

Development of a Simulation Environment for Automated Model Configuration for the Design and Validation of AI-based Driving Functions

Or Aviv Yarom^a and Xiaobo Liu-Henke

Ostfalia University of Applied Sciences, Salzdhulmer Str. 46/48, 38302 Wolfenbuettel, Germany

Keywords: Autonomous Driving, Artificial Neural Networks, Matlab, Simulink, Deep Learning Toolbox.

Abstract: The further development of autonomous driving requires the increased use of innovative and intelligent algorithms. In order to develop these effectively and efficiently, suitable development methods and tools are required. Therefore, this paper presents the development of a simulation environment for automated model configuration for the design and validation of AI-based driving functions. Based on the current state of the art, the conception including requirement definition and realization of the simulation environment are described in detail. In addition, the simulation environment is validated in an application for automated vehicle guidance with Artificial Neural Networks.

1 INTRODUCTION

The innovation alliance autoMoVe (Dynamically Configurable Vehicle Concepts for Use-Specific Autonomous Driving), funded by the European Regional Development Fund (ERDF), aims to develop an autonomous, modular and electric vehicle concept. By exchanging application specific modules during runtime, a wide range of applications from internal freight transport to passenger conveyance in road traffic shall be realized autonomously. Within the scope of this research project, the Ostfalia subproject autoEVM (Holistic Electronic Vehicle Management for Autonomous Electric Vehicles) focuses on the model-based development of innovative intelligent algorithms and functions for autonomous driving.


Higher automation of driving operations is accompanied by an increase in the requirements to be met by the vehicle or the automated driving functions. Current functions and algorithms based on methods of control theory or classical information processing can no longer fully meet these (Milz and Schrepfer, 2020). Therefore, artificial intelligence (AI) represents a key technology in this project or for many domains involved in the development and use

of intelligent, automated vehicles (Fayjie et. al., 2018).

Regardless of the type of information processing, modern vehicles and driving functions are complex mechatronic systems with a high degree of internal and external interconnection. In order to handle this complexity in the development and validation process, a design methodology that is well established in mechatronics research is used. This consistent and verification oriented methodology is based on digital models and simulations to make the design and validation process of complex mechatronic systems in a crosslinked environment easier, faster and safer. (Liu-Henke et. al., 2016)

The development and simulation environments currently available for intelligent vehicle functions are very extensive in general, but they concentrate largely on conventional algorithms for information processing. The use of AI functions in development and validation is either not possible or only possible with a great amount of effort. Conversely, current development environments for AI algorithms do not offer the advantages or the functional scope of tools that are specifically designed for automated driving functions. (Stančin and Jović, 2019)

This current incompatibility of the two worlds for automated driving functions and AI is therefore now

^a <https://orcid.org/0000-0001-5627-4199>

inhibiting the research and development progress of automated and autonomous driving. Consequently, in order to realize the mobility transformation envisioned in the projects and society, new methods and tools are needed that can unite both worlds. Therefore, this paper presents the development of a simulation environment for automated model configuration for the design and validation of AI-based driving functions. On the one hand, this simulation environment's scope of functions is geared towards the development of automated driving functions and, on the other hand, it offers the possibility to use not only conventional but also AI algorithms.

2 METHODOLOGY

The complexity of modern vehicles is constantly increasing due to the higher degree of internal and external networking and the growing number of intelligent and efficient hardware and software components. In order to handle the system complexity and to avoid errors at an early stage in the design of the information processing, a holistic design methodology is indispensable. Therefore, the continuous, verification-oriented, model-based design methodology based on Rapid Control Prototyping (RCP) and Model-in-the-Loop (MiL), Software-in-the-Loop (SiL) and Hardware-in-the-Loop (HiL) simulations has been established. (Liu-Henke et. al., 2016)

The methodology is based on function-oriented physical models of a controlled system. The control function is then simulated depending on the system behavior and validated in MiL simulations at an early stage. To avoid manual programming, the model and control function are developed in block diagram-based programming languages. The subsequent automatically generated function code is again tested against the control system model in SiL simulations. HiL simulations are used for further validation and optimization of the information processing with real-time capable simulation models and real subcomponents of the system to be controlled.

