Robert Kastner, BSc

Modellbildung und Simulation eines kooperativen Stauassistenten über ein Co-Simulationsframework

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur(in)

Masterstudium Maschinenbau

eingereicht an der

Technischen Universität Graz

Betreuer

Assoc.Prof. Dipl.-Ing. Dr.techn., Arno Eichberger

Fogh-lis. Lis. Ph.D., Sajjad Samiee

Institut für Fahrzeugtechnik
Member of [FSI]

Graz, Oktober 2017

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich im Rahmen dieser Masterarbeit und während des Studiums direkt oder indirekt unterstützt haben.

Bedanken möchte ich mich bei Dr. Arno Eichberger und Ph.D. Sajjad Samiee für die Betreuung und Unterstützung während dieser Arbeit. Außerdem möchte ich Allessandro Colla, Takayuki Miyata und Hannes Schneider von der AVL List GmbH für die technische Unterstützung und gute Zusammenarbeit danken.

Ein großes Dankeschön gilt meiner Familie und insbesondere Eltern, die mir durch ihre Unterstützung diese Ausbildung ermöglicht haben. Abschließend gilt ein spezieller Dank meinen Freunden und Mitstudenten, welche mich während arbeitsreichen Monaten motiviert und zu einer schönen und abwechslungsreichen Studienzeit beigetragen haben.

Danke!

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....
(Unterschrift)

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
(date) (signature)

Abstract

In the course of the European research project (ENABLE-S3), cooridnated by the AVL List GmbH, a Traffic Jam Assist (TJA) shall be developed. This TJA is supposed to guide a vehicle autonomated, safely, economically and therefore as comfortably as possible for the human driver, through a driving situation with a high traffic density and low speeds. This master thesis describes the modelling and simulation of a cooperative TJA via a co-simulation framework. The simulation supports the physical verification and validation of driver assistance systems. The software Model.CONNECT delivers the framework for the co-simulation of the TJA. The vehicle dynamics with the input parameters accelerator- and brake pedal position and steering angle is calculated by the software VSM (Vehicle Simulation). The software VTD (Virtual Test Drive) simulates the environment, i.e. infrastrucure and traffic and visualises the co-simulation. The controllers for the automated driving are programmed in MATLAB/Simulink. At first, the co-Simulation is set up. As a next step a model that consists of VTD, VSM and the ACC (Adaptive Cruise Control) and LKA (Lane Keeping Assist) controllers is developed. This model is simulated with two test scenarios. The simulation results are compared to on-road test runs with compareable scenarios. Afterwards a TJA-model with the platooning function is implemented and verified with the scenarios defined in the research project.

Kurzfassung

Im Rahmen des europäischen Forschungsprojekts (ENABLE-S3), koordiniert von der AVL List GmbH, soll ein Stauassistent (engl. Traffic Jam Assist, TJA) mit Platooningfunktion entwickelt werden. Mit diesem Stauassistenten soll es ermöglicht werden, ein Fahrzeug automatisiert, sicher, ökonomisch und dadurch für den Fahrer möglichst komfortabel, durch Fahrsituationen mit hohem Verkehrsaufkommen und geringen Geschwindigkeiten zu navigieren. In dieser Masterarbeit wird die Modellbildung und Simulation eines kooperativen Stauassistenten über ein Co-Simulationsframework beschrieben. Die Simulation unterstützt die physische Validierung und Verifizierung von Fahrerassistenzsystemen. Mit der Software Model.CONNECT wird die Co-Simulationsumgebung für den Stauassistenten bereitgestellt. Die Fahrzeugdynamik mit den Eingangsparametern Gas- und Bremspedalstellung sowie Lenkradwinkel wird mit der Software VSM (Vehicle Simulation) berechnet. Die Software VTD (Virtual Test Drive) simuliert die statische und dynamische Umgebung wie z.B. Infrastruktur und Verkehr und visualisiert die Co-Simulation. Die Regler für automatisiertes Fahren werden in MATLAB/Simulink programmiert. Zunächst wird die Co-Simulation aufgebaut. Als nächsten Schritt wird ein Modell bestehend aus VTD, VSM und den ACC (Adaptive Cruise Control)- bzw. LKA (Lane Keeping Assist)-Reglern aufgebaut. Dieses Modell wird mit mit zwei Testszenarien simuliert und mit Messfahrten vergleichbarer Testszenarien verifiziert. Anschließend wird auf Basis des ersten Modells ein Modell für den TJA (Traffic Jam Assist) bzw. die Platooningfunktion implementiert und mit den im Rahmen des Forschungsprojekts definierten Szenarien verifiziert.

