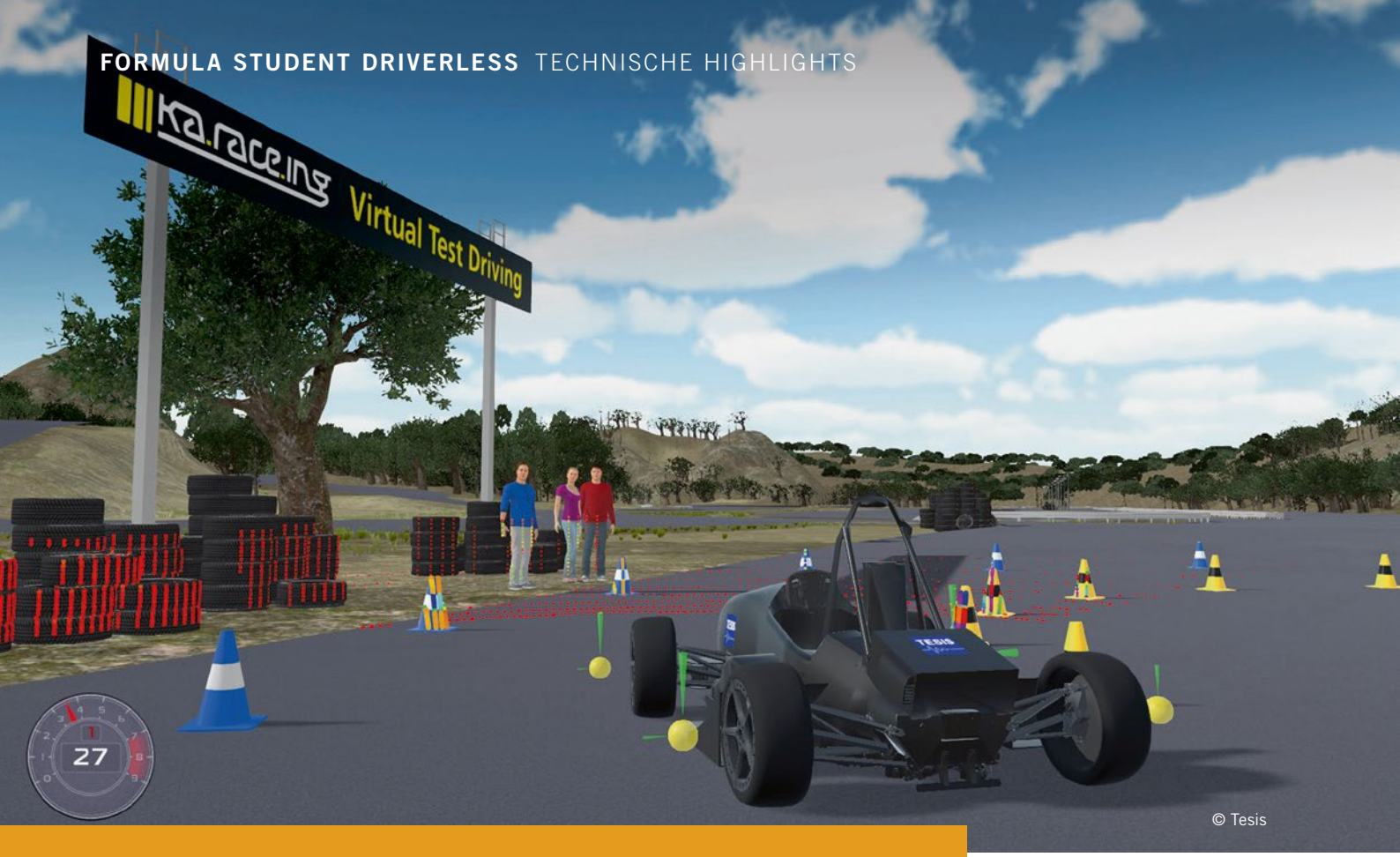


© Tesis

Virtuelle Fahrversuche in der Entwicklung des KIT Driverless 18D

AUTOREN

Olaf Dünkel
studiert Elektro- und Informationstechnik (Bachelor) am Karlsruher Institut für Technologie (KIT) und ist Mitglied im KA-RaceIng-Team in der Saison 2018.

Lars Ohnemus
studiert Maschinenbau (Bachelor) am Karlsruher Institut für Technologie (KIT) und ist Mitglied im KA-RaceIng-Team in der Saison 2018.

