attribute of a human driver. In general, the parameters are used to generate driver specific physical parameters, which are then dependent on each other by this humanised characteristic. With these parameters a characterisation of the driver is easier because the number of parameters is reduced to a smaller number. The challenge is to create a mathematical dependency which returns realistic results based on these fictive parameters. The humanised parameters used in the driver model for the DESERVE platform are named *sportiness*, *need for safety*, *law-abiding* and *estimation ability*. However, these parameters have no scientific physical or psychological meaning; they only represent groups of drivers and influence the underlying parameter block of physical parameters like *desired following time gap*, *acceleration profile* and many more. In Figure 3.6, the parameter concept of the DESERVE platform is shown: In the first block, the humanised parameters are shown. These parameters influence the physical parameters of the driver model. In this example, the *need for safety* parameter influences the *lower and upper following time gap* (see [15]) and the *acceleration profile* of the driver model. Parameters are not influenced by the dynamic inputs.

The set-up of a suitable parameter concept influencing all models in a realistic way is difficult and extremely dependent on the implemented model

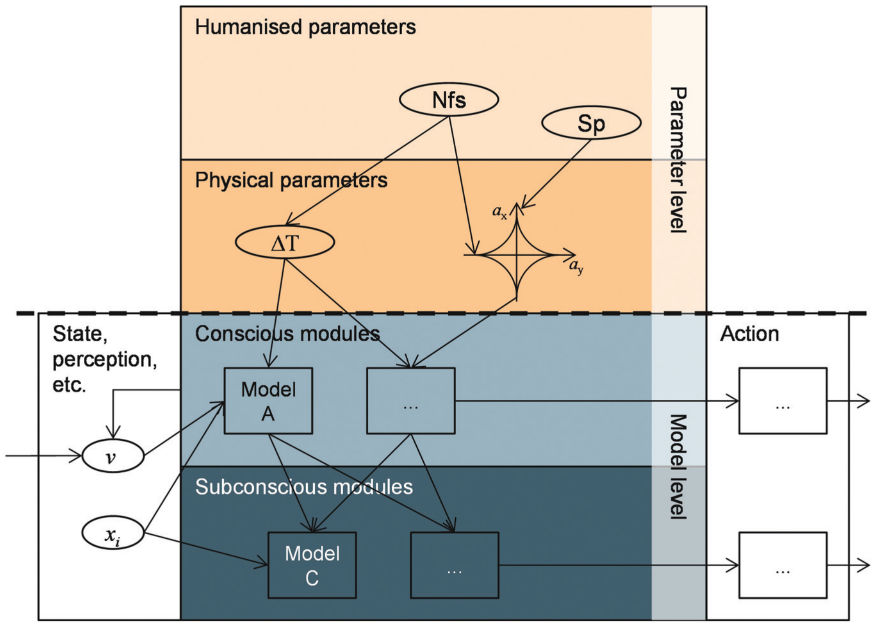

Figure 3.6 Sketch of the parameter blocks (brown) and model blocks (blue) of the driver model.

approaches. A concept to solve this problem could be to measure a large set of reference data and run an optimization to find the best fitting parameters. After that a validation has to be done with another set of data to prove the concept.

To create a traceable connection between the parameter blocks, in the DESERVE model, cubic polynomial functions are used. In a review of floating car data for example, the distribution of lower following time gaps of the Wiedemann model was generated. Basis of the distribution of these time gaps is a Gaussian distribution of the need for safety parameter with μ = 0.5 and σ = 0.15 as described in [15]. With the polynomial

$$\Delta T_{\mathrm{lower}}\left(p_{\mathrm{NFS}}\right) = 1.4 \cdot p_{\mathrm{NFS}}^{3} + 0.9 \cdot p_{\mathrm{NFS}}^{2} + 0.9 \cdot p_{\mathrm{NFS}} \tag{3.1}$$

with

$\Delta T_{\mathrm{lower}}$: Lower following time gap [s]

