A METHOD FOR THE SIMULATION-BASED PARAMETER OPTIMIZATION OF AUTONOMOUS EMERGENCY BRAKING SYSTEMS

Thomas Hierlinger
Tobias Dirndorfer
Tobias Neuhauser
AUDI AG
Germany

Paper Number 17-0188

ABSTRACT

This paper presents a simulation-based method automizing the application, performance evaluation and testing of predictive safety functions using the example of current AEB systems. The approach addresses the growing scenario complexity and the increasing performance requirements with several intended uses along the function development process. The approach overall aims at reducing specification, application and test costs by continuous simulation along the whole development process. The toolchain consists of three tools: Matlab, rateEFFECT and Optimus. Matlab serves as controller of the toolchain where the pre- and postprocessing takes place and the objective functions are defined. rateEFFECT is used as the underlying simulation environment. The driving simulation itself runs in this program and all kinds of load cases can be simulated. The optimization algorithms are provided by the tool Optimus due to its easy integration in the toolchain and above all its hybrid optimization techniques. Two simple ideas are presented to measure safety performance and customer acceptance. Furthermore, several optimization strategies and algorithms are analyzed: the metamodel-based-, the direct- and the hybrid-optimization.

MOTIVATION

The increasing performance of electronic control units, sensor and actuator components allows the development of advanced driver assistance and piloted driving systems supporting or performing the driving task on various automation levels. Adaptive cruise control (ACC) systems for instance overtake the longitudinal vehicle movement in comfort driving situations whereas automatic emergency braking (AEB) functions temporarily intervene in impending accident scenarios. Both systems assist in defined driving modes, require manual environment monitoring and can always be oversteered by the driver. Piloted driving functions gradually increase the system level of automation by executing longitudinal as well as lateral vehicle movement, additionally monitoring the vehicle environment and addressing more and more driving situations. Along with the growing automation levels the implemented driving functions have to fulfill more and more complexity and performance requirements concerning their behavior in various scenarios and situations (see [1] and Figure 1).

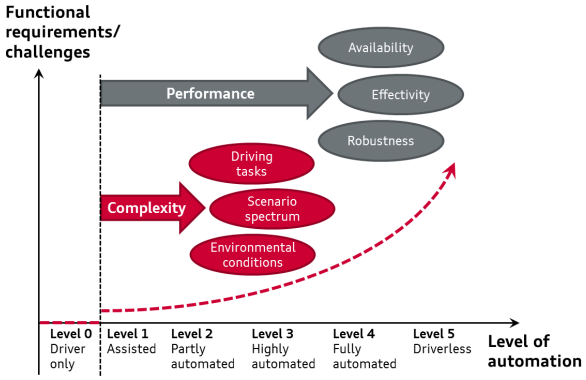

Figure 1. Increasing requirements for assisted and automated driving functions

The requirement growth is supported by legislation, insurance and consumer protection organizations as well as the car manufacturers themselves continuously defining new targets for maximizing the effecivity of assistance or piloted functions in a given environment. The approach is often to abstract the respective traffic and accident occurrence to a finite number of representative sampling points addressing the relevant events to a maximum extent. In case of current AEB systems the number and difficulty of functional requirements is already approaching a level where automated system application becomes essential. In order to address the whole relevant traffic scenario spectrum future automated driving systems in principal need to handle an infinite

number of potentially critical situations with maximum effect on collision avoidance or severity reduction. That is why both manual system application as well as real car tests have to be substantially supported by automated simulation and optimization methods for a sufficient coverage of relevant scenarios.

OBJECTIVE

This paper presents a simulation-based method automizing the application, performance evaluation and testing of predictive safety functions using the example of current AEB systems. The approach addresses the growing scenario complexity and the increasing performance requirements with several intended uses along the function development process (see Figure 2).

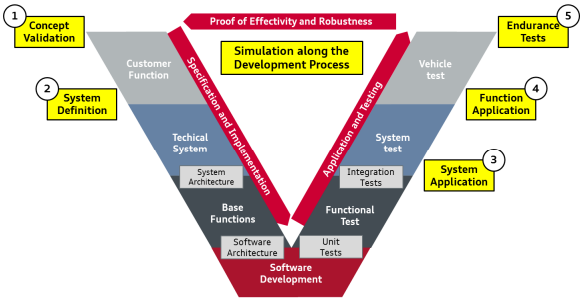

Figure 2. Simulation support along the functional development process (V-Model, e.g. see [2])

The presented approach is supposed to:

- support the validation of function concepts (1) and system definition (2) in the early development phase
  - early predict function performance in consumer ratings and real world scenarios
  - evaluate the effect of system parameter variations (e.g. sensor and actuator characteristics)
- balance safety performance and customer acceptance during system (3) and function application (4)
  - provide a base parameter set for refinement in real system and vehicle tests
  - quickly evaluate the effect of changes in parameter sets
- enlarge test coverage by means of virtual endurance tests (5)

The approach overall aims at reducing specification, application and test costs by continuous simulation along the whole development process.

