Validation of Automated Driving Functions

Ruben Schilling and Torsten Schultz

Abstract The validation of the development of driver assistance functions over millions of driven test kilometers finds its limits already today. The systems evolve rapidly from pure assistants over automated driving maneuvers towards fully autonomous driving. In the process an explosion of complexity is generated through interconnection and addition of sensors as well as strongly increased model complexity to represent the environmental aspects relevant for driving dynamics better and timelier. Here we would like to present an idea how an effort reduction of the validation can be realized. We propose to start the validation of intelligent systems before the road test: Features that intelligent systems use to classify the environment can selectively be varied in regards of test stimuli. The sensor state space is far easier to handle here than in the road test. Variations can be generated noticeably more efficient and goal oriented. In conclusion relevant parts of intelligent systems can be validated in driving dynamics relevant scenarios with less effort.

1 Introduction

A major source of innovation for today's vehicles is the addition of advanced driver assistance systems (ADAS) and their improvement over generations. In the big picture these systems develop from pure assistants over automated driving maneuvers towards fully automated driving. Fully automated driving is a scenario that provides whole new opportunities and drastic change for people's daily lives.

In the validation of driver assistance systems often 100,000 km and more of road tests scenes are collected for a single system. Driving maneuvers and situations are usually specified at the start of the development and the tests are recorded to

R. Schilling (✉) · T. Schultz
Berner & Mattner Systemtechnik GmbH, Gutenbergstr. 15, Berlin 10587, Germany
e-mail: Ruben.Schilling@berner-mattner.com

T. Schultz
e-mail: Torsten.Schultz@berner-mattner.com

C. Gühmann et al. (eds.), *Simulation and Testing for Vehicle Technology*,
DOI 10.1007/978-3-319-32345-9_25

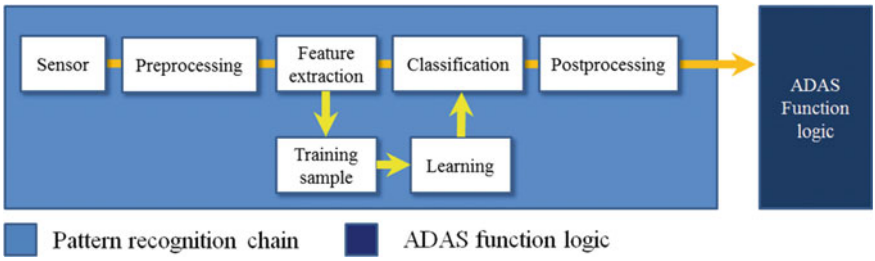

Fig. 1 The general architecture of ADAS or intelligent system can often be depicted by the combination of a pattern recognition chain with an ADAS specific function logic such as zones, warning or concepts utilizing actuating elements

produce a repeatable benchmark suite for the monitoring of the maturity grade of the system as it is developed (Fig. 1).

Machine learning techniques are methods that pave the way towards autonomous driving. The ability to learn from limited data generalized recognition concepts, that work in an otherwise unexplored world is what makes these technique very powerful concepts for intelligent or even autonomous vehicles. Yet the concept of generalization is typically phrased in terms of an error. We give an introduction to errors of machine learning models in the next section.

The general architecture of ADAS or intelligent system can often be depicted by the combination of a pattern recognition chain [1] that sends its recognized classes to the ADAS function logic. The pattern recognition chain starts with the raw sensor data. This is followed by a typically sensor specific preprocessing. From preprocessed data features are extracted. These can then be used to train the system offline or in the live system to feed a trained classifier. The classes of the classifier are then often post processed using additional heuristic knowledge to filter false positives. The final classes are then passed to the ADAS function logic. Typical example of ADAS function logics are warning zones, warn concepts (timings and warning levels) combined with specific conditions.

In practical development projects often the test focus lies on vehicle maneuvers and scenarios relevant for the ADAS function, such as e.g. take over maneuvers. Systematic testing of the intelligent system's internals is rarely employed. This leads to a high risk, as the developers do not know if the system generalizes well, once it is deployed to customers. Typically this puts a lot of pressure on people who participate in road tests to manually recognize all abnormal behavior during the road tests. When driving 100,000 km of road tests or more, keeping track of all issues and their potential interactions becomes an error prone process. Errors are usually put into a database and fixed in the progress. If the encountered errors are relevant in the sense of, that they are representative of situations that will occur in the future use of the vehicle is hard to estimate. Furthermore during typical road tests often the same routes are driven. Each drive gives rise to variations of course, as every test drive is somehow specific. But the opportunities to stress specific aspects of the intelligent system are strongly limited as only the system as a whole can be stressed from the outside.

Hence the abstraction layer of features for intelligent systems is inefficient to stimulate when executing testing according to traditional test concepts that typically comprise hierarchical levels such as unit tests, integration tests, system tests and road tests. The level of indirection to stimulate this abstraction layer is too high to efficiently test intelligent systems here. However the variations of features that would appear under relevant driving conditions are one of the most critical factors for the performance of intelligent systems.

2 Validation of ADAS Functions Through Simulation of Features

In real ADAS series development projects the data to train systems can only be collected piece by piece, as the required kilometers of road driving are time consuming.

Collecting data to train and validate machine learning models is also generally a time consuming activity. For intelligent systems this means, that although we usually have a lot of data in total, it has to serve so many different training purposes, that for each recognition problem we do not end up in a data rich situation. This has consequences for the assessment of faults of the intelligent systems.

