Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 49-15 (2016) 038–043

Testing Automated Vehicles Against Actuator Inaccuracies in a Large State Space

Pascal Minnerup [*] **Alois Knoll** [**]

[*] *fortiss, An-Institut der Technischen Universität München, Germany*
[**] *Robotics and Embedded Systems, Technische Universität München, Germany*

Abstract: Inaccurate actuator responses affect the behavior of an autonomous driving system. A disadvantageous combination of such inaccuracies might lead to a collision, but is hard to test in advance due to the exponentially large number of possible combinations. This paper introduces STARVEC, a tool to test autonomous driving systems for undesired behaviors in the presence of sensor and actuator inaccuracies in a simulation environment. It stores intermediate states of the simulation and uses these states to efficiently explore the space of possible behaviors. Each step continues with the execution of the state with the highest distance to its neighbors. Thus, the potentially large space of reachable states is covered fast and increasingly dense. The approach is applied to an autonomous parking system with inaccurate actuators and its performance is compared to a Monte-Carlo algorithm and a previous prototype.

Keywords: Fault Detection, Diagnosis, Tolerance and Removal; Path Planning; Advanced Driver Assistance Systems

1. INTRODUCTION

Automated driving functions have to ensure reliable execution in all situations for which they can be activated. Different external and internal conditions lead to nondeterministic behavior of the sensors and actuators, on which the planning and control system depends. External conditions include weather and road quality. Among the internal conditions are internal states of the brake and motor controller which are not monitored by the control algorithm. If the time and magnitude of these conditions is discretized, the number of possible combinations still grows exponentially with the length of a driving scenario. Planning and control systems usually cope with these uncertainties by adding safety margins around planned positions. A large number of tests is necessary to validate that the safety margins are sufficient. These tests have to be repeated after each change in the implementation or the parameter set that affects the planning and control system. Simulation can support such tests as a faster method to evaluate a scenario than an experiment with a physical vehicle. This paper introduces the STARVEC (Systematic Testing of Autonomous Road Vehicles Against Error Combinations) algorithm. It systematically tests sensor and actuator error combinations searching for those that lead to undesired behavior. Fig. 1 shows the tree of trajectories resulting from different actuator inaccuracies in a cross parking scenario. In addition to the prototype presented in Minnerup and Knoll (2014), the algorithm presented in this paper quickly covers the space of reachable states and delivers arbitrarily dense results. This is particularly important if the state space is very large as in long scenarios or scenarios including multiple direction changes.

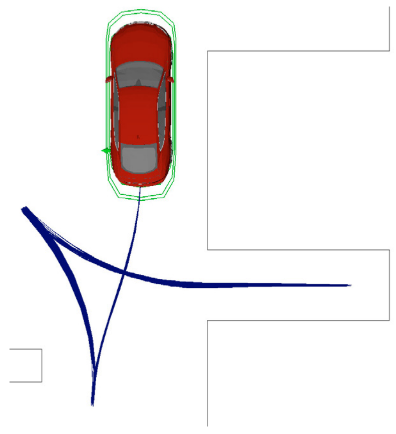

Fig. 1. Application of the STARVEC algorithm introduced in this paper to a cross parking scenario. The blue lines represent the different analyzed trajectories.

The results depend on the configured relative but not the absolute weight of the regarded dimensions. Furthermore, an actuator error pattern is applied to the system that approximates the behavior observed in physical test drives. The STARVEC system is applied to several different scenarios including multiple direction changes and its performance is compared to a Monte Carlo Simulation and the previous prototype (Minnerup and Knoll (2014)).

The main contributions of this paper are:

- A search method allowing to cover large state spaces
- A comparison to state of the art algorithms
- An actuator error model that is precise but allows efficient analysis.

2. RELATED WORK

The challenge of systematically testing an autonomous driving system can be solved by different means. The major approaches can be categorized into:

- Realistic and efficient simulation,
- systematic search for bad inputs for software systems,
- transfer of methods to the automotive domain
- and formal verification of abstract models.