METHODICAL APPROACH

During the definition and application phase of a predictive AEB function the major goal is to create effective automatic braking interventions with maximum coverage of relevant load cases and maximum respective velocity reduction which is only activated legitimately and does not disturb the driver executing the driving task. This problem represents a goal conflict towards the AEB function F with its global parameters P_{glob} adressing a number of load cases LC with weighting $W(LC)$. Its solution requires an optimization process for finding the best parameter set P_{opt} in terms of a defined deployment cost function C characterizing the functional effect and acceptance (see Figure 3).

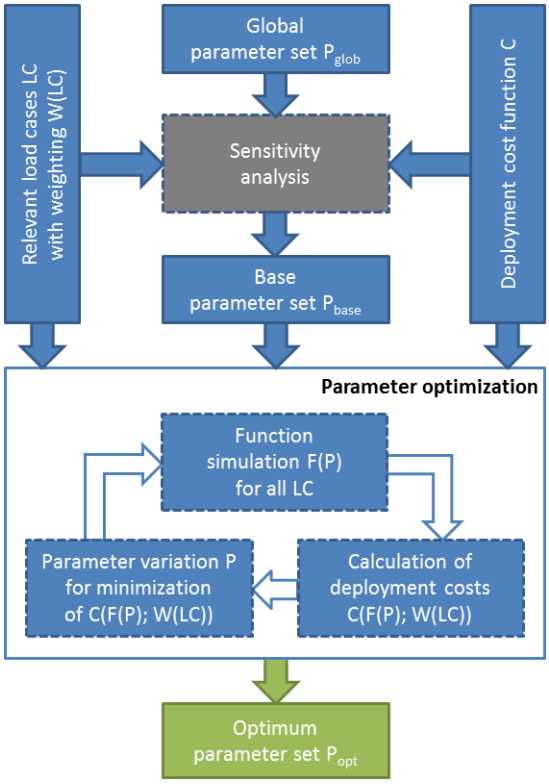

Figure 3. Fully automated parameter optimization process

In order to assure the best optimization results for a large amount of parameters under given time restrictions the global parameter set P_{glob} can optionally be reduced to the functionally most relevant base parameter set P_{base} by means of a sensitivity analysis checking the individual parameter effects on the deployment costs C for the weighted load cases LC.

Based on the general optimization process a concrete toolchain fulfilling the following requirements was developed:

- Simulation in multiple real time to conduct many simulations in short time
- Ability to perform global optimization / multi-objective optimization
- Easy modular expandability
- Closed-loop simulation ability
- Open interface
- Integrability of series ECU algorithms

Description of the general toolchain

The method is software-in-the-loop (SIL) based. Figure 4 shows the used toolchain for optimizing the AEB function parameters. The core consists of three tools:

(1) *rateEFFECT* is a tool developed and used by Volkswagen Group to evaluate the effectivity of active safety systems [4]. In this approach it is used as the underlying simulation environment. The vehicle dynamics and the scenery is based on PC-Crash where *rateEFFECT* interacts in every integration step of PC-Crash. All kinds of load cases like load cases from consumer ratings, load cases from real world accidents like GIDAS (German In-Depth Accident Study) or no collision load cases can be simulated. Via a system editor it is possible to define own active systems with predefined or self-developed function blocks. The system configuration generally consists of sensors, algorithms, driver models and actuators. The described system in this paper uses idealized ego- and object sensors, the wrapped series ECU algorithm and the brake actuator. System delays are modelled between the sensor and algorithm as well as between the algorithm and actuator. A driver model is currently not used for the optimization of the AEB function. Besides the use of idealized sensors it is clearly possible to incorporate detailed sensor models into the overall system. The same applies to the driver model. The behavior of the AEB function is defined by its parameters.