The verification oriented and iterative approach of this methodology also supports the development process in the challenging task of validation. The methodology addresses the weaknesses of classical validation based on physical prototypes, such as a high expenditure of resources or safety risks for humans, machines and the environment. Due to their virtual character, MiL, SiL and HiL simulations save time and costs (Yarom et. al., 2020a). They enable

feasible and reproducible tests at any time without direct dependence on physical prototypes, times of day or human experts. Thus, simulation cycles, for different functional variants or scenarios, can be automated. This makes this methodology particularly suitable for training AI algorithms. This is because, with rare exceptions, machine learning is always iterative.

Virtual design methods like these form the basis for many intelligent systems, such as highly automated vehicles. With prototype-based testing, the hundreds of thousands of test kilometers required would not be achievable in a reasonable amount of time and at a reasonable cost. (Yarom et. al., 2020a)

3 STATE OF THE ART

3.1 Intelligent Driving Functions

In automated driving, individual driving tasks are taken over from the human driver by so called advanced driver assistance systems (ADAS). Such ADAS, e.g. for speed control or lane keeping assistance, have been available in series production for some time (Kukkala et. al., 2018). ADAS process the data collected by vehicle and environment sensors and thus calculate driving commands, which are then implemented by means of controlled actuators. With an increasing number and interconnection of ADAS, the human driver successively delegates driving tasks to the vehicle until he finally becomes a passenger in autonomous driving. Then we no longer speak of ADAS, but of (automated) driving functions.

Increasing automation of the driving process means an extreme increase in the requirements for the driving functions. Not only more but also different sensors are needed for environment perception, whose inhomogeneous raw data must be processed and implemented repeatedly, intensively and with the highest real-time requirements. If information from internal bus or external vehicle-to-everything (V2X) communication is added, the complexity increases even further. This pushes conventional algorithms for control and data processing to their limits. (Milz and Schrepfer, 2020) For this reason, AI algorithms are already being used today for automated driving functions. Artificial neural networks (ANNs) in combination with machine learning (ML) are particularly promising. Prominent applications are image-based semantic segmentation of the driving environment (Lyu et. al., 2019) or automated vehicle guidance (Huang et. al., 2019). AI algorithms are promising for the further development of autonomous

vehicles due to their performance, robustness and adaptability (Kuutti et. al., 2021). There is also potential for intelligent vehicle functions outside of automated vehicle guidance, e.g., for battery management (Alaoui, 2019). On the other hand, conventional approaches are often still used at lower levels of information processing, e.g. local control of actuators (Milz and Schrepfer, 2020).

3.2 Fundamentals of Artificial Neural Networks and Machine Learning

The term AI covers a variety of different methods and algorithms that deal with the autonomous and automated solving of problems (Togelius et. al., 2018). ANNs and ML form a subfield of AI that has been shown to be suitable in numerous problems in a wide variety of domains, including autonomous driving. Therefore, this paper focuses on this subfield. The numerous positive properties of ANNs and ML, such as adaptability, error resistance, versatility, and above all learning ability, can be traced back to their similarity to the structure and functioning of the human brain.

Analogous to biology, (artificial) neurons are processing units that accumulate input stimuli via weighted connections and compute an output using an activation function. The interconnection of several neurons in at least two layers makes up the ANN. Combinations of up to several hundreds of neurons in up to more than one hundred layers are common. Not only arbitrary forward but also time-feedback connections are possible in the ANN. (Skansi, 2018) The optimal architecture of an ANN cannot be determined analytically so far (Tirumala, 2020). Therefore, experience and test series are necessary to find a suitable architecture in the trade-off between computational effort and performance. The number, interconnection and weighting of connections characterizes the "intelligence" of an ANN. Generally speaking, more neurons and connections mean a higher performance of the ANN, while at the same time the computational effort increases.

Just like a human brain, the ANN must first learn or train a task. These terms refer to the adaptation of the connection weights. In the environment of autonomous driving, supervised learning (SL) and reinforcement learning (RL) are relevant for this. In SL, the ANN is provided with input data and the corresponding output. The ANN iteratively learns the relationship between the two variables (Duriez, Brunton and Noack, 2017). This learning procedure is particularly suitable for image based object recognition, for example (Lyu et. al., 2019). In RL,

the ANN successively learns the optimal strategy from the experience of past sequences in terms of a given reward function (Duriez, Brunton and Noack, 2017). This procedure is used when no training data is available, e.g., in automated vehicle guidance (Huang et. al., 2019). SL and RL are head categories of learning procedures, with diverse concrete training algorithms. Just like the ANN architecture, the optimal training algorithms or their parameters cannot be determined analytically. Thus, experience and experimentation are required here as well.