Dr.-Ing. Jakob Kaths
ist Produktmanager bei der Tesis GmbH in München.

Im Driverless-Wettbewerb kommt es vor allem auf die Trajektorienplanung und die Regelung an – beides muss intensiv getestet werden. Das Rennteam vom Karlsruher Institut für Technologie (KIT) nutzt Simulationen zur Validierung des autonomen Systems und spart dadurch viel Entwicklungszeit.

EINLEITUNG

Autonome Fahrzeuge bringen nicht nur große Chancen für den Mobilitätswandel mit sich, sie bergen auch enorme Herausforderungen im Hinblick auf die Entwicklung und Absicherung des komplexen Gesamtsystems. So müssen diese Fahrzeuge auch kritische Verkehrssituationen sicher beherrschen, um das Unfallrisiko im künftigen autonomen Verkehr zu senken. Hierzu müssen autonome Fahrfunktionen in einer Vielzahl von Szenarien entwickelt und getestet werden. Mit realen Testfahrten ist das sehr zeitaufwendig, kostenintensiv und potenziell gefährlich. Der Einsatz von Simulationen für virtuelle Fahrversuche ist daher unabdingbar, denn so lässt sich eine Vielzahl von Szenarien für Entwicklung und Absicherung effizient und gefahrlos untersuchen.

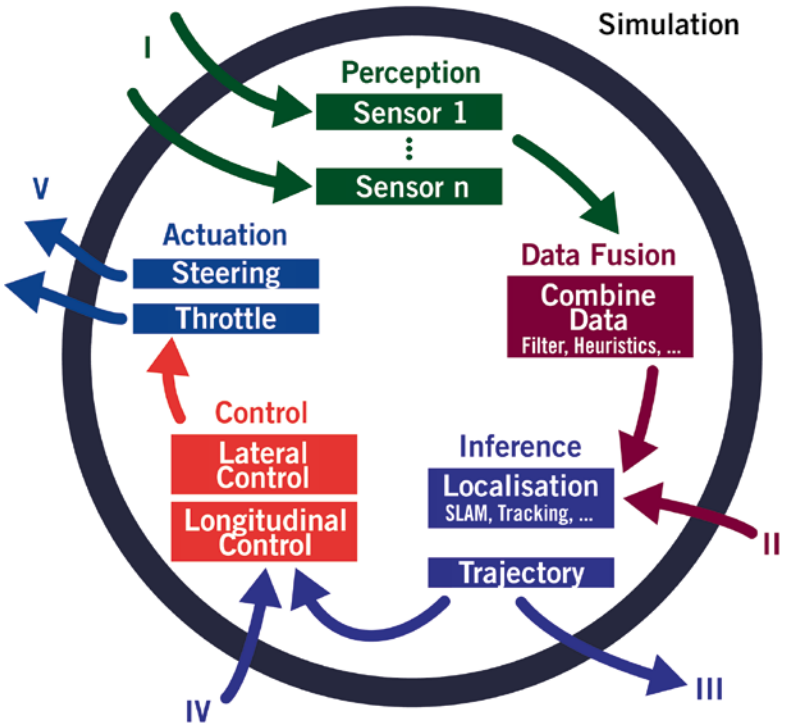


BILD 1 Modulübersicht des autonomen Systems und Ansatzpunkte der Simulation (I-V) (© Lars Ohnemus)

SIMULATION ZUR ENTWICKLUNG AUTONOMER FAHRFUNKTIONEN

Noch vor dem Test realer Hardware kann die Model-in-the-Loop-Simulation den Entwicklungsprozess unterstützen. Hier können unter realitätsnahen Bedingungen die Funktionsweise eines Teilsystems oder auch des gesamten Systems analysiert werden. Um eine derartige Simulation zu ermöglichen, ist eine ganzheitliche Simulationsumgebung erforderlich. Diese muss vor allem die Modellierung des gesamten Fahrzeugs mit dessen Dynamik und Sensoren sowie dessen Umwelt ermöglichen.

Besonders wichtig sind zudem die Schnittstellen zu anderen Tools. Während die Simulation des Gesamtfahrzeugs in der Vergangenheit oft eher abgekoppelt betrachtet werden konnte, werden die Schnittstellen bei der Simulation autonomer Fahrfunktionen immer wichtiger, da das Simulationstool Daten mit den Algorithmen bidirektional austauschen können muss.