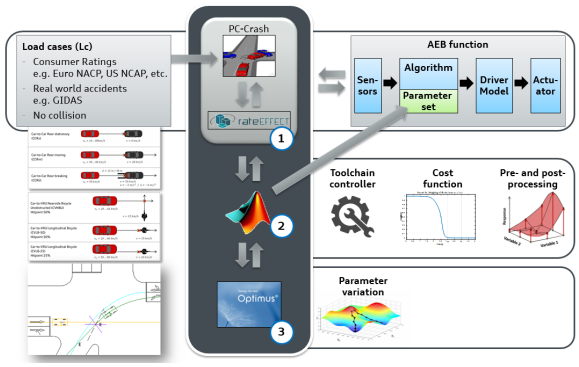

Figure 4. Overview of the used toolchain

(2) *MATLAB* serves as controller of the toolchain where the main GUI for pre- and post-processing runs. Several characteristic values (e.g. impact velocity, begin of specific action, etc.) can be visualized, the achieved points in consumer ratings are calculated automatically and single load cases can be studied in detail by visualizing signal sequences. The cost functions are defined here and the parameters, which shall be used in the optimization process, are chosen. To preselect appropriate parameters, sensitivity analysis can be performed to omit non-influential parameters for the actual optimization.

(3) The results after each *rateEFFECT* simulation are collected and passed to *Optimus* [3] where the actual parameter variation process takes place. After the parameter variation step *MATLAB* takes the results from Optimus, hands the new parameter set to *rateEFFECT* and a new iteration begins. Optimus from NOESIS is a process integration and design optimization software platform and is used due to its easy integration in the toolchain and above all its hybrid optimization techniques. Especially in the beginning, the analyzing features of Optimus helped a lot to find the best optimization algorithms. With enough knowledge about the simulated system, its behavior and the most suitable optimization algorithms the *MATLAB* optimization toolbox could be used as well.

Specific example

The following paragraphs show the approach for the specific simulation-based optimization of an AEB system. The shown work in this paper restricts itself to optimize the parameter set, which triggers the AEB and the previous brake jolt (see Figure 5, red outlined). The optimization of suppression and abortion criteria is out of scope.

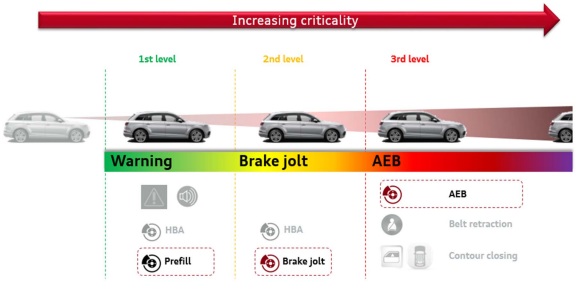

Figure 5. Possible action cascade of an AEB function

Optimization methods and strategies

Several optimization methods were analyzed for the simulation-based parameter optimization of the AEB system described above:

- Direct optimization [5]: The direct optimization applies the optimization algorithm directly on the simulation model. However, the computational costs might prohibit this approach, especially for global optimization, as they are caused by many evaluations of the simulation model
- Metamodel-based optimization [5]: In the metamodel-based optimization the optimizer works on a metamodel instead of the simulation model. Therefore, the metamodel-based optimization makes use of the benefits of a metamodel. The key benefit is the low computational costs of a metamodel. Additionally, smoothness of metamodels simplifies the use of gradient-based optimization algorithms. However, the metamodel introduces an additional source of error as the metamodel cannot fully reproduce the behavior of the actual simulation model. Various interpolating and extrapolating methods can be used as metamodel (e.g. Kriging [9], radial basis functions (RBF) [6], quadratic least squares [9], neuronal networks [9], etc.)
- Hybrid optimization [6]: The quality of a metamodel also depends on the number of samples. However, to sample the whole design space in a fine granular manner is computationally costly - especially since samples are wasted on regions, which are far off the optimum and not of interest. Hybrid optimization methods sample the design space in an adaptive manner. After an initial sampling and an initial metamodel, new samples are systematically added to support the metamodel in interesting areas – areas with a high probability to find the optimum.

Besides the optimization method, a key point is the selection of the load cases. Figure 6 theoretically shows the classification of load cases in “collision” and “no collision”. It is important to preselect only the load cases, where the critical object is in the sensor field of view and the AEB function is possible to interact or where the AEB must not brake. An AEB function shall brake in critical situations (1), but must not brake in uncritical situations (2). Apart from that there are also some situations where the AEB may brake (3). Load cases from the “may brake” area should not be selected since it is not possible to formulate a single goal for this kind of load cases. The operational effect of the AEB has boundaries and the AEB works only within these boundaries (e.g. the ego velocity).