3.3 Development Environments for Intelligent Driving Functions

For the model-based design of automated driving functions, several development and simulation environments exist, such as MATLAB/Simulink with toolboxes, dSPACE Automotive Simulation Models (ASM), and IPG CarMaker, to name just a few. They all differ in terms of their primary focus or their specific advantages and disadvantages. All of them offer extensive model libraries for traffic, vehicle dynamics, component, sensor or even driver models. They are specifically designed for the configuration and reproducible simulation of a wide range of driving scenarios and vehicle variabilities for the purpose of vehicle development or validation. The tools usually have integrated model configurators, visualization, experiment environments, and in some cases scenario and test managements. The tools listed are suitable for the design methodology described in section 2 using the RCP process. They enable not only MiL simulations, but also real-time SiL and HiL simulations through automatic code generation in conjunction with MATLAB/Simulink. (Deter et. al., 2021) However, the tools do not offer the possibility to integrate ANNs into the respective simulation and configuration tools without further effort, let alone to train them.

The only exception here is MATLAB's Deep Learning Toolbox (DLT). With this toolbox, any ANN architecture can be configured and trained with a large number of pre-implemented algorithms. The DLT is basically compatible with Simulink and automatic code generation. However, ANN configuration is quite complicated and is usually done manually for individual ANNs. As a result, it is not very suitable in its native form for carrying out test series with varying architectures, training parameters or even different driving functions.

There are also specific development environments for ANN training. Most of them are based on the Python programming language.

Prominent tools are, among others, TensorFlow, PyTorch, Keras and Caffe. They all have comfortable and extensive functionalities regarding the creation and training of ANNs. However, there is only the possibility of scripted programming. A clearer block diagram-based programming suitable for the RCP process is not available. (Stančin and Jović, 2019) Therefore, it is difficult to design simulations with the same accuracy, computation time and comfort with the ANN-specific tools as with the tools for automated driving functions. The effort for this is so large that even computer games like Grand Theft Auto (Wang et. al., 2019) or TORCS (Zhang and Cho, 2017) have been coupled for vehicle and traffic simulations in combination with the mentioned ANN tools. However, these approaches rather served the investigation of ANNs and ML and do not meet the requirements of accuracy, real-time capability, reproducibility and variation possibility of a real driving function development.

4 CONCEPTION OF THE SIMULATION ENVIRONMENT

4.1 Deriving the Problem Statement

The findings from section 3 can be used to derive a problem statement for pursuing the project goals (section 2). The design of intelligent automated driving functions requires the use of AI algorithms. However, as described, these cannot easily be used in the usual development environments. ANNs and ML methods quickly become very complex and confusing. A manual programming of different ANN architectures with the associated calculation rules would not only be error-prone but would also take a lot of time. The same applies to the ML procedures, which would have to be parameterized and adapted for each ANN architecture. The fact that architecture design, training and test processes are always empirical and iterative requires many simulation cycles and further worsens the situation.

The use of ANN-specific development environments is also ruled out. If the automated driving functions are to be designed under realistic conditions, the depth of modeling must be sufficiently precise. In the classical way of analytical physical modeling, the mathematical equations and their numerical solution methods would have to be programmed independently in a script language. If one now considers the number of subsystems of a vehicle, further driving functions or other road users

that are to be simulated, a highly complex simulation system results. In addition, there are different variants, configurations, scenarios and, if necessary, real-time requirements. Although the implementation of such a project would be possible in principle, it would be very error-prone and not very effective. Finally, there are already resource-optimal simulation environments for this purpose.

In summary, none of the currently available simulation environments is suitable for the design and validation of AI-based driving functions according to the development methodology from section 2. The conclusion is therefore that a separate simulation environment must be developed for this design methodology.