p_{NFS}: Need for safety; Gaussian distributed (0.5, 0.15) [–],

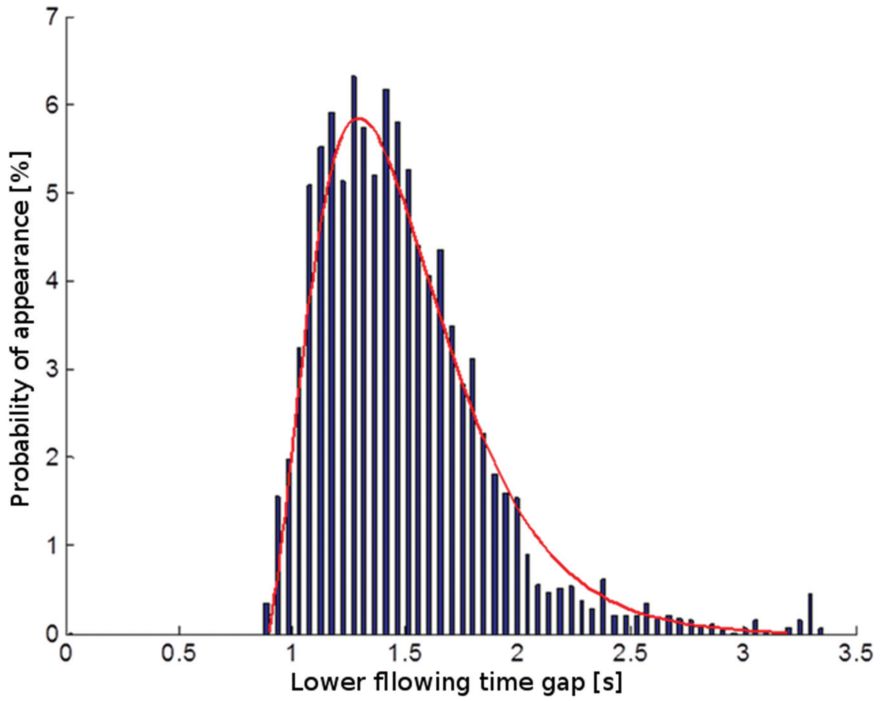

Figure 3.7 Distribution of lower following time gaps for real drivers (blue bars) and the modelled distribution dependent on a normal distributed *need for safety* parameter (red line).

the distribution of the lower following time gap returns a result shown as red curve in Figure 3.7. The blue bars show the floating car data which is the basis of the polynomial curve in this example.

This principle can be used and optimised analogously for the other physical parameters.

3.5 Implementation

The graphical programming tool *Matlab/Simulink* provides the implementation features described in the requirements in Section 3.3. The 2D graphical GUI allows a clear and well-arranged implementation close to the visual structure of the model. The implementation is easy to understand and easy to debug. In the university environment, many students and scientific assistance work with the driver model for a limited time range (e.g. Bachelor/master theses or PhD theses). Thus, a further important requirement is the comprehensibility of the model. Programming in Simulink is easy to learn also without deep knowledge of classic programming languages. The code can be capsulated in subsystems with defined inputs and outputs and several storage concepts can be used to implement the driver's memory. The data connection between the model and other tools can be established by using UDP or TCP/IP or other versatile techniques.

For the DESERVE example implementation, PELOPS is used as the simulation kernel with the support of environmental structures (road network, traffic lights, etc.) and vehicle models. The core of the new version of PELOPS is implemented in Java. The integration of a Simulink model is possible with the UPD communication interface. For the simulation of one vehicle this solution is suitable and is real-time capable in the current version of the ika driver model and PELOPS. If multiple vehicles use the same driver model instance with their specific inputs, at least time-dependent and memory-containing modules do not work properly. For the simulation of at least two vehicles, the Simulink-model needs to be duplicated to have an independent copy (second instance) of the driver model. This becomes difficult for a high or flexible number of vehicles in a simulation. Another problem is the high execution time due to the UDP connection and the Simulink model itself. A native execution combined with direct data exchange, e.g. by shared memory, is much faster. The Matlab/Simulink tool-chain brings the possibility of code generation: The desired model can be converted to C or C++ code which can be integrated in other C/C++ or FORTRAN code or can be compiled to a shared library in almost all computing platforms. In DESERVE this solution is used to integrate the driver model into PELOPS. For that purpose, a class wrapper is used around the generated code. That allows the simulation environment to create almost infinite numbers of independent driver model instances. Multiple test cases have been performed to show the capability of running traffic simulations with the full functionality of the driver models and a large number of traffic units in real time.

Except for the decision module, all modules are implemented in standard Simulink subsystems with mathematical blocks. The decision module is implemented in *Stateflow*, which is an integrated Simulink feature. Stateflow allows implementing state machines, which is a suitable implementation technique for discrete decision structures. To demonstrate a possible implementation of a manoeuvre decision the lane change shall be used as an example: In Figure 3.8, a state machine implementation is shown for a lane change decision including the *progress* and *sequence* control. The progress describes the state or the 'position' in the lane change like *initialisation (init)*, *origin lane*, *lane crossing (LC)*, *target lane* and *termination (term)*. The phases describe the phase control of the lane change by the driver. In this example the driver uses two phases to perform the lane change: In the first phase the driver accelerates laterally to a desired lateral velocity (anticipatory) dependent on the lateral offset. In the second phase, the driver 'switches' to the lane-keeping mode with the focus on the target lane (compensatory) by using the fix-point approach (see Figure 3.5). Dependent on the phase and the progress, the conscious guidance module, calculates the reference values which are needed to steer the vehicle to the desired lane. The transition A denotes the decision to perform the lane change, which is valid if there is a lane next to the ego driving path with

