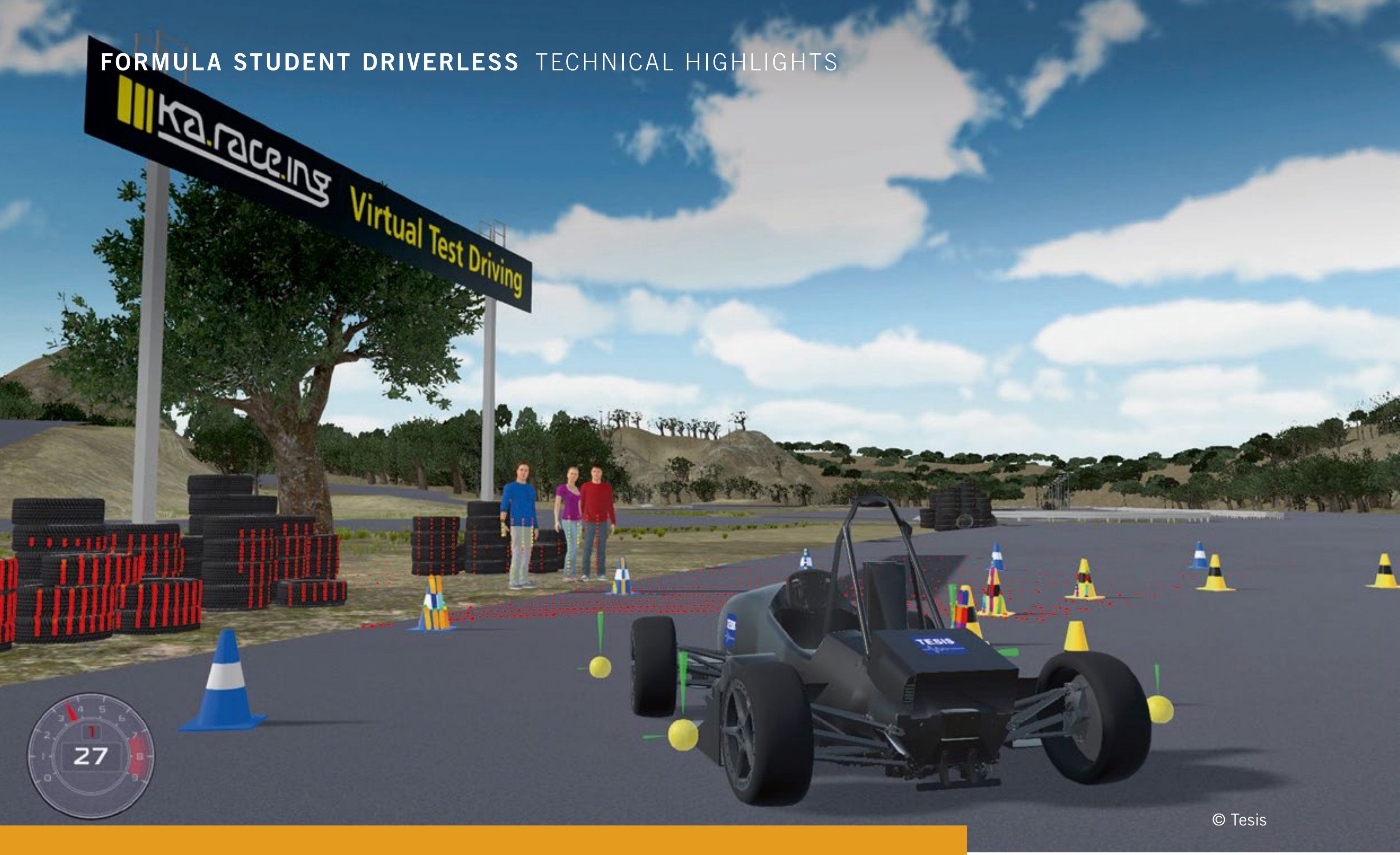


© Tesis

Virtual Test Drives for the Development of the KIT Driverless 18D

In the Driverless competition, trajectory planning and control are the most important factors – both have to be tested intensively. The racing team from the Karlsruhe Institute of Technology (KIT) uses simulations to validate the autonomous system and thus saves a lot of development time.