Inhaltsverzeichnis

Abkürzungen

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
BASt	Bundesanstalt für Straßenwesen
CC	Cruise Control
CIL	Cut Into Lane
ENABLE-S3	**E**uropean I**n**iti**a**tive to Ena**b**le Validation for Highly Automat**e**d **S**afe and **S**ecure **S**ystems
FMI	Functional Mockup Interface
FMU	Functional Mockup Unit
GUI	Graphical User Interface
DARPA	Defense Advanced Research Projects Agency
LDACC	Lane Dependant Adaptive Cruise Control (spurabhängiges Sensormodell)
LDW	Lane Departure Warning
LKA	Lane Keeping Assistant
OEM	Original Equipment Manufacturer
PID Regler	Proportional–Integral–Derivative Regler
SAE	Society of Automotive Engineers
TJA	Traffic Jam Assistant
TCP	Transmission Control Protocol
TCL	Target Changes Lane
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VSM	Vehicle Dynamics Simulation
VTD	Virtual Test Drive

Symbole

Variablen

$\boldsymbol{s}$	interner Status eines Elements der Co-Simulation
ΔT	Makrostep der Co-Simulation
T	Zeitpunkt der Co-Simulation
Δt	Mikrostep der Co-Simulation
t	Startzeit eines Mikrosteps
u_t	Eingangswerte eines Elements der Co-Simlation zum Zeitpunkt t
y_{t+1}	Ausgangswert eines Elements nach einem Zeitschritt der Co-Simulation
$\Delta s_{x,min}$	Mindestabstand zum Target-Fahrzeug in x-Richtung
$\Delta s_{x,t}$	Abstand zum Target-Fahrzeug in x-Richtung
Δt_T	Zeitabstand zum Target-Fahrzeug
$\Delta v_{x,t}$	Geschwindigkeitsdifferenz Target-Fahrzeug zu Ego-Fahrzeug
$v_{x,t}$	Geschwindigkeit des Target-Fahrzeugs
Δt_{PL}	Zeitabstand zum Platooning-Leader
$v_{x,e}$	Geschwindigkeit des Ego-Fahrzeugs
$s_{x,PL}$	Abstand zum Platooning-Leader in x-Richtung

1. Einleitung

Der Entwicklung von Fahrerassistenzsystemen kommt neben der Reduktion des Treibstoffverbrauchs und der Entwicklung alternativer Antriebskonzepte eine immer größere Bedeutung zu. Fahrerassistenzsysteme können die Sicherheit für Verkehrsteilnehmer erheblich erhöhen, da ein Großteil der Unfälle auf den Straßen nach Abbildung 1.1 auf menschliches Fehlverhalten zurückzuführen ist. Das Fahren mit automatisierten Fahrzeugen ist wesentlich komfortabler, da einzelne Aufgaben, die zum Betreiben eines Fahrzeugs nötig sind, vom Assistenzsystem übernommen werden und der Fahrer somit entlastet wird. Fahrerassistenzsysteme die die Längsführung eines Fahrzeugs übernehmen, können auch die Energieeffizienz von Fahrzeugen erhöhen. Da die Reaktionszeit eines Fahrers entfällt, können die Abstände zwischen den Fahrzeugen verringert werden, womit der Luftwiederstand sinkt. Aus diesen Gründen investieren viele OEMs in die Weiterentwicklung von Fahrerassistenzsystemen.