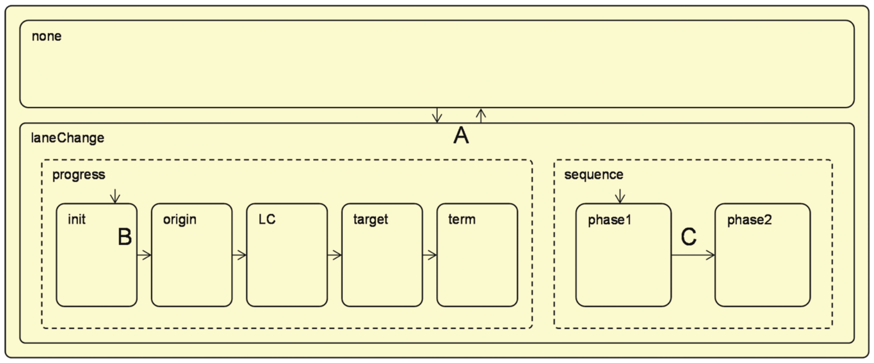

Figure 3.8 Stateflow model for a two-phase lane change including decision (A), progress control (B) and sequence control (C).

higher correlation to the route and some other conditions, like distance to the end of the lane, preference lane and a hysteresis. The basis for the decision is described in [16]. A decision for a lane change does not mean an immediate reaction. The driver model can decide before the lane or the desired gap is reached. In the case of a positive decision, the lane change is initialized. This is a continuous process as long as the active lane change is not started. The transitions B control the progress of the lane change and transition C represents the transition from the first phase to the second one in this example.

3.6 Applications in DESERVE and Results

Within the DESERVE project, the driver model was used for two different applications: The validation of left turn simulations within the full parameter range and the prediction of a real driver regarding the acceleration during free driving, approaching and following.

For the validation of left turn simulations (in this example without stopping), real traffic data from laser scanners were used to measure the trajectories of 136 vehicles on a junction in Alsdorf, close to Aachen in Germany. Figure 3.9 shows the results of the simulations for different parameter sets (coloured curves). The measured real-driver data are shown

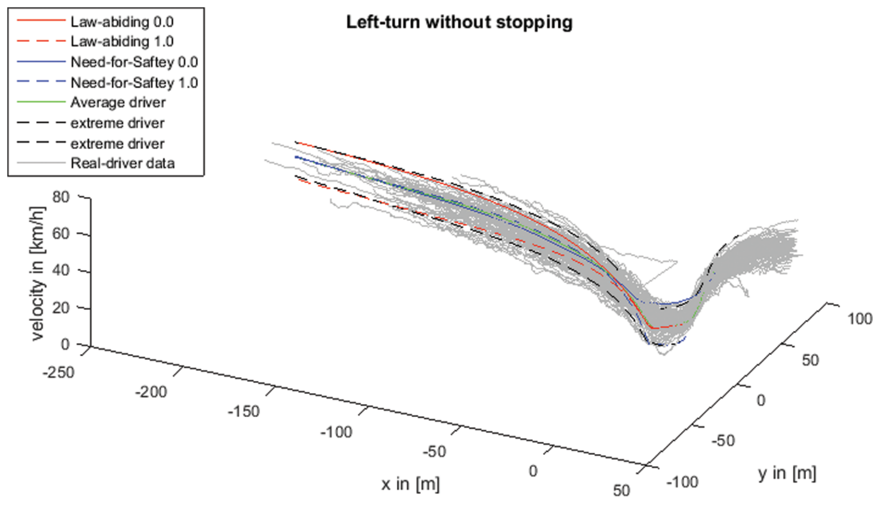

Figure 3.9 Trajectories (velocity over x- and y-position) for left turn including the simulation results for different parameter sets. The real driver data is measured on one intersection with 136 different drivers during day time.

in grey and the boundaries (extreme driver) as well as the average driver are included. The extreme drivers are generated by choosing respectively, the maximum and the minimum, of the need-for-safety and law-abiding parameters. For this example, the upper extreme driver is created by setting the need-for-safety and the law-abiding parameters to zero and the lower extreme driver is created by setting these parameters to one. As it can also be seen in the figure, the law-abiding parameter influences the speed the driver reaches before and after passing the intersection but not the velocity during the turning (red lines). Opposite to this, the need-for-safety parameter influences the speed within the turning only (blue line). This result depicts the statement that the turning speed is mainly driven by the safety and comfort motivations of the driver and the speed on straight roads is defined by the acceptance of speed limits. The phases between approaching and turning are representing a mixture of all motivations and result in a transition of the speed. In this example, the other parameters are set to the average value (0.5).