2.1 Realistic and efficient simulation

Realistic and efficient simulation is recognized as important for developing autonomous driving functions by many teams. Gomez et al. (2014) support such developments including inter vehicle communication. Other teams, for example of the DARPA Urban Challenge created a simulation environment for supporting their own development project. Berlin (2007) and Bacha et al. (2008) implemented simulation environments that can be configured by DARPA scene files. Urmson et al. (2008) and Miller et al. (2008) explain how they applied simulation environments. Patz et al. (2008) describe how to adjust the architecture, such that software in the loop tests are possible. None of the teams mentions systematically introducing sensor and actuator errors to their tests or otherwise methodically testing variations of the scenarios.

For industrial applications such "Model-in-the-Loop" (MIL) tests are extended by "Hardware-in-the-Loop" (HIL) tests. In HIL tests, some parts of the test scenario are modeled by physical hardware in order to add realism. For example, Gietelink et al. (2006) describe MIL tests and focus on realistically modeling other vehicles using physical robots. In such environments, safety critical maneuvers including faults are tested. Additional sensor inaccuracies that might intensify the effect of failures are not tested systematically.

2.2 Systematic search for error patterns

The second method is to methodically search for unfavorable input combinations. There are many approaches targeting software specific problems like memory access violations. Cadar et al. (2008) use symbolic execution in order to virtually test all possible combinations of input values. In order to cover more complex problems, Groce and Joshi (2008) combine model checking with dynamic analysis. Similarly to the approach described in the present paper, they use both sound and unsound abstractions for the decision whether or not a state has been visited. This allows model checkers to efficiently test complex software.

2.3 Application of methods to the automotive domain

Some research groups also apply systematic test approaches to vehicle software. Buhler and Wegener (2004) present a method for generating test cases for a parking system by evaluating the results of each test run. Using a heuristic they try to push the test executions toward collisions by altering the starting conditions such as the shape of obstacles. This way, they find software defects occurring in a simulation without disturbances. Hes et al. (2013) regard disturbances to state variables and use rapidly exploring random trees in order to determine the worst case performance of a controller algorithm. Our paper extends a similar approach regarding also the interaction between planner and controller and modeling time dependent actuator error patterns. Ramirez et al. (2011) also regard the impact of sensor noise on driver assistance functions using search for novelty Lehman and Stanley (2008). As they do not compare intermediate states, their method only finds constant noise patterns leading to undesired behavior. In contrast, the STARVEC algorithm finds scenarios in which non constant error patterns are worse.

For general software testing there are several coverage criteria like testing all statements or finding mutations at any position in the source code. In the present paper test cases consist of actuator inaccuracy combinations and the main problem is not faulty statements, but inadequate collision prevention concepts. Hence, source code based test concepts are not directly applicable.

2.4 Formal proofs

Formal approaches as described by Althoff (2010) are an alternative to simulation. His concept computes reachable states of a system using the example of an autonomous car. The dissertation uses a simplified model of the car with some parameters being given as a probability distribution. For this model, he can prove that certain states are not reachable. The disadvantage of such a verification technique is that it considers an abstract version of the system rather than the software system itself. Plus, the system is not able to provide error combinations that lead to the contemplated states. Finally, adding new error types requires considerable effort.

3. ERROR MODEL

The behavior of a vehicle can be described by a deterministic and a non deterministic component or error model. The deterministic model describes the ideal behavior of the vehicle. The error model describes deviations from the ideal model due to the reasons listed in the introduction. In this paper deviations, in the actuators: steering and acceleration are considered as depicted in Fig. 2. The chosen error model has to match three requirements. Firstly, it has to be valid, i.e. it must be able to model all physically possible behavior that is observed in a real vehicle. Secondly, it should allow efficient analysis and model validation, which means a reduced model containing a low number of parameters. Finally, it should be precise, i.e. if possible it should only allow physically possible behavior. The most reduced model would contain only a non constant offset to the actuator input. This model would be valid as all observed actuator behavior can be described as a desired value plus a deviation. However, it would not be precise. A typical performed acceleration profile contains several time related effects like dampening and delays as depicted in Fig. 3. Due to these effects the performed actuation depends not only on the current requested value but also on the values requested in the previous moments. Modeling them as offset would require a very high maximal offset leading to a large set of behaviors possible in the simulation but not in the physical world. Therefore, this time interval is also added to the parameters of the error model and referred to as delay. The combination of a delay