AUTHORS

Olaf Dünkel
studies Electrical Engineering and Information Technology (Bachelor) at the Karlsruhe Institute of Technology (KIT) and is a member of KA-RaceIng team in the 2018 season.

Lars Ohnemus
studies Mechanical Engineering (Bachelor) at the Karlsruhe Institute of Technology (KIT) and is amember of KA-RaceIng team in the 2018 season.

Dr.-Ing. Jakob Kaths
is Product Manager at Tesis GmbH in Munich (Germany).

INTRODUCTION

Autonomous vehicles offer great chances to change future mobility, however these chances come along with big challenges with regard to the development and testing of the complex vehicle system. These vehicles have to cope with critical traffic situations in a safe manner to increase the safety of future autonomous traffic. To this end, autonomous driving functions have to be developed and tested in a multitude of scenarios. Relying on real test drives is time consuming, costly and potentially dangerous. Therefore, virtual test drives in simulations are key for an efficient and safe development and test.

SIMULATION FOR THE DEVELOPMENT OF AUTONOMOUS DRIVING

Even before real hardware is available, Model-in-the-Loop simulation can

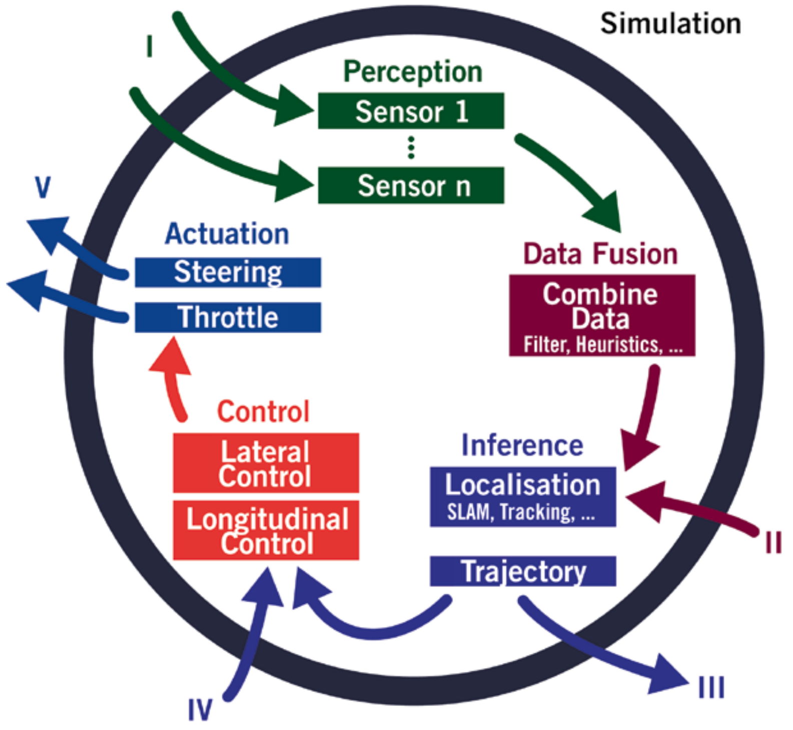


FIGURE 1 Overview of all autonomous system modules and stages of interaction between autonomous system and simulation (I-V) (© Lars Ohnemus)

support the development process. The functionality of subsystems or even the entire vehicle can be analyzed under realistic conditions, if a holistic simulation environment is available. The entire vehicle has to be modelled including its dynamics, sensors and the surrounding environment.

Very important are the interfaces to other tools. While vehicle simulation could be seen as a solitary process in the past, the interfaces to other tools gain importance when it comes to the development of autonomous driving, because data have to be exchanged bidirectionally between the simulation tool and the developed algorithms. Furthermore, the interfaces have to be flexible to enable a gradual development and test of the overall system with a modular simulation. For an early test during the development of the autonomous driving software, simulation comes into play at five stages.

In the beginning of the development, the selected sensor systems have to be embedded and the generated data needs to be processed. A test is possible with recorded data, but simulation offers a suitable alternative, **FIGURE 1** I. As such, critical scenarios are modelled to test the robustness of the developed software.