4.2 Definition of Requirements

To address the aforementioned challenges, the requirements for the new simulation environment for automated model configuration for the design and validation of AI-based driving functions are defined below:

- R1 Usable for various driving functions
- R2 Operation with Simulink and compatibility with corresponding blocks and models
- R3 Use of various existing or creation of own models and functions with any modeling depth
- R4 Easy creation and calculation of ANNs
- R5 Automatic integration of ANNs into the Simulink models
- R6 Variation of model configuration
- R7 Automatic model configuration
- R8 Automated execution of simulation series
- R9 Training by means of different ML methods
- R10 Use for generation of training data for ML
- R11 Visualization in 2D and 3D
- R12 Operation with user interface or scripts
- R13 High-performance computing times for large simulation series and later real-time applications
- R14 Compatibility with dSPACE ASM for later development of the simulation environment
- R15 Possibility of automatic code generation

4.3 Concept Formation

The first step to fulfill the previously defined requirements, is the realization of the simulation environment in MATLAB and Simulink (R2). This ensures compatibility with standard blocks and

existing models available in Simulink (R3). Furthermore, any other models and automated driving functions can be created (R1). In principle, toolboxes can also be used for this purpose. It is only important to ensure that these support automatic code generation (R15). In this case, compatibility with dSPACE ASM is also guaranteed (R14).

For the configuration and creation of ANNs, the DLT is used. For this purpose, an additional ANN generator is designed, which creates arbitrary ANNs architectures automatically by simple user input via a graphical user interface (GUI) or a script (R4) and generates them directly into the simulation model (R5). By using the DLT, several ML algorithms are automatically available (R9). Several algorithms are already usable for SL, given that one has generated training data (R10). RL algorithms are available in the RL toolbox of MATLAB. Alternatively, it is possible to implement custom training algorithms for both SL and RL, since the DTL provides convenient access to the connection weights.

With a scenario generator, different simulation scenarios can be configured with respect to the route and the road participants (R6). This configuration is then automatically transferred to the simulation model. In order to better interpret the events in the scenario during and after the simulation, an additional rudimentary visualization is implemented. The 2D and 3D visualization is based on the Bird's-Eye Scope and the plot function of MATLAB (R11).

The configuration and creation of the scenario is also done by a GUI or a script (R12). With the possibility of script-based scenario and ANN generation, simulation sequences can be automated (R8). The corresponding model parameters are automatically configured and updated (R7). Just as with the developed driving functions and models, a lean and computationally optimized programming must also be taken into account for the realization of the scenario and ANN generator as well as the automation mechanisms (R13).

5 REALIZATION OF THE SIMULATION ENVIRONMENT

Figure 1 shows the setup and structure of the simulation environment. It mainly consists of the simulation model itself and the control of the simulation environment. In the simulation model, the modeling and the calculation of the simulation take place. It contains the models, parameters and functions of the entire ego vehicle in the desired

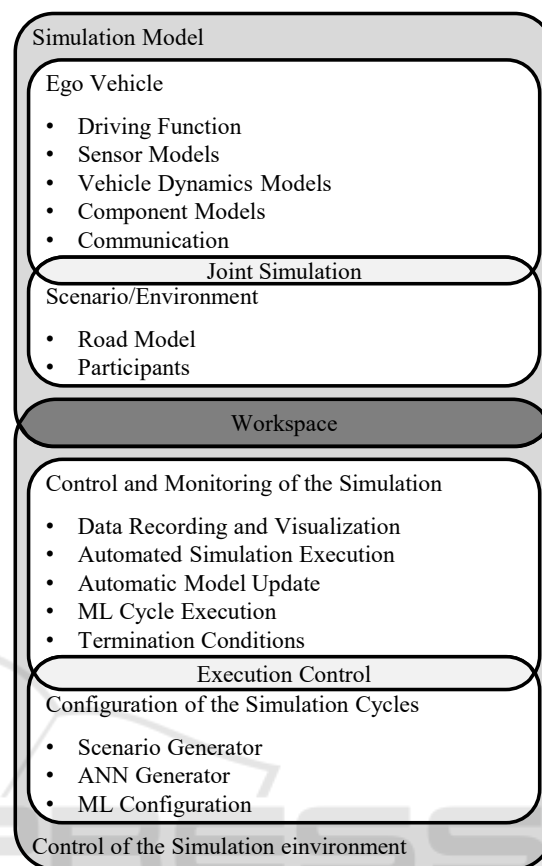


Figure 1: Setup and structure of the simulation environment for automated model configuration for the design and validation of AI-based driving functions.

modeling depth as well as the environment simulation defined in the scenario. The ego vehicle is the "own" vehicle for which the automated driving functions are developed. The ego vehicle is usually modeled in a very detailed way (in contrast to the other third-party vehicles) and contains several components in the desired modeling depth, depending on the application. These include, for example, the driving functions to be developed (AI-based or otherwise), sensors, vehicle dynamics, communication or component models. These individual model components of the ego vehicle communicate with each other as well as with scenario model components via defined interfaces. The scenario model includes the various components of the driving environment, such as route models with infrastructure, third-party vehicles, other traffic participants including their behavior or information from V2X communication.