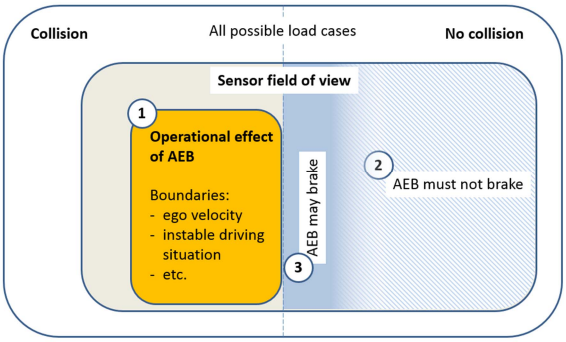

Figure 6. Classification of load cases

Again, the main objective is the optimization of the AEB function for safety performance and robustness concerning false braking (false positives) which is crucial for customer acceptance.
Therefore, two different approaches are possible:

1. Simulating only collision load cases in the optimziation
   a. Restrict the AEB function to a certain maximum amount of velocity reduction
   b. Limit the time of first braking via a specific cost function
   c. Check the behavior of the AEB function after the optimization with some no collision load cases
2. Simulating both collision an no collision load cases in the optimization process
   a. No restriction of velocity reduction necessary
   b. No limitation of first time of braking necessary

The decisive factor for both approaches is the right load case selection. On the one hand, the selection should cover the whole operational effect of the AEB (and if approach 2 is used, the “AEB must not brake” area as well), on the other hand, too many load cases slow the optimization process down and could prevent finding a reasonable solution. Therefore, another important issue is the weighting of the load cases. If the weighting is conducted according to their occurrence in real life one could take this occurrence from in house accident data or accident studies like [7] and [8].

The whole load case selection process could look like described below:

a) Filter relevant load cases (see Figure 6) from load case database (e.g. consumer rating load cases, real world load cases, car manufacturer load cases)
b) Determine the most important load cases from a) and group them into scenarios if necessary
c) Weight the load cases according to e.g. occurrence, objects involved, customer

importance, etc. Either the number of load cases per scenario matches the calculated occurrence or each load case is weighted separately.

Cost functions

The AEB system is supposed to prevent collisions and at the same time must not brake in uncritical situations. Cost functions are needed to define the desired behavior of the AEB system. A cost function outputs costs, which measure the function performance with respect to the desired behavior. Therefore, the cost function C (Equation 1) processes the simulation output $\boldsymbol{F}(P)$. The costs are normalized. Zero costs represent the optimum and the worst case corresponds to costs of 1.

$$cost: \mathbb{R}^n \to \mathbb{R};\quad \boldsymbol{y} \mapsto C(\boldsymbol{F}(\boldsymbol{P})),\quad cost \in [0,1] \qquad \text{(Eq. 1)}$$

As stated above the goal is to balance safety performance and customer acceptance. Below we are presenting two simple ideas to measure these two goals.

Safety Performance: The *safety performance* is measured as the reduction of the impact speed (Equation 2).

$$cost_{sp} = \frac{v_{rel}(t_{coll})}{v_{Ego}(t_{brakeStartEgo})} \qquad \text{(Eq. 2)}$$

$t_{brakeStartEgo}$ denotes the time of initiating the braking and t_{coll} denotes the time of the collision. If there is no collision, the costs are obviously zero. v_{rel} states the relative velocity in direction of the ego vehicle (Equation 3)

$$v_{rel}(t) = v_{Ego}(t) - \cos(\phi(t)) * v_{obj}(t) \qquad \text{(Eq. 3)}$$

which is computed using the velocity of the ego vehicle v_{Ego}, the velocity of the target object v_{Obj} and the angle ϕ between the velocity vectors of the ego vehicle and the target object (see Figure 7).