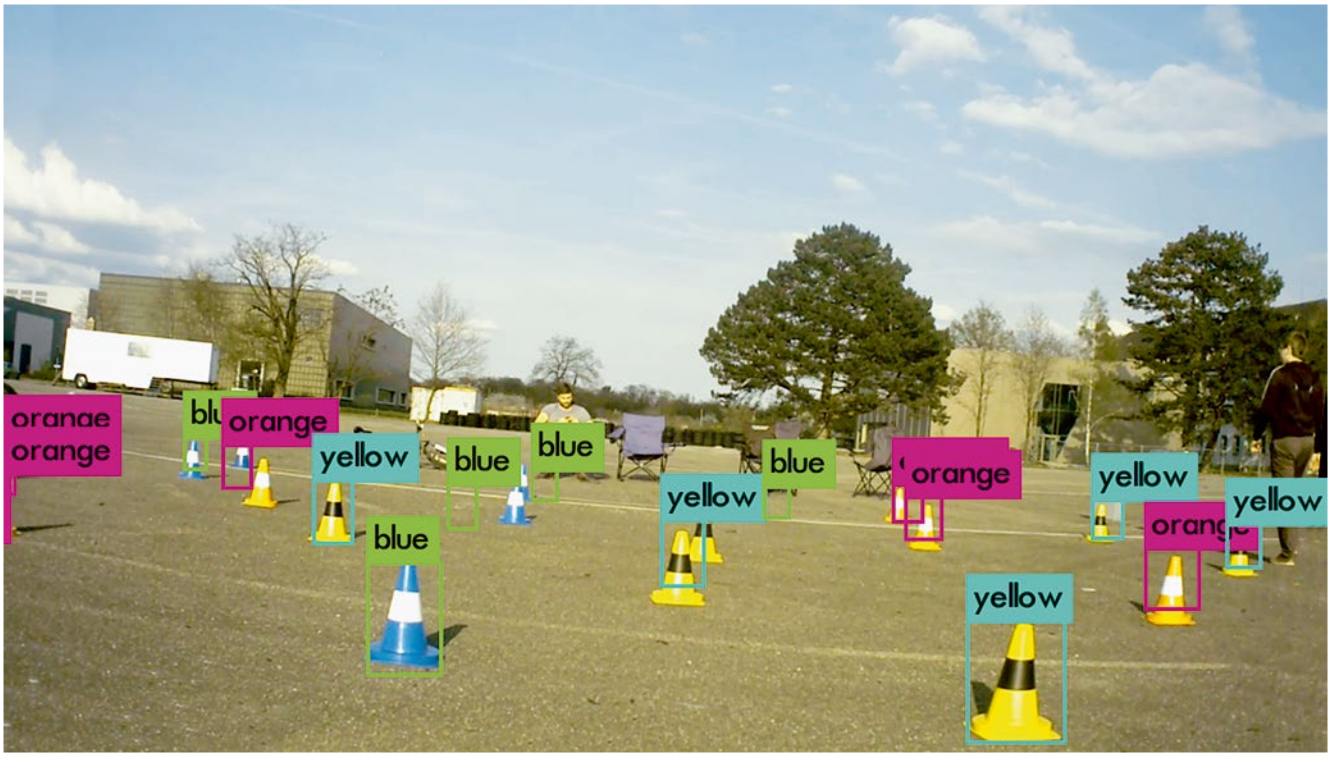
At the same time, the autonomous system can be tested on a higher level by evaluating trajectory planning and control. These two modules require intensive testing, which would have to be shifted to a later stage in the development cycle if simulation was not used, because of a high interdependency with previous inference steps. With the help of simulation, reference points (in case of Formula Student cones are used) can be measured, **FIGURE 1** II, and thus, a test of the trajectory planning is enabled. It is furthermore possible, to provide an optimal trajectory for tests of lateral and longitudinal control in early stages, **FIGURE 1** IV. A combined testing of both modules is possible to detect feedback effects and evaluate their impact, something that would not be possible with separate testing of the modules.