Before any simulation can take place, the simulations must first be configured. Thus, the control of the simulation environment is divided into the configuration of the simulation cycles on the one

hand, and the simulation control and monitoring on the other hand. The configuration of the simulation cycles is basically the input interface of the user. With the scenario generator it is possible to select how exactly each simulation should look and run according to the function specification. For example, the route, the type, number and movement of other road users, etc. can be set. Of course, the composition and modeling depth of the ego vehicle must also be defined.

As mentioned, this simulation environment can be used to design not only AI-based functions, but also functions based on conventional methods. However, since this paper focuses on the use of ANN, the connection of the simulation to their training will now be described. After the general setting of the simulations has been determined, it is important to first configure the ANN architectures to be used. The ANN generator allows a convenient configuration of these ANNs in order to insert them into the simulation as a model component (driving function). The training of the ANNs is always iterative and requires the execution of several simulations in the whole scenario in which the ANN is used with the respective architecture and the associated weights. The goal of the training or the simulations is the successive improvement of the ANN performance with respect to the development requirements. The simulations run according to a specific scheme depending on the training algorithm and training parameters. The configuration of the training in the ML Configuration allows the simulations (also ML cycles) to be automatically adapted and executed according to this scheme. In summary, the information from the user input is used as execution control for the “Control and Monitoring of the Simulation” (Figure 1).

Thus, an internal flowchart is generated to automate the simulation cycles. In it, for example, several ML cycles are started first in one scenario and then in another. Before each cycle, the corresponding parameters of the active configuration are loaded and transferred to the simulation model via the MATLAB workspace. In this way, the model is automatically updated. The workspace serves as an interface between the simulation control and the simulation model. It enables the recording and storage of simulation data for visualization and later evaluation. Furthermore, it transmits selected variables for model monitoring. In case of previously defined, inadmissible conditions, the simulation is automatically aborted.

6 SIMULATION AND EVALUATION

6.1 Description of the Use Case and Modeling

To demonstrate this, a driving function for automated lateral guidance at constant speeds on arbitrary, one-lane routes without other road users is to be designed in this use case. The example is consciously chosen in a compressed way to show the simulation and time effort. This is to illustrate the benefit of the automated model configuration.

Since the ego vehicle’s speed is constant in this application, a linear single-track model is used to represent the vehicle dynamics. Automatically generated routes according to the guidelines of the German Federal Highway Research Institute with a constant width of 3.5 m form the environment. The sensor model consists of eleven lines, which detect the distance to the lane boundaries in an angular range of $\pm 40^\circ$ and a radius of 8 m. Figure 2 a) and b) illustrate the use case with the visualization function of the simulation environment. Other vehicle components and communication systems of the ego vehicle are not considered. The automated lateral guidance is executed by an ANN in the simulation. The eleven sensor values are the inputs, a corresponding steering angle is the output.

The ANN is supposed to learn a natural steering behavior with a self-implemented RL method, so called Genetic Algorithms (GA). Natural steering behavior in this case means constantly keeping the center of the road and avoiding strong or high-frequency oscillations to ensure safe and comfortable driving behavior. GAs are a group of algorithms that imitate the natural process of evolution in order to successively approach an optimal solution. A so

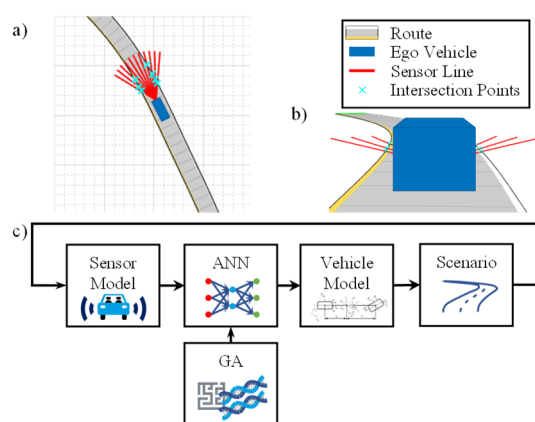


Figure 2: Overview of the use case and modeling.

called population, which consists of several solution candidates (individuals), evolves generation by generation by selection, recombination and mutation. A population size of 50 means here that 50 different ANNs are simulated in one generation. The iterative character of the GA tends to result in relatively many simulations.