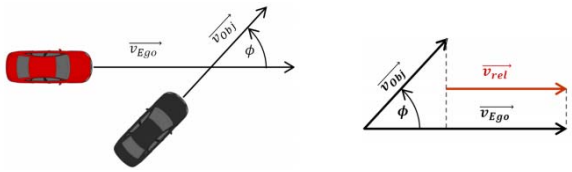

Figure 7. Relative velocity in the direction of velocity vector of the ego vehicle

The relative velocity could be negative yielding negative costs, which contradicts the normalization. A negative relative velocity occurs if the target object "escapes" from the ego vehicle or approaches the ego vehicle from behind. Hence, a collision caused by the ego vehicle is impossible. Non-negativity is assured by taking the maximum value of the relative velocity and zero. Assuming that the driver of the target object is a wrong way driver, the relative velocity is larger than the velocity of the ego vehicle giving costs greater than 1. However, the ego vehicle can reduce its own speed to standstill at best. To account for this, the minimum of the relative velocity and the ego speed is taken. Incorporating these two considerations gives the final cost function (Equation 4) for the *safety performance*:

$$cost_{sp} = \frac{\min(\max(v_{rel}(t_{coll}), 0), v_{Ego}(t_{coll}))}{v_{Ego}(t_{brakeStartEgo})} \qquad \text{(Eq. 4)}$$

Customer acceptance: The *customer acceptance* in Equation 5 is composed of two sub-cost functions "brake profile"($cost_a$) and "distance" ($cost_d$) .

$$cost_{ca} = w_a * cost_a + w_d * cost_d \qquad \text{(Eq. 5)}$$

w_a and w_d denote the weights of the sub-cost functions. Since the customer acceptance shall be normalized, it must hold that $w_a + w_d = 1$ and that all sub-cost functions are normalized.

The sub-cost function *brake profile* (Equation 6)

$$\text{cost}_a = c * dt \sum_{i=1}^{n} -a_{Ego}(t_i) * w(t_i, t_{ua}) \qquad \text{(Eq. 6)}$$

evaluates the time when the AEB is initiated and the strength of the deceleration with the time step dt, the acceleration of the ego vehicle a_{Ego} and a weighting function w, which takes as arguments the current time step t_i and the instant of time of unavoidableness t_{ua}. The time of unavoidableness is defined as the time instance of the last braking and/or steering maneuver to avoid an impending collision. The coefficient c is used to normalize the sub-cost function with the ego velocity at the first time of braking. The weighting function rates the start of deceleration as a function of time and can be modelled as a sigmoidal membership function (see Figure 8). This function is zero for start of braking after t_{ua}. Any braking after this point of time does not produce any costs. Going backwards in time, the function increases smoothly to 1 and prior to that a braking action would be fully rated. Apart from the shown behavior, the weighting function could be also dependent on the ego velocity and other characteristics.

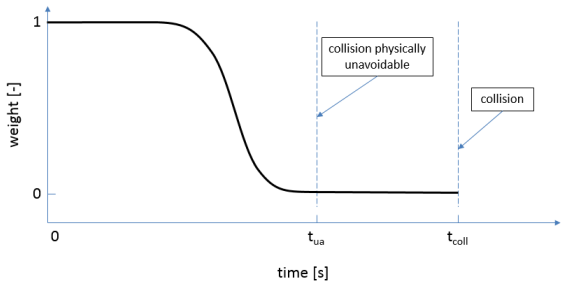

Figure 8. Weighting function depending on time of braking

The sub-cost function *distance* evaluates the minimum distance between the ego vehicle and the most critical object (see Figure 9). It is unwanted to stop too far away from the critical object, but at the same time not too close either. If the minimum distance equals a certain value defined as the optimum distance ($dist_{opt}$), the costs are zero. Depending on whether the AEB is open-loop or closed-loop controlled, this function is useful or not.

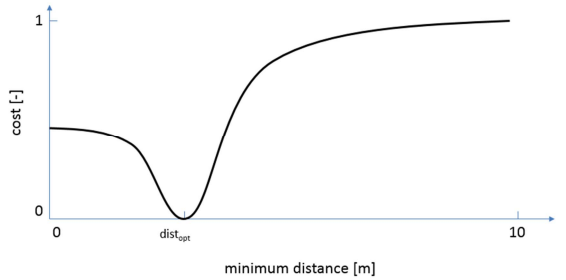

Figure 9. Weighting function depending on time of braking

RESULTS

This chapter shows an analysis of the presented optimization methods, exemplary results from the optimization and the validation of some simulation results with vehicle tests.

Analysis of presented optimization methods

In the chapter "Methodical approach" three different optimization methods were presented: direct, metamodel-based and hybrid optimization. These methods were investigated and compared in terms of their performance.