For the assessment of the system including potentially erroneous behavior, a selection of intermediate results of the autonomous driving software is possible, **FIGURE 1** III. This is facilitated by the usage of the Robot Operating System (ROS) and well-defined interfaces between separate modules called nodes. The calculated actuation variables, especially steering angle, wheel speed and wheel torque, are fed back into the simulation, **FIGURE 1** V. As such, the simulation includes the entire autonomous system from raw sensor data until actuation of the engines. Thanks to the holistic modelling approach, a feedback of the output variables of the developed algorithms into the system under test is possible and resulting in a closed-loop simulation.

For the autonomous system of the KIT18D, it was very valuable to test the control independent of the overall system, **FIGURE 1** IV, because various complex development tasks could be implemented and tested in a parallelized manner. Additionally, the usage of Dyna4, a tool for overall vehicle simulation, enabled a realistic test of the control, leading to a successful first real test drive. Typically, in the first real test drives, undesired oscillations of the actuating signals cause problems for the control that have to be eliminated in a lengthy parameter optimization process. A large amount of development time was saved in this season. However, the validity of the simulation could be further improved by a more complete parameter set. As such, despite the usage of simulation, later adaptations were necessary, because certain parameters that are crucial for driving dynamics were not available.

As **FIGURE 2** indicates, the realistic environment model enables the test of complex visual algorithms such as neuronal networks in simulation. The Convolutional Neural Net (CNN) was inte-

FIGURE 2 Trained CNN on real augmented data (© KA-RaceIng)



grated into the overall system and even trained with virtual camera pictures. The synthetic pictures were used to train the network for adaptivity and avoid an overfitting that could be caused by the limited set of training data.

SIMULATION SETUP

Various tools that are capable of simulating the entire vehicle with its environment were reviewed to find alternatives to Gazebo. Gazebo is widely used in the context of robotics development with ROS and is used by many Formula Student teams for testing their software. Gazebo is easy to use and well-integrated in ROS. However, it is lacking a number of crucial components that have to be implemented by user, especially driving dynamics are not reflected accurately. This is of no concern for the intended use-case of robotics development, because operation at physical limits is usually not of interest. In contrast, a driving dynamics model is key for a realistic simulation of an autonomous racing car.

In early development stages, a simple modelling approach is sufficient, however, it was decided at the start of the season that a sustainable and flexible development and simulation tool chain should be established. The tool review showed that Dyna4 by Tesis is matching the requirements best. Main reason for this decision was, besides the available ROS interface, the considerable possibilities for vehicle modelling, scenario generation and visualization. Dyna4 allows for an efficient and flexible creation and extension of the vehicle model through an intuitive graphical user interface that outperforms other tools. Another benefit is the Matlab/Simulink basis of the included models that enables the usage of powerful Matlab analyzation tools and the direct integration of Simulink controllers.

The simulation setup includes two desktop computers, the simulation runs on a Windows computer while the autonomous driving software runs on a Linux computer, **FIGURE 3**. The advantage of this setup is that the autonomous driving system can be tested virtually without special consideration of the hardware requirements of the simulation. These include for example a powerful graphics card for a performant display of the 3D environment. The connection to the ROS nodes is made with a self-implemented interface in which ROS messages are defined and sent to the autonomous driving software in the desired data format. This ROS-Node (Manage_sim) receives messages from two elements of the simulation. The reason for the relatively complex simulation lies in the structure of the Matlab/Simulink-based simulation: While Matlab contains a direct ROS interface, specific animation tools require a Windows platform. A ROS1 connection is not directly available, instead the data is sent over the data distribution service (DDS). These data can be received from ROS2 and over a network connection, ROS1 messages from Matlab and ROS2 messages from the animation are sent to the Linux computer. This computer then converts the ROS2 messages into regular ROS messages using the included bridge. Eventually, all messages can be managed and distributed from the central Manage_sim node.