Für die sukzessive Entwicklung und Absicherung des Systems müssen die Schnittstellen zudem flexibel verwendbar sein, da eine Simulation zunächst modular verläuft. Um die Software zum autonomen Fahren schon während der Entwicklungsphase testen zu können, setzt die Simulation an fünf Punkten des autonomen Systems an.

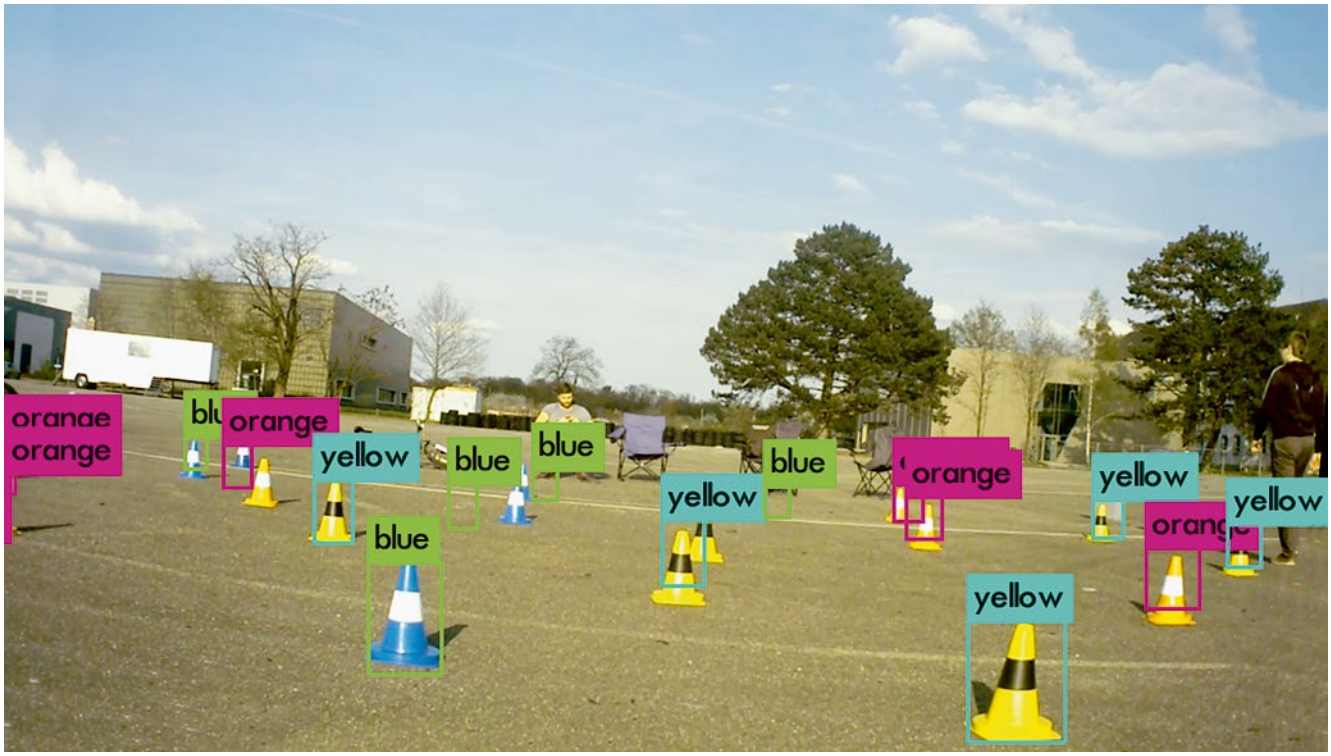
Zu Beginn der Entwicklung gilt es, die ausgewählte Sensorik einzubetten und die erlangten Daten zu verarbeiten. Das kann gut mithilfe aufgenommener Sensordaten validiert werden, aber auch simulierte Daten bieten ein probates Werkzeug, **BILD 1** I. So lassen sich etwa kritische Szenarien modellieren und die Software auf Robustheit testen. Gleichzeitig kann der High-Level-Bereich des autonomen Systems validiert werden, also die Trajektorienplanung und die Regelung – zwei Module, die sehr intensiv getestet werden müssen. Ohne die Möglichkeit zur Simulation könnte dies erst spät im Entwicklungsprozess geschehen, da eine starke Abhängigkeit von den vorhergehenden Inferenzschritten besteht. Mithilfe der Simulation können Koordinationspunkte (im Falle der Formula Student sind es Pylonen), **BILD 1** II, abgegriffen werden, wodurch die Trajektorienplanung des zu entwickelnden Systems getestet werden kann. Gleichzeitig kann die optimale Trajektorie vorgegeben werden, um bereits frühzeitig die Längs- und Querregelung zu testen, **BILD 1** III. Denkbar ist auch die Verifizierung beider Module gleichzeitig, um Rückkopplungen zu erkennen und deren Einfluss zu bewerten, was beim Testen der einzelnen Module nicht möglich wäre.