The optimization setup looks as follows. All Euro NCAP 2016 Car-to-Car and Car-to-Pedestrian situations represent the load cases, which give in total 74 load cases. The Euro NCAP load cases are taken because they are well known by a broad audience. Note that only load cases with collisions were used in the optimization. The optimization addresses in total 18 parameters which are responsible for triggering the AEB in a specific velocity area.

Metamodel-based optimization: In a first step, the parameters are sampled 700 times with a Latin hypercube sampling [9]. The cost functions *safety performance* and *customer acceptance* are applied on the samples. A Kriging, radial basis function (RBF), quadratic least squares and neuronal network metamodel are built for each cost function. Except for the Kriging metamodel, all other metamodels yield negative costs for the *safety performance* and *customer acceptance* after an optimization with the ClearVu global multi-objective optimizer [10] (see Figure 10 as an example).

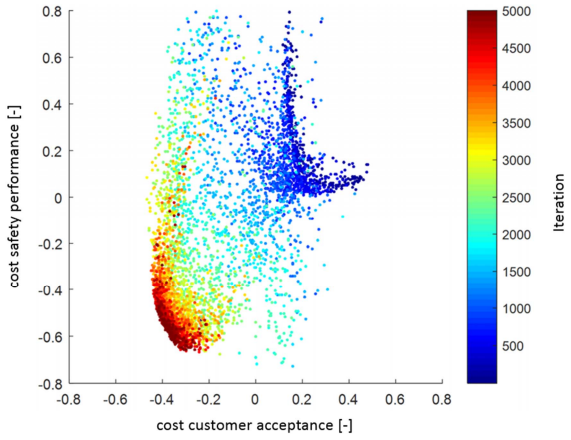

Figure 10. Negative costs for metamodel-based optimization on neural network metamodel

The optimization based on the Kriging metamodel gives plausible results (see Figure 11). The red circle shows the optimal parameter set where the *customer acceptance* and *safety performance* are weighted each with a weight of 0.5.

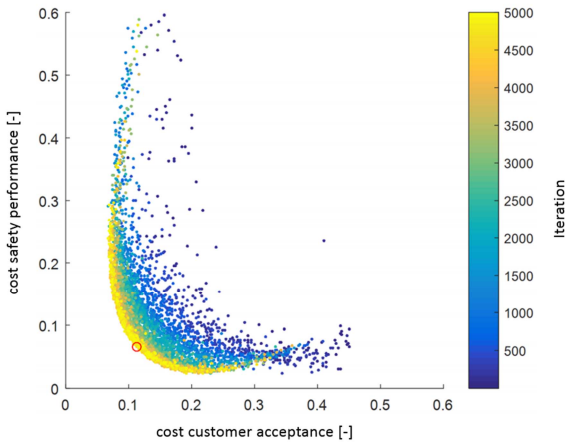

Figure 11. Metamodel-based optimization on Kriging metamodel

Table 1 shows a validation of this specific parameter set with the simulation model. The value of the metamodel is compared to the value from the actual simulation conducted with *rateEFFECT*. Especially the costs of customer acceptance differ from the metamodel to the actual simulation model with 0.049 which corresponds to an error of about 5 percent.

Table 1.

	Metamodel	Validation on simulation model	Delta
Cost safety performance [-]	0.117	0.131	0.014
Cost customer acceptance [-]	0.061	0.110	0.049

Direct and hybrid optimizers: In the following, the direct and hybrid optimization methods are investigated with a few methods available in *Optimus*. Figure 12 shows the convergence of the investigated optimizers with regard to the total cost (Equation 7) which is defined as

$$cost_{total} = 0.5 * cost_{sp} + 0.5 * cost_{ca} \qquad \text{(Eq. 7)}$$

The combination of the cost functions *safety performance* and *customer acceptance* is required to convert the multi-objective optimization (MOO) into a single–objective optimization (SOO) since the optimizers are all single-objective optimizers with the exception of the ClearVu Global Optimizer. Besides, the number of evaluations of the simulation model is limited to 700 in order to obtain results in a convenient computation time of about 9 hours on the used workstation.