The ROS1 messages from Simulink are data from the vehicle simulation, such as position, acceleration and speed values. The reason for this is that the Dyna4 vehicle model is implemented in Simulink. On the other hand, visualization data such as raw sensor data from lidar and camera, which are generated on the basis of the rendered environment, is

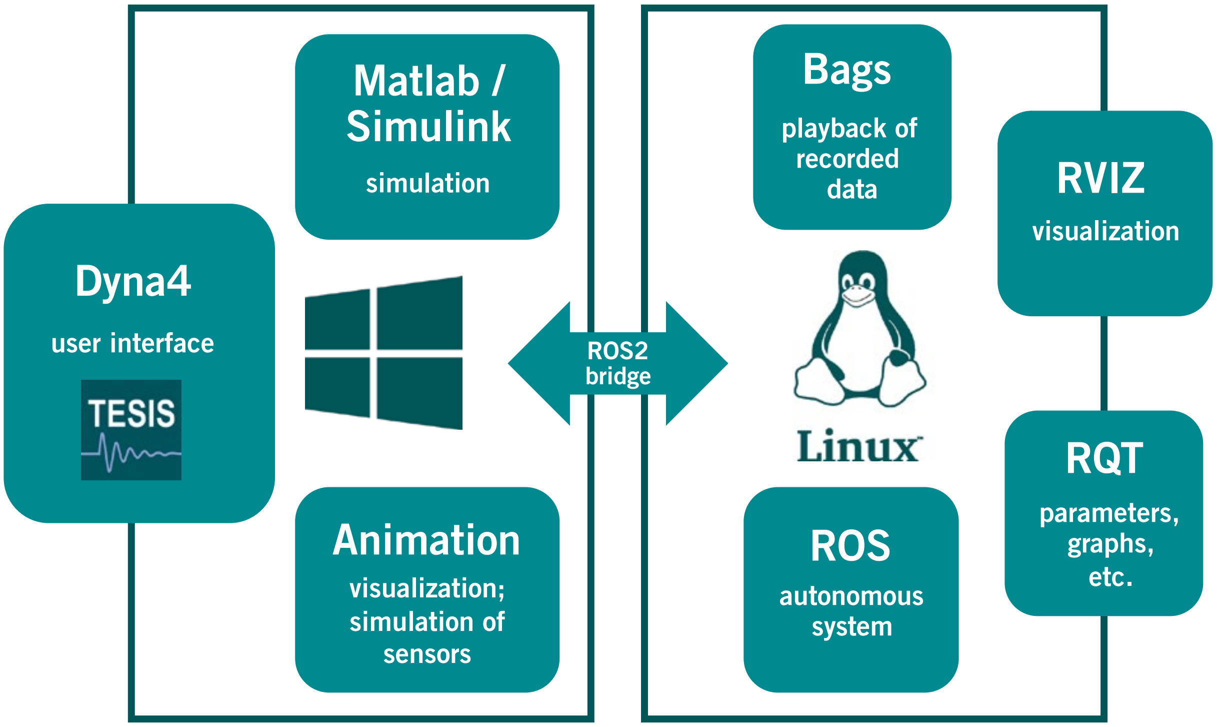


FIGURE 3 Simulation setup and integration of ROS (© Lars Ohnemus)

coming from the Dyna4 visualization tool DynaAnimation.

In addition to this complex simulation structure, our simulation process also includes two other tools. For example, a Matlab tool was implemented to optimize the position of the sensors and another tool for the intuitive creation of routes with cones.

OUTLOOK

Based on the experiences, gathered last formula student season, we want to give a brief outlook on possible improvements of simulation setup. Despite the complex structure of the overall system, a stable performance was achieved after the inital setup. Critical factors are the low robustness of the ROS2 implementation and the low efficiency of the ROS interface that Matlab/Simulink offers. The second problem can be addressed by switching to an interface that is completely based on DDS. As such, Dyna4 offers a more efficient implementation by now.

By using an self-implemented interface on s-function the first problem could also be addressed. However, the question arises as to whether such a challenge is suitable for a Formula Student team, since a pragmatic and fast working solution is required in order to validate algorithms directly before the start of the test period. Right at the start of the season the decision was made to use an existing interface that can be used right away. Therefore, first virtual test drives were already possible in February after only four months of development.

In the upcoming seasons, an increasing complexity and stronger requirements for the simulation in Formula Student Driverless are expected. Only with the usage of simulation an efficient development and a detailed testing of algorithms can be performed early in the season.