Um Fehlermechanismen und Verhalten des Systems besser einschätzen zu können, lassen sich zu jedem Zeitpunkt Teilergebnisse der Software abgreifen, **BILD 1** IV. Dank des verwendeten Robot Operating System (ROS) wird das durch die Definition klarer Schnittstellen zwischen den einzelnen Modulen erleichtert. Die erzeugten Stellgrößen der Aktorik, also vor allem Lenkwinkel, Raddrehzahl und -moment, werden wieder an die Simulation weitergegeben, **BILD 1** V. So umfasst die Simulation von der Verarbeitung der Rohdaten der Sensoren bis zur Aktuierung der Motoren das gesamte autonome System. Eine perfekte Simulation kann unter anderem für die Validierung des Mapping verwendet werden – dem Aufbau einer lokalen oder globalen Karte. Durch die ganzheitliche Systemmodellierung ist die Rückkopplung der Ausgangsgrößen der jeweiligen Algorithmen auf die Eingangsgrößen des zu simulierenden Systems in der Simulation möglich. Hier spricht man von einer Closed-loop-Simulation.

Für das autonome System des Rennwagens KIT18D war es vor allem hilfreich, die Regelung unabhängig vom Gesamtsystem testen zu können, **BILD 1** IV. So konnten verschiedene komplexe Entwicklungsschritte parallel implementiert und validiert werden. Zusätzlich konnte dank der Verwendung von Dyna4, einem Werkzeug für Gesamtfahrzeugsimulation, die Regelung bereits realitätsnah getestet werden. Dadurch war die Jungfernfahrt bereits in Ansätzen erfolgreich. In der Regel muss bei der ersten Realfahrt mit Problemen in der Regelung gerechnet werden, die vor allem durch ungewollte Schwingung der Stellgrößen entstehen und nur durch langwierige Parameteroptimierung beseitigt werden können. Hier konnte viel Entwicklungszeit gespart werden. Um die Aussagekraft der Simulation weiter zu erhöhen, müsste die Parametrierung des Fahrzeugs verbessert werden. So waren trotz Simulation noch Anpassungen notwendig, da einige fahrdynamisch relevante Parameter nicht vorlagen.

Wie in **BILD 2** dargestellt, ist es durch die realitätsnahe Umgebungsmodellie-

BILD 2 Trainiertes CNN auf augmentierten realen Daten (© KA-RaceIng)



rung möglich, auch komplexe visuelle Algorithmen wie neuronale Netze in der Simulation zu validieren und in das simulierte Gesamtsystem einzubinden. Zudem konnte die Möglichkeit genutzt werden, durch die Einstreuung von Simulationsbildern ein Convolutional Neural Network (CNN) auf Adaptivität zu trainieren, da durch die Verwendung von synthetischen Bildern einem sogenannten Overfitting auf die begrenzte Anzahl an vorhandenen Trainingsdaten entgegengewirkt werden kann.

SIMULATIONSSTRUKTUR

Ausgehend von der Notwendigkeit eines Simulationssystems, das für eine Gesamtfahrzeugsimulation inklusive dessen Umwelt geeignet ist, wurde eine ausgiebige Recherche zu diversen Tools durchgeführt, um Alternativen zum Tool Gazebo zu untersuchen. Gazebo ist im Bereich der Roboterentwicklung mit ROS weit verbreitet und wird vom Großteil der Formula-Student-Teams zur Verifikation der Software verwendet. Gazebo ist zwar ein einfach zu handhabendes Werkzeug und wegen seiner direkten Integration mit ROS sehr beliebt, dennoch muss der Entwickler viele Komponenten, die für eine realitätsnahe Simulation nötig sind, allen voran eine gezielte Modellierung der Fahrdynamik, selbst implementieren. Das ist für den ursprünglichen Zweck, das virtuelle Testen von Robotern, unkritisch, da hier in der Regel nicht an den physikalischen Grenzen des Systems operiert wird. Ganz andere Anforderungen stellt natürlich die Entwicklung eines autonom fahrenden Rennwagens, bei dem die Fahrdynamiksimulation weit über einfache physikalische Modelle hinausgehen muss, um realistische Ergebnisse zu erzielen.