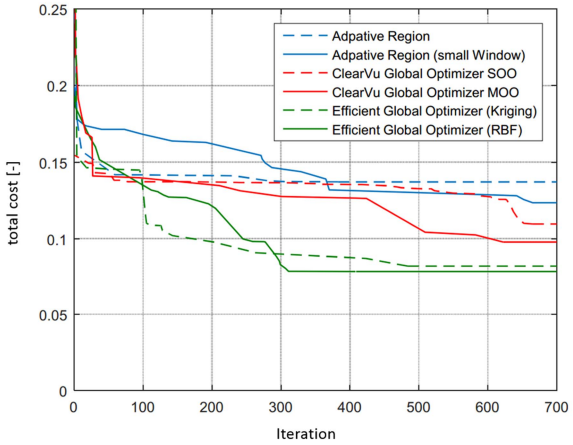

Figure 12. Convergence of different optimizers

The hybrid optimizer "Adaptive Region" [11] performs worst, regardless of the size of the initial search window. The performance of the ClearVu Global Optimizers, both SOO and MOO, range between the Adaptive Region and the Efficient Global Optimizer . The ClearVu Global Optimizer is a direct optimizer. The hybrid optimizer Efficient Global Optimizer (RBF) [10] performed best and converges after only approximately 300 iterations. The additions Kriging or RBF in brackets describe the internally used metamodels during the optimization process.

Exemplary results from the optimization

The exemplary results again are from the same optimization setup as before. 74 Euro NCAP load cases from 2016 and 700 iterations. 7.5 hours computation time on a workstation with 8 cores and 51.800 simulations were needed to obtain the optimization result. Figure 13 shows a pareto plot with both cost functions *safety performance* and *customer acceptance*. Additional information is added into the plot for two specific parameter sets $P_{opt,1}$ and $P_{opt,2}$ on the pareto front (red dotted). For reasons of clarity, only the Euro NCAP points are displayed and the average and maximum initiating time of the AEB before the time of unavoidableness. Of course, many more values could be plotted as well. $P_{opt,1}$ shows a better customer acceptance than $P_{opt,2}$ but at the same time worse safety performance. The average time of first braking before t_{ua} is 0.26 s later with $P_{opt,2}$ compared to $P_{opt,1}$.

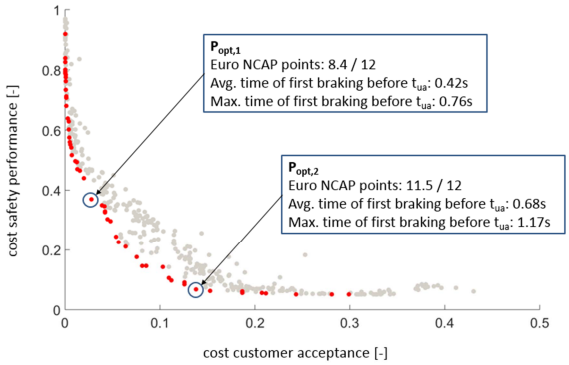

Figure 13. Pareto plot with results from the optimization

Additional analysis with both chosen parameter sets $P_{opt,1}$ and $P_{opt,2}$ are made with load cases from GIDAS (only collision load cases). For this exemplary analysis, the load cases are selected depending on the following criteria:

- object class of the collision opponent (vehicle or pedestrian)
- velocity vectors point in the same direction with a maximum deviation of 45° (only for vehicle opponents)
- collision opponent in the sensor field of view
- no driver braking before the collision

247 load cases with vehicle opponent and 341 load cases with pedestrian opponent fulfill the criteria.

Table 2 shows the percentage of avoided collisions and the average velocity reduction.

Table 2.

	Load cases with vehicle collision		Load cases with pedestrian collision	
	$P_{opt,1}$	$P_{opt,2}$	$P_{opt,1}$	$P_{opt,2}$
Number of simulated load cases [-]	247	247	341	341
Percentage of avoided collisions [%]	18.2	44.5	10.3	39.3
Average velocity reduction [kph]	15.5	25.0	13.9	21.8

In both vehicle and pedestrian load cases parameter set $P_{opt,2}$ obviously performs better. In load cases with vehicle collisions 26.3 % more collisions could be avoided and the velocity reduction is 9.5 kph higher. In load cases with pedestrian collisions even 29 % more collisions could be avoided and the velocity reduction is 7.9 kph higher.

Validation of simulation results with vehicle tests
Vehicle tests with an optimized parameter set P_{opt} have been conducted to validate the simulation results. Typical Car-to-Car Rear Stationary (CCRs) tests were used and the vehicle under test (VUT) and the target objects were equipped with reference instrumentation. The remaining gap between VUT and the target objects was measured and compared to the simulation results. A very good transferability of the simulation results could be found. The maximum error was 0.71 m and the average error was about 0.28 m.