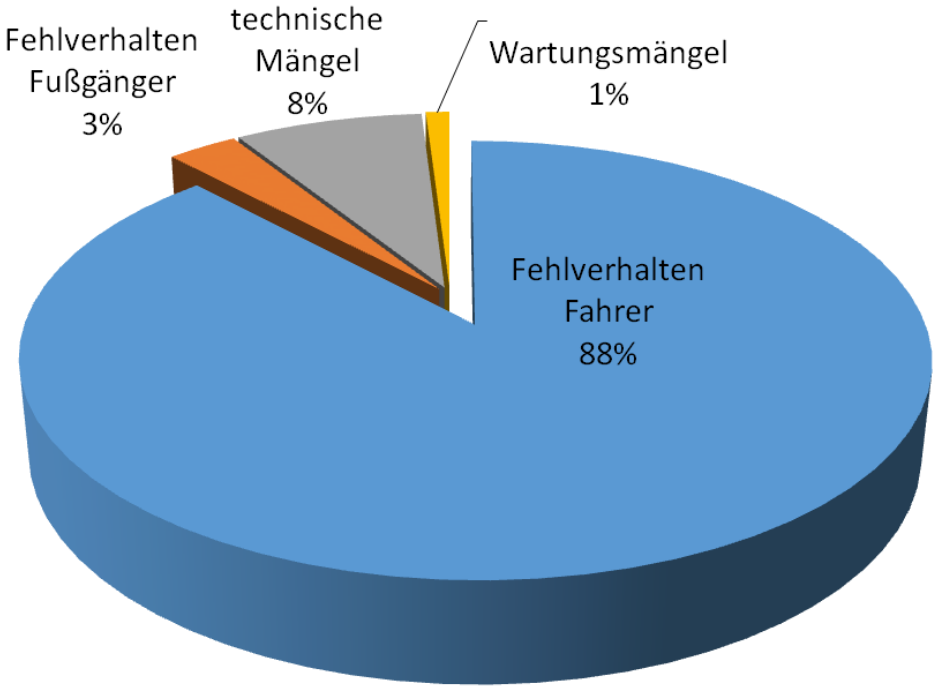

Abbildung 1.1.: Unfallursachen in Deutschland 2015, [Des16]

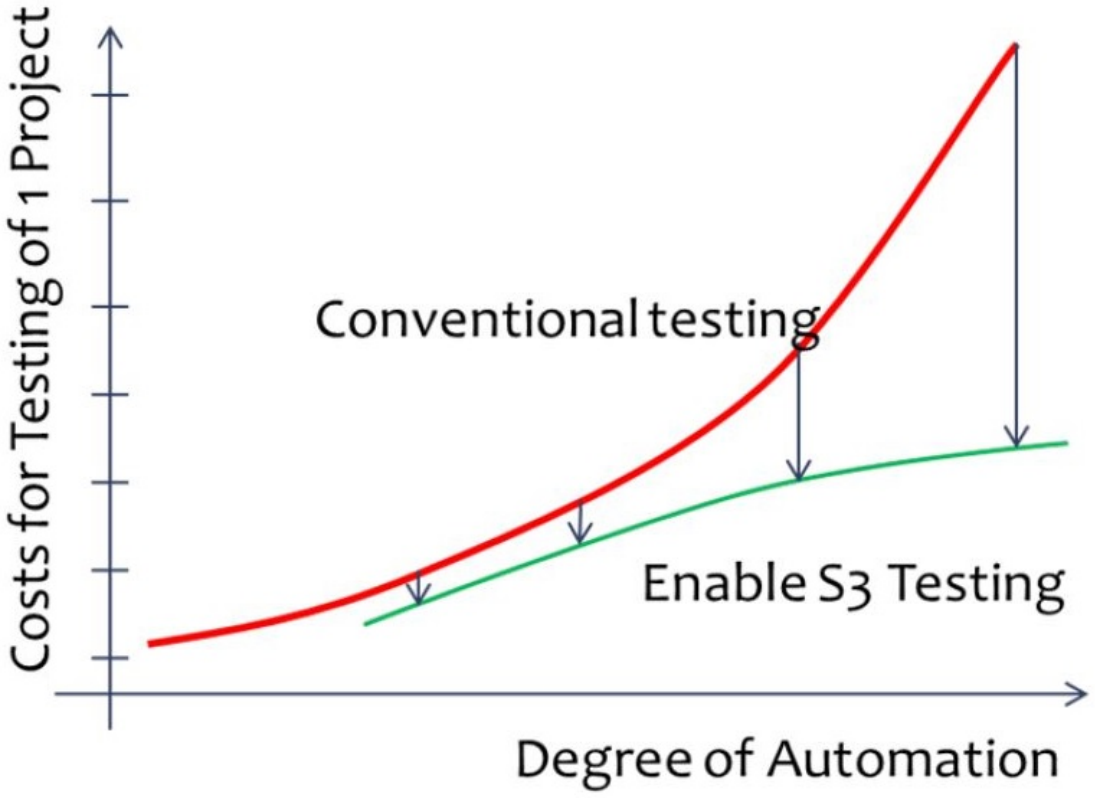

Abbildung 1.2.: Ziel der Kostenreduktion von ENABLE-S3, [Ena17]