Figure 2 c) shows an overview of the basic structure of the simulation model. In a simulation, the ego vehicle is placed on the track or scenario. There, in one simulation step, the sensor model implicitly captures the position and orientation of the ego vehicle on the track via the eleven sensor lines. The eleven sensor signals are passed to the ANN which calculates a steering angle. This is used in the vehicle dynamics model to calculate a new vehicle position on the track. This process is repeated until the ego vehicle reaches the destination or exceeds one of the road markings. The GA is actually located outside the simulation model. The associated reward function accesses simulation data to evaluate the behavior of individuals. The GA uses this information to update the connection weights of the ANN. For this purpose, several simulation runs are performed according to a certain scheme of the ML cycle (Section 6.2).

6.2 Configuration of the Simulation Cycles

As described in section 3, the determination of the ANN architecture and the parameters of the ML algorithm is always empirical. Furthermore, it is important to pay attention to the generalization capability when training ANNs. In this case, this means that the ANN must be able to perform the automated lateral guidance even on unknown routes. To consider this aspect directly during the training, several test runs are performed directly after every single training. Thus, it is necessary to configure several different simulations.

In order to successively achieve an ideally designed driving function, multi-step tests must be implemented in the configuration of the simulation cycles. The simulation cycle always consists of several ML cycles. One ML cycle always includes a training of one ANN architecture with one GA parameter set on one route with one reward function, followed by four test runs on additional routes. While the training phase of the GA consists of many separate simulations for the individuals and generations, only the best individual from the training is tested in each of the test runs. Thus, the number of simulations S_i to be performed per ML cycle i is a function of the population size P_i , the number of

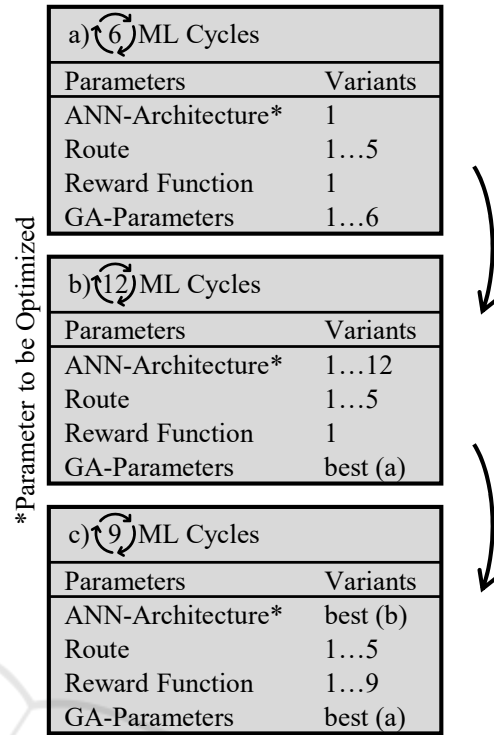


Figure 3: Sequence of the automatic simulation execution.

generations G_i and number of test runs T_i . The number of simulations in the total cycle N_G is the sum of all S_i for the different ML cycles or configurations K_{ML} :

$$N_G = \sum_{i \in K_{ML}} S_i = \sum_{i \in K_{ML}} P_i \cdot G_i + T \quad (1)$$

Assuming a population size of 50 and a generation number of 25, 1254 simulations are consequently performed in one ML cycle.

The first configuration involves the investigation of six variants of the GA parameter sets for training a base ANN (Figure 3 a)). The base ANN is a non-optimized ANN that is assumed to be able to perform the function fundamentally. The same is true for the basic reward function. The resulting optimal GA parameter set is used to determine the best of twelve preconfigured ANN architectures, as shown in Figure 3 (b). In the final step (Figure 3 c)), the actual optimization of the ANN behavior is performed by performing further ML cycles with nine different reward functions. According to equation (1), the total number of simulations N_G performed is in the high six-digit range. Without the automatic model configuration of the automated simulation environment, the required effort would have exceeded a reasonable level.