To predict the driving behaviour of a real driver in a vehicle, the driver model was integrated as a module on a real-time system in the car, equipped with real sensor data by radar and camera sensors. A five second simulation is calculated in each prediction step and the result is written to the CAN-Bus. With that data ADAS like ACC can react dependent on the estimated wish of the driver. The system and the results are published in [12].

3.7 Conclusions and Outlook

In the DESERVE project a driver model structure was developed with the focus on the realistic generation of driver-vehicle-environment interactions. For the usage in traffic simulations the driver model has been implemented in Matlab/Simulink and exemplarily been integrated in PELOPS. The addressed traffic area covered the inter-urban road network including generic intersections. Therefore, common driver model approaches but also conceived approaches to create the modules needed in DESERVE were used to obtain realistic driving behaviour. The elementary interactions between the driver models, the associated vehicles and the surrounded environment result in realistic traffic phenomena and effects occurring in equivalent real traffic situations which was shown by comparing the simulation results with measured data on a real intersection. The model behaviour is tuneable via parameters on two levels, a humanized and a physical level, which have indirect and direct influence on the model behaviour.

The structure of the model was designed to offer the possibility of enhancing the driver model by using different model approaches or expanding it with the capability of performing yet unimplemented manoeuvres and driving tasks. In those cases, the challenge is to tune the added model approaches while maintaining the realistic influence of the parameters. To simplify and partly automate the tuning process a tool can be implemented which uses real data to optimize the mathematical influence of the parameters to the model. This work will be done in the future to increase the usability of the driver model for the simulative analysis of traffic situations. The traffic simulation and thus the driver model shall be an inherent part of the tool chain used in the development of ADAS and functions of automated driving.

References

[1] T. Jürgensohn and K.-P. Timpe, *Kraftfahrzeugführung*. Berlin, Heidelberg, Springer, 2001.

[2] D. Raudszus, M. Ranovona, S. Geronimi, M. Kunert, E. Schubert, and T. Schaller, "Report on Driver and Pedestrian Reaction Models", Project Deliverable, ASPECSS, 2013.

[3] S. Fruttaldo, G. Piccinini, D. Pinotti, R. Tadei, G. Perboli, L. Gobbato, A. Zlocki, J. Klimke, F. Christen, N. Pallaro, F. Palma, and F. Tango, "D3.1.1 – Standard Driver Model definition", Project Deliverable, DESERVE, 2013.

[4] E. Donges, "A two-level model of driver steering behavior," *Human Factors,* Vol. 20, No. 6, Dec 1978, pp. 691–707, 1978.

[5] J. Rasmussen, "Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-13, no. 3, pp. 257–266, 1983.

[6] J. Klimke, F. Christen, N. Pallaro, A. Kyytinen, P. van Koningsbruggen, E. Nordin, and X. Savatier, "D4.2.1 – Control functions solution design", Project Deliverable, DESERVE, 2013.

[7] J. Klimke, F. Christen, and L. Eckstein, "Definition of a Microscopic Traffic Simulations Driver Model for Inter-urban Intersections for 21st World Congress," in *ITS World Congress 2014*, Detroit, 2014.

[8] SAE International, *Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems*. SAE International Standard J3016, 2014.

[9] "PELOPS Whitepaper," Forschungsgesellschaft Kraftfahrwesen Aachen mbH (fka), Aachen, 2014 http://www.fka.de/pdf/pelops_whitepaper.pdf.

[10] L. Eckstein, *Active Vehicle Safety and Driver Assistance Systems, Automotive Engineering III*. Lecture Notes, Institute for Automotive Engineering (ika), Aachen, 2015.

[11] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical Observations and Microscopic Simulations," Rev. E 62, Issue, Vol. 62, p. 2000, 2000.

[12] J. Klimke, P. Themann, C. Klas, and L. Eckstein, "Definition of an embedded driver model for driving behavior prediction within the DESERVE platform," in *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014*, 2014, pp. 343–350.

[13] D. D. Salvucci and R. Gray, "A two-point visual control model of steering," *Perception*, Vol. 33, No. 10 (2004), p. 1233–1248, 2004.

[14] J. Klimke, C. Klas, and L. Eckstein, "Konzept zur Strukturierung eines generischen Fahrermodells anhand des realen Informationsflusses," in *VDI-Fortschritt-Berichte: Reihe 22, Mensch-Maschine-Systeme*, 2015.

[15] R. Wiedemann, *Simulation des Straßenverkehrsflusses*. Karlsruhe: Institut für Verkehrswesen, 1974.

[16] D. Ehmanns, *Modellierung des taktischen Fahrerverhaltens bei Spurwechselvorgängen*. Dissertation, Institute for Automotive Engineering (ika), Aachen, 2003.