Im Rahmen des europäischen Forschungsprojekt ENABLE-S3 (**E**uropean **I**n**i**ti**a**tive to Ena**b**le Validation for Highly Automat**e**d **S**afe and **S**ecure **S**ystems), koordiniert durch die AVL List GmbH, soll ein kooperativer Stauassistent entwickelt werden, welcher Fahrzeuge und ihre Passagiere sicher durch Stauszenarien bei geringen Geschwindigkeiten führen soll. Das Ziel des europäischen Forschungsprojekts ENABLE-S3 ist es, physische Validierungs- und Verifizierungsaufwände durch virtuelles Testen und Verifizieren zu unterstützen, da reale Tests teuer, zeitaufwändig, unvollständig, nicht wiederholbar oder exakt reproduzierbar und potentiell gefährlich sein können. In Abbildung 1.2 ist das theoretische Ziel der Kostenersparnis der Simulation gegenüber dem konventionellen Testen hochautomatisierter Systeme dargestellt. ENABLE-S3 versucht eine Lösung zu finden Simulation und Realversuch zu kombinieren. [Ena17]

Das Forschungsprojekt umfasst die Domänen Luftfahrt, Automobil, Landwirtschaft, Gesundheit, Seefahrt und Schienenverkehr. Im Rahmen dieser Arbeit wird ein Anwendungsfall im Automobilbereich umgesetzt. Zielsetzung ist die Modellbildung und Simulation eines kooperativen Stauassistenten (engl.: Traffic Jam Assistant, TJA) über ein Co-Simulationsframework. Es soll ein parallel entwickelter TJA-Regler mit Platooningfunktion (siehe Kapitel 1.2.4) in der aufgebauten Simulationsumgebung getestet werden. Die

Simulationsumgebung besteht aus einer Verkehrssimulation, den Fahrzeugmodellen für die Ego-Fahrzeuge (Fahrzeuge an denen der TJA getestet wird) und dem TJA-Regler. Um die Funktionsfähigkeit des TJA-Reglers nachzuweisen, werden verschiedene Szenarien (Kapitel 3.4 TJA mit Platooning, Seite 50 und Kapitel 3.5 ENABLE-S3 UC4 - Szenario, Seite 57) simuliert. Die Funktionen des Stauassistenten sind auf der Homepage des ENABLE S-3 Projekts [Ena17] beschrieben. Es handelt sich um einen Stauassistenten mit Platooningfunktion welcher Fahrzeuge automatisiert und sicher durch verschiedene Stauszenarien führen soll. Dazu werden folgende bestehende Assistenzsysteme kombiniert:

- **„Stop-and-Go“ und Adaptive Cruise Control (ACC)**
  Ermöglicht das Folgen eines vorherfahrenden Fahrzeugs mit Verzögerung bis zum Stillstand und wieder Anfahren.
- **Lane Keeping Assistant (LKA)**
  Führt das Fahrzeug innerhalb des Fahrstreifens.
- **Vehicle to Vehicle(V2V) und Vehicle to Infrastrucutre (V2I)**
  Ermöglicht die Kommunikation von Fahrzeugen untereinander bzw. von Fahrzeugen mit der umliegenden Straßeninfrastruktur.

Der Stauassistent soll je nach Verkehrslage dem Fahrer einen im ENABLE-S3 Projekt definierten Fahrmodus [PHdC17] vorschlagen und auf dessen Aufforderung aktivieren. Fährt das Fahrzeug mit einer mittleren Geschwindigkeit größer als 60 km/h wird vom TJA die Umgebung überwacht, jedoch keine Aktivierung des automatisierten Fahrmodus empfohlen. Fährt das Fahrzeug langsamer als diese Geschwindigkeit wird dem Fahrer die Aktivierung des TJA vorgeschlagen. Bestätigt der Fahrer die Aktivierung des TJA, wird in den semi-automatisierten Modus umgeschaltet. Das Fahrzeug fährt dann mit den Assistenzsystemen ACC und LKA. Verringert sich die mittlere Geschwindigkeit weiter unter 20 km/h wird automatisch die Platooningfunktion aktiviert. Beim Platooning werden Daten der V2V-Kommunikation verwendet. Fahrzeuge innerhalb eines Platoons können in geringem Abstand hintereinanderfahren. Ihr Längsdynamikverhalten entspricht dabei näherungsweise dem eines einzelnen Fahrzeugs. Die Vorteile und Potentiale des Platoonings sind laut [MWK17] und [ABT+15] geringerer Kraftstoffverbrauch und Emissionen durch den verringerten Luftwiderstand beim Fahren in geringeren Abständen. Geringere Abstände zwischen Fahrzeugen im Platoon ermöglichen auch eine höhere Verkehrseffizienz und Ausnutzung der Infrastruktur. Durch die V2V-Kommunikation kann schneller auf andere Fahrzeuge im Platoon reagiert werden, wodurch beispielsweise die benötigte Verzögerungsrate beim Bremsen verringert werden kann. Dies erhöht die Sicherheit und den Fahrkomfort für die Fahrzeuginsassen.