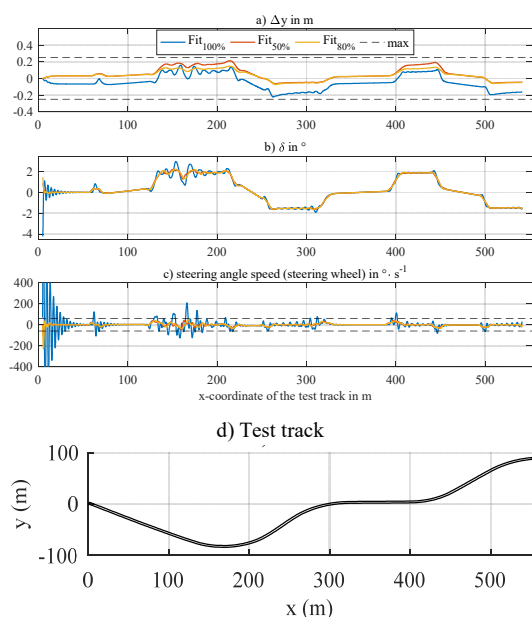


Figure 4: Simulation results of the use case on a test track.

6.3 Simulation Results and Evaluation

After the automatic execution of all configurations from Figure 3, a function for automated lateral guidance has been created. The ANN can safely and comfortably take over lateral guidance on any route in a speed range from 30 to 70 km/h. Corresponding simulation results are shown in Figure 4. There, the lateral deviation from the middle of the lane Δy (Figure 4 a)), the steering angle δ (Figure 4 b)) and the steering angle velocity $\dot{\delta}$ (Figure 4 c)) are shown for different fitness functions over the x-coordinate of a test track (Figure 4 d)). The figure thus illustrates a subsection of the ML cycle from Figure 3 c) for optimizing the behavior with the reward function. The best driving behavior in terms of comfort and safety was achieved with the $Fit_{80\%}$ function (yellow) due to the lowest lateral deviation and oscillations. In this fitness function, the ego-vehicle or ANN was rewarded for moving forward on the track as well as for reaching the destination. It was penalized for deviations from the middle of the lane as well as for large steering angle changes. For more detailed descriptions of the design and validation of the driving function from this use case, please refer to the previous work (Yarom et. al., 2020a) and (Yarom, Jacobitz and Liu-Henke, 2020b).

The focus of this paper is on the simulation environment. For its evaluation, a single automated ML cycle was hand-programmed in MATLAB and compared with the simulation environment in terms of runtime. In the hand-programmed version, all

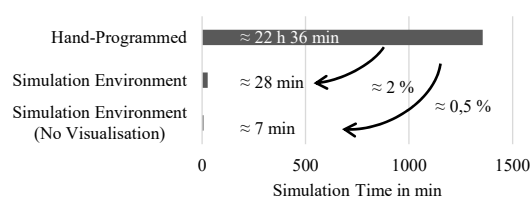


Figure 5: Comparison of simulation times in the use case.

model components according to Figure 2 c) as well as the visualization were implemented. The comparison according to Figure 5 showed that a single ML cycle in the hand-programmed environment takes about 22 h and 36 min. In the simulation environment for the automated model configuration, on the other hand, an ML cycle with identical configuration takes about 28 min, which is only about 2% of this time. Furthermore, it is possible to switch off the visualization in the simulation environment. In this way, the time required for each ML cycle can be further reduced to a quarter.

The simulation environment presented in this paper now makes it possible, on the one hand, to design automated vehicle functions based on ANN. On the other hand, all further requirements from section 4.2, e.g. optimized computing time, compatibility, usability and automation, have been implemented. Thus, it exactly fulfills the originally intended purpose of uniting the worlds of "modeling and design of automated vehicles" and "AI development" or their respective advantages. Thus, a tool has been created with which the development and validation of automated vehicles can be developed safely and efficiently according to the methodology presented in section 2. The result is a significant contribution to the progress of the project and autonomous driving.

7 CONCLUSION AND OUTLOOK

In this paper, a simulation environment for automated model configuration for the design and validation of AI-based driving functions was presented. Starting with an introduction and motivation, the design methodology and the state of the art were presented. From this, the necessity for the development of the presented simulation environment was derived and requirements for it were defined. The implementation of the requirements was described in the concept and the realization. The result is a simulation environment based on MATLAB and Simulink, which has a high degree of compatibility with existing development environments. In addition, it can automatically

execute and visualize large simulation series in a short time after a user-friendly configuration. This makes it ideally suited for use in the design methodology presented. Finally, the simulation environment was used in an example application, demonstrating its benefits and functionality. Individual results of the application as well as their relevance for the simulation environment were presented and critically discussed. Future work steps include extending the model and function library and integrating it with dSPACE ASM.