In den ersten Entwicklungszyklen ist vorerst ein einfaches Fahrzeugmodell ausreichend, dennoch wurde zu Beginn der Saison festgelegt, dass die Entwicklungs- und Simulationskette nachhaltig und flexibel ausgelegt sein sollte. Nach Abschluss der Recherche erschien das Simulationstool Dyna4 von Tesis am besten geeignet. Gründe für diese Entscheidung waren, neben der vorhandenen ROS-Schnittstelle, die umfangreichen Möglichkeiten zur Fahrzeugmodellierung, Szenariengenerierung und Visualisierung der Testumgebung. Mit Dyna4 ist die Modellierung und Erweiterung eines Fahrzeugmodells über eine intuitive grafische Benutzerschnittstelle effizienter und flexibler möglich als bei anderen untersuchten Lösungen. Ein weiterer Entscheidungsgrund war die modellgetriebene Entwicklung mit Matlab/Simulink, auf dessen Basis die Dyna4-Modelle aufbauen. So konnten neben den Funktionalitäten von ROS und dessen Erweiterungen die mächtigen Visualisierungs- und Analysemethoden von Matlab genutzt und in Simulink implementierte Regler direkt eingebunden werden.

Der Hardwareaufbau für die Simulation besteht aus zwei Rechnern – einem Windows-PC, auf dem die Simulation läuft, und einem Linux-PC, auf dem die Software für das autonome System läuft, **BILD 3**. Das hat den Vorteil, dass unser System unabhängig von den Rechneranforderungen der Simulation getestet werden kann. Vor allem für die Visualisierung ist für eine realitätsnahe Umgebungssimulation eine performante Grafikkarte notwendig. Die Anbindung an die ROS-Nodes geschieht durch eine selbst implementierte Schnittstelle, in der ROS-Messages definiert und im gewünschten Datenformat an die Software weitergeleitet werden. In dieser ROS-Node (Manage_sim) werden Meldungen aus zwei verschiedenen Elementen der Simulation empfangen. Die ver-