DISCUSSION

This chapter is structured into two subchapters: the discussion of the specific results in this paper and the discussion of the overall approach.

Specific results
In total three optimization methods were analyzed out of which the hybrid optimization performed best. Both convergence was reached most rapidly and the costs were lowest no matter what internal metamodel was used. Despite this result a new analysis should be conducted if the AEB function changes drastically with regard to the number of parameters investigated, linearity of the model, etc.
An exemplary optimization of an AEB function was conducted. The intention of the authors is the demonstration of the whole optimization process and parameter determination, not the analysis of the single outcome.
The validation of the simulation results were conducted only in Car-to-Car Rear stationary tests with reference instrumentation. Nevertheless, in many other load cases with dynamic VUT and dynamic objects the transferability of the simulation results is still very good.

Overall approach
The presented SIL-based approach could also be done HIL-based, but would then be much slower. The advantages of this SIL-based approach are:

- Hazard-free
- Low time exposure
- High test coverage
- Reproducability
- Complex load cases possible

But there are also challenges which have to be considered for broader future applications. The limitations of the presented methodology are primarily given by the quality of the embedded vehicle and environment simulation model. The current simulation model gives ideal 2D algorithm input signals sufficient for good weather conditions and stationary vehicle maneuvers with little vehicle yaw, pitch and roll movement. For realistic simulation results even under complicated driving or perception conditions more sophisticated vehicle and environment sensor models are required. Nevertheless a potential analysis of predictive algorithms can be done using ideal sensor signals in the optimization process. The degradation of effectivity and robustness by the artificial worsening of sensor input signals can be analyzed afterwards to incrementally separate algorithm and sensor effects. Of course in terms of future piloted driving functions real vehicle validation will additionally be necessary to a large extent both to generate validated simulation models and to reveal unconsidered statistical effects in the simulation models.

CONCLUSION

This paper presents a simulation-based method automizing the application, performance evaluation and testing of predictive safety functions using the example of current AEB systems. The approach addresses the growing scenario complexity and the increasing performance requirements with several intended uses along the function development process. The approach overall aims at reducing specification, application and test costs by continuous simulation along the whole development process. It provides a valuable contribution to the design and testing of safe assisted and piloted driving functions.

REFERENCES

[1] Verband der Automobilindustrie e. V. "Automatisierung – Von Fahrerassistenzsystemen zum automatisierten Fahren.", Potsdam: Brandenburgische Universitätsdruckerei und Verlagsgesellschaft, 2015

[2] U.S. Department of Transportation. "Systems Engineering for Intelligent Transportation Systems – An Introduction for Transportation Professionals", Publication No. FHWA-HOP-07-069, Washington, D. C., 2007

[3] Noesis Solutions, Optimus Rev 10.17 - User Manual, Leuven: Noesis Solutions, 2015.

[4] J. M. Wille, M. Zatloukal, rateEFFECT – Effectiveness evaluation of active safety systems, In proceedings of 5th Internationa Conference on ESAR, Hannover, 2012

[5] A.-B. Ryberg, R. D. Bäckryd and L. Nilsson, Metamodel-Based Multidisciplinary Design Optimization for Automotive Applications, Linköping: Linköping University, 2012

[6] D. R. Jones, M. Schonlau and W. J. Welch, Efficient Global Optimization of Expensive Black-Box Functions, Journal of Global Optimization, 1998

[7] M. Wisch, P. Seiniger, C. Pastor, M. Edwards, C. Visvikis, C. Reeves, Scenarios and weighting factors for pre-crash assessment of Public integrated pedestrian safety systems, AsPeCSS Report, 2013

[8] O. Op den Camp, A. Ranjabar, J. Uittenbogaard, E. Rosen, R. Frederiksson, S. de Hair, Overview of main accident scenarios in car-to-cyclist accidents for use in AEB-system test protocol, Haus der Technik "Fahrerassistenz und Aktive Sicherheit", Essen, 2015

[9] K. Siebertz, D. v. Bebber and T. Hochkirchen, Statische Versuchsplanung: Design of Experiments (DoE), Heidelberg: Springer, 2010

[10] Noesis Solutions, Optimus Theoretical Background, Leuven: Noesis Solutions, 2015

[11] N. Stander and K. Craig, "On the Robustness of a Simple Domain Reduction Scheme for Simulation-Based Optimization," in Engineering Computations, Livermore, 2002