ACKNOWLEDGEMENTS

This publication resulted from the subproject "autoEMV" (Holistic Electronic Vehicle Management for Autonomous Electric Vehicles) in the context of the research project "autoMoVe" (Dynamically Configurable Vehicle Concepts for a Use-specific Autonomous Driving) funded by the European Fund for Regional Development (EFRE | ZW 6-85030889) and managed by the project management agency Nbank.



REFERENCES

- Alaoui, C. (2019). Hybrid Vehicle Energy Management Using Deep Learning. *2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS)*, Taza, Morocco.
- Deter, D., Wang, C., Cook, A., Perry N. K. (2021). Simulating the Autonomous Future: A Look at Virtual Vehicle Environments and How to Validate Simulation Using Public Data Sets. In *IEEE Signal Processing Magazine*, vol. 38, no. 1.
- Duriez, T., Brunton, S., Noack, B. R. (2017). *Machine Learning Control*. Springer International Publishing, Cham, Switzerland.
- Fayjie, A. R., Hossain, S., Oualid D., Lee, D. (2018). Driverless Car: Autonomous Driving Using Deep Reinforcement Learning in Urban Environment. *2018 15th International Conference on Ubiquitous Robots (UR)*, Honolulu, Hawaii.
- Huang, Z., Xu, X., He, H., Tan, J., Sun, Z. (2019). Parameterized batch reinforcement learning for longitudinal control of autonomous land vehicles. In *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 4.
- Kukkala, V. K., Tunnell, J., Pasricha, S., Bradley, T. (2018). Advanced Driver-Assistance Systems: A Path Toward Autonomous Vehicles. In *IEEE Consumer Electronics Magazine*. vol. 7, no. 5.
- Kuutti, S., Bowden, R., Jin, Y., Barber, P., Fallah, S. (2021). A Survey of Deep Learning Applications to Autonomous Vehicle Control, In *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2.
- Milz, S., Schrepfer, J. (2020). Is artificial intelligence the solution to all our problems? Exploring the applications of AI for automated driving. In Bertram T. (eds) *Automatisiertes Fahren 2019*. Springer Vieweg, Wiesbaden, Germany.
- Liu-Henke, X., Scherler, S., Fritsch, M., Quantmeyer, F. (2016). Holistic development of a full active electric vehicle by means of a model-based systems engineering. *2016 IEEE International Symposium on Systems Engineering (ISSE)*, Edinburgh, UK.
- Lyu, H., Fu, H., Hu, X., Liu, L. (2019). Edge-Based Segmentation Network for Real-Time Semantic Segmentation in Traffic Scenes. *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan.
- Skansi, S. (2018). *Introduction to Deep Learning. Undergraduate Topics in Computer Science*. Springer, Cham, Switzerland.
- Stančin I., Jović A. (2019). An overview and comparison of free Python libraries for data mining and big data analysis. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia.
- Tirumala, S.S. (2020). Evolving deep neural networks using co-evolutionary algorithms with multi-population strategy. In *Neural Comput & Applic*, vol. 32.
- Togelius, J., Juul, J., Long, G., Uricchio, W., Consalvo, M. (2018) What Is (Artificial) Intelligence?. In *Playing Smart: On Games, Intelligence, and Artificial Intelligence*. MIT Press.
- Wang, D., Devin, C., Cai, Q. -Z., Yu, F., Darrell, T. (2019). Deep object centric policies for autonomous driving. *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, Canada.
- Yarom, O. A., Scherler, S., Goellner, M., Liu-Henke, X. (2020a). Artificial Neural Networks and Reinforcement Learning for model-based design of an automated vehicle guidance system. *12th International Conference on Agents and Artificial Intelligence (ICAART)*, Valletta, Malta.
- Yarom O. A., Jacobitz S., Liu-Henke X. (2020b). Design of Genetic Algorithms for the Simulation-Based Training of Artificial Neural Networks in the Context of Automated Vehicle Guidance. *2020 19th International Conference on Mechatronics - Mechatronika (ME)*. Prague, Czech Republic.
- Zhang, J., Cho, K. (2017). Query-efficient imitation learning for end-to-end autonomous driving. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, San Francisco, USA.