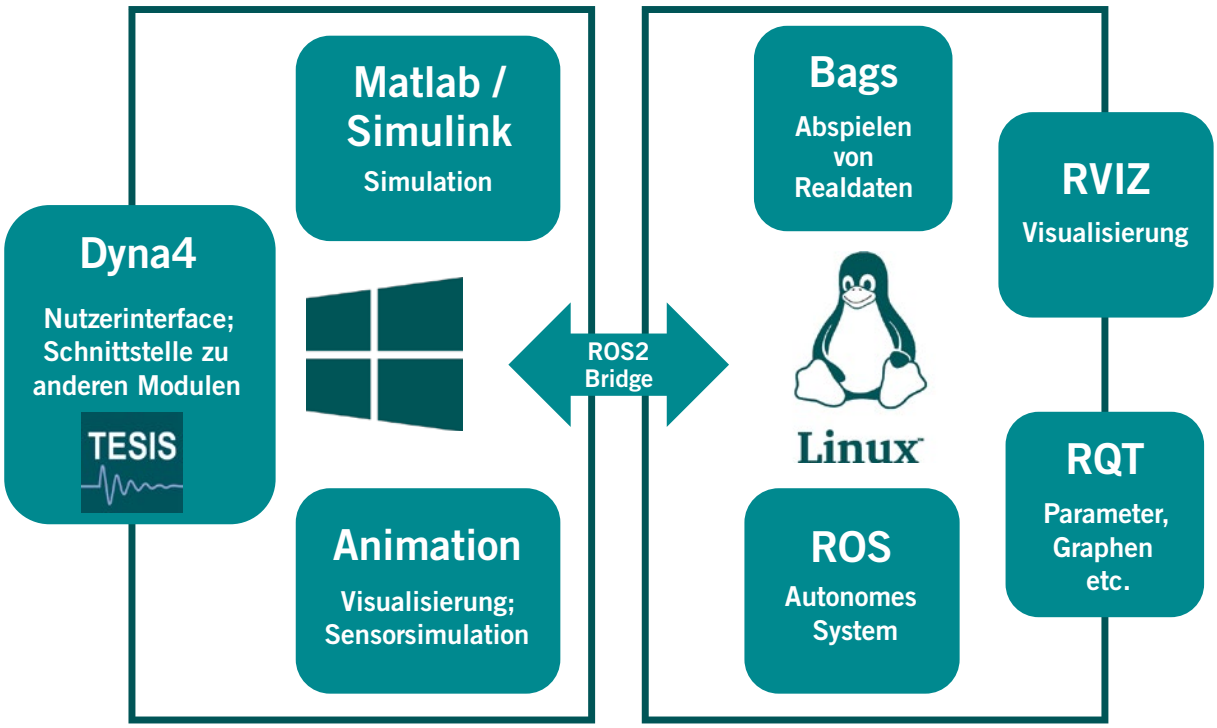


BILD 3 Aufbau der Simulation und Integration von ROS (© Lars Ohnemus)

gleichsweise hohe Komplexität der Simulation liegt an der Struktur einer Matlab/Simulink-basierten Simulation: Während Matlab über eine direkte ROS-Schnittstelle verfügt, können spezifische Animationstools oft nur auf einer Windows-Plattform betrieben werden. Eine ROS1-Anbindung liegt hier aber nicht direkt vor. Stattdessen werden die Daten als DDS (data distribution service)-Messages versendet. Diese können von ROS2 empfangen werden. Über eine Netzwerkverbindung werden dann sowohl die ROS1-Messages aus Matlab/Simulink als auch die ROS2-Messages an den Linux-PC gesendet. Auf diesem werden dann die ROS2-Messages mithilfe einer seitens ROS mitgelieferten Bridge zu gewöhnlichen ROS-Messages konvertiert. Somit können schließlich alle Messages in der Manage_sim-Node zentral verwaltet und weitergegeben werden.

Bei den ROS1-Messages aus Simulink handelt es sich um Daten aus der Fahrzeugsimulation, etwa Position sowie Beschleunigungs- und Geschwindigkeitswerte. Grund hierfür ist, dass das Dyna4-Fahrzeugmodell in Simulink implementiert ist. Visualisierungsdaten wie Rohsensordaten von Lidar und Kamera, die auf Basis der gerenderten Umgebung generiert werden, kommen hingegen aus dem zu Dyna4 zugehörigen Visualisierungstool DynaAnimation.

Neben dieser komplexen Simulationsstruktur gehören zu unserem Simulationsablauf noch zwei weitere Tools. So wurde ein Matlab-Tool für die Optimierung der Position der Sensoren implementiert und ein weiteres Tool zur intuitiven Erstellung von Strecken mit Pylonen.

AUSBLICK

Aufgrund der Erfahrungen aus der vergangenen Formula-Student-Saison soll zum Abschluss noch ein Ausblick auf mögliche Verbesserungen des Gesamtsystems gegeben werden. Trotz der komplexen Struktur des Gesamtsystems funktionierte es nach abgeschlossener Einrichtung stabil. Kritische Faktoren sind jedoch die wenig robuste ROS2-Implementierung und -Bridge sowie die ROS-Schnittstelle von Matlab/Simulink, da diese nicht sehr effizient implementiert wurde. Durch eine Schnittstelle, die vollständig auf DDS-Messages basiert, kann das zweite Problem umgangen werden. Inzwischen bietet Dyna4 eine alternative Implementierung mit höherer Effizienz.

Durch eine auf s-functions selbst implementierte Schnittstelle kann auch der erste Kritikpunkt umgangen werden. Allerdings stellt sich die Frage, ob eine derartige Herausforderung für ein Formula-Student-Team geeignet ist, da hier eher eine pragmatische und schnell funktionierende Lösung erforderlich ist, um direkt vor Beginn der Testzeit Algorithmen validieren zu können. Bereits zu Beginn der Saison fiel die Entscheidung für eine bestehende Schnittstelle, die schnell verwendet werden konnte. So wurden erste virtuelle Testfahrten bereits im Februar nach vier Monaten Entwicklungszeit durchgeführt.

In den kommenden Saisons ist mit einer Zunahme der Komplexität und der Anforderungen an die Simulation in der Formula Student Driverless zu rechnen. Nur so können eine effiziente Entwicklung und ein ausdifferenziertes Testen der Algorithmen schon frühzeitig in der Saison erfolgen.