Generally the error of a machine learning model can be defined as in [2]: During training a machine learning model is adapted to predict the training data as good as possible. The training error is the error that remains between the models prediction and its training data. The generalization error of a machine learning model is the performance of that model on independent test data, i.e. data that has not been utilized to train the model. The generalization error is what actually matters in practice. This error reflects the promise that the system will behave well during many years of ownership of a customer and while being exposed to unseen conditions and environments.

The textbook picture (see [2]) is to split data into training, validation and if possible test data. The training data is then used to fit the models, the validation data to compare alternative models to each other before a final model is chosen and the test data to predict the final chosen models generalization capability [2]. This is an ideal case for data rich situations that unfortunately do not happen often in ADAS development projects.

Therefore we propose in this article to augment the data for testing purposes through simulation (see Fig. 2). For many scenarios it is possible to come up with simulations how certain features would behave. This enables goal specific, direct stimulation of feature variations and helps to validate the system in less time. Also in sparse data situations (e.g. when learning rare events, such as “20” speed signs in traffic sign recognition), it may prove practically impossible to stimulate the feature space through road tests. Here simulation can help to explore real world variations and test the intelligent systems performance. A low fault level is always the goal as

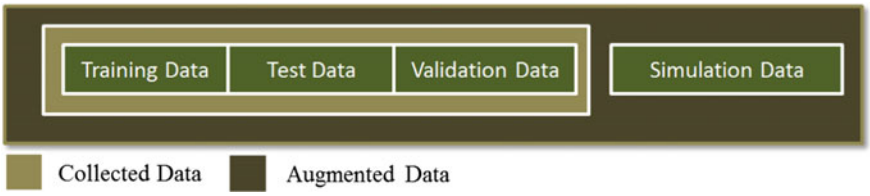

Fig. 2 Testing purposes through simulation

todays intelligent systems are either safety relevant or pure comfort functions, that shouldn't negatively affect customer experience.

As an example consider said "20" speed sign. Instead of trying to sample this sign under various conditions we can simulate the sign (see Fig. 3). Here we take the original sign (left), apply aspect ratio changes, motion blur, white balance adjustments or a combination of these effects (right). This corresponds directly to real world scenarios that could be encountered. The classification tree method [3] is one typical, systematic approach to generate test cases and structure the problem. The sensor space of a gray image sensor has $256^{1280*720}$ states. The classification tree in (Fig. 4) requires two test cases for minimal coverage. The classifications in the classification tree correspond directly the relevant scenarios and can be simulated as in Fig. 3.

The simulation data can be rich due to the cheap cost to produce them. Testing with simulated data enables predictions if the developed intelligent system can generalize from the sparse training data in richer, realistic scenarios.

As described above usually the individual components of the pattern recognition chain of intelligent systems are not systematically tested for faulty behavior. With this approach it is possible to do so in a systematic way.

It is often possible to find realistic scenarios and write a sufficient simulation to replicate their effects on relevant features. Above we show exemplary a rare data for traffic sign recognition ("20" speed limit), that is hard to sample in the real world. In the above example we augment the test data by simulating aspect ratios, motion blur, white balance issues or a combination of these effects. If we were to require the sampling of all these scenarios on the road this would prove impossible given the time constraints of real projects. The simulation itself on the other side was straightforward to do and could be easily extended and refined to cover more

Fig. 3 Simulation of relevant scenarios; from left to right: Original rare sample speed sign; Change of aspect ratio; (Motion) blur; White balance change; Aspect ratio, blur and white balance changes combined

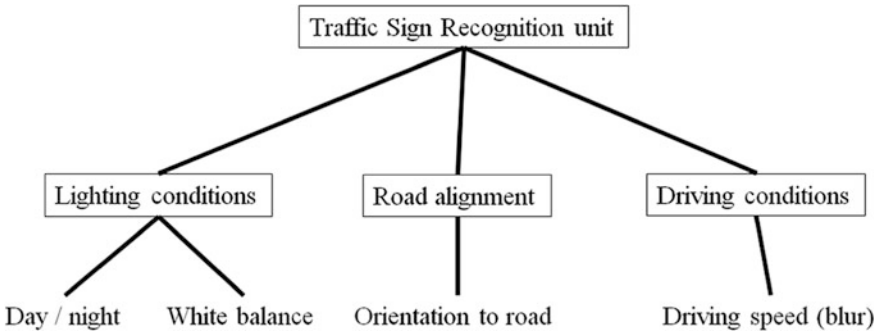

Fig. 4 A classification tree (without concrete test cases) for some realistic scenarios considered here

scenarios. Test cases can then be generated by varying the simulation parameters using standard test methodology, such as limit value checking or generating representatives.

3 Conclusion

In this article we proposed an alternative way to test intelligent systems in real world development projects. We exemplified how simulations are often possible with low effort and how they cover a vast set of test cases. We showed, that this can be a helpful technique to explore an otherwise complex sensor space and to provide an assessment method for situations that are hard to sample. Furthermore we showed, that it is possible to naturally employ standard testing techniques to help generating test cases, e.g. to cover combinatorial testing needs. We believe this is an opportunity already for today's projects to improve the real world quality of intelligent systems and provides a method to help dealing with increasing complexity of these systems in their evolution towards autonomous driving. We believe, that for future development, when complexity of the systems rises and their decision playground is largely increased it will be a necessity to come up with validation approaches similar to the one we outlined here.

References

1. Theodoridis, S., Koutrumbas, K.: Pattern Recognition, 3rd edn. Academic Press, London (2006)
2. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, 2nd edn. Springer Series in Statistics (2009)
3. Grochtmann, M., Grimm, K.: Classification trees for partition testing. Softw. Test. Verification Reliab. **3**(2), 63–82 (1993)