Der beschriebene Stauassisten wird in einem Co-Simulationsframework simuliert. Mit der Co-Simulationsplattform Model.CONNECT können mehrere Simulationsprogramme gleichzeitig simuliert werden. In diesem Anwendungsfall werden VTD (Virtual Test Drive), VSM (Vehicle Simulation) und Simulink miteinander verbunden. VTD simuliert die Fahrzeugumgebung wie Verkehr und Umwelt. VSM berechnet das Fahrzeug-

verhalten mit den Regelgrößen Gas- und Bremspedalstellung sowie dem Lenkradwinkel als Eingangsparameter. Die Assistenzsysteme wie ACC und LKA werden mit MATLAB/Simulink ausgeführt.

1.1. Fahrerassistenzsysteme und automatisiertes Fahren

Erste Fahrerassistenzsysteme wurden bereits früh in der Automobilentwicklung umgesetzt. Cadillac brachte 1912 den ersten elektrischen Starter auf den Markt und bot bereits 1918 einen Vorläufer des Kurvenlichts an. 1932 führte Chrysler den Bremskraftverstärker ein und 1951 folgte die Servolenkung. Moderne assistierende Systeme arbeiten meist mit Sensoren. Beispielsweise liefert ein Regensensor die benötigten Informationen zur automatischen Bedienung des Scheibenwischers oder ein Lichtsensor erkennt eine Tunnelfahrt und schaltet das Licht automatisch ein. [Rei10]

Erste Versuche zum automatisierten Betrieb eines Fahrzeugs wurden in den 1950er Jahren vom General Motors Research Lab entwickelt. Es wurden Magnete im Boden eingesetzt um die Straße für das Fahrzeug erkennbar zu machen. In Japan wurden in den 1970er und 1980er Jahren Forschungen zur Automatisierung von Fahrzeugen ohne Anpassungen der Infrastruktur durchgeführt. Die Erkennung von Objekten und Fahrstreifen erfolgte mit Kamerasystemen. In den 1990er Jahren wurde in verschiedenen amerikanischen und europäischen Projekten versucht, vorgegebene Strecken automatisiert abzufahren. Allerdings überwachte das Fahrzeug immer ein menschlicher Fahrer, welcher in manchen Situationen eingriff. Nach mehreren Versuchen auf einer Wüstenstrecke veranstaltete die Defense Advanced Research Projects Agency (DARPA) aus den USA im Jahr 2007 einen Wettbewerb zum automatisierten Fahren in einer vorstadtähnlichen Umgebung. Die fahrerlosen Fahrzeuge mehrerer Teilnehmer bewegten sich durch eine Umgebung mit zusätzlichen Fahrzeug mit menschlichen Fahrern und mussten Verkehrsregeln befolgen. Die automatisierten Fahrzeuge konnten nur durch eine Funkverbindung gestoppt werden. Einige Teams lösten die Aufgabe und präsentierten ihren Versuchsträger im öffentlichen Verkehr. [WHLS15] (Seiten 1140 bis 1142)

Fahrerassistenzsysteme haben die Aufgabe, den Menschen beim Steuern eines Fahrzeugs zu unterstützen und übernehmen einzelne Aufgaben, die zum Betreiben eines Fahrzeugs benötigt werden. Diese Aufgaben sind nach [WHLS15] Seite 12 folgendermaßen definiert:

- **Wahrnehmung von Informationsquellen**
  - optische und akustische Informationen im Fahrzeug
  - andere Verkehrsteilnehmer
  - Straßencharakteristik, Verkehrsschilder, Beschaffenheit der Fahrbahnoberfläche
- **Beurteilungsleistung**
  - Abstände zu anderen Fahrzeugen

  - Geschwindigkeiten anderer Verkehrsteilnehmer
  - Antizipieren kritischer Situationen
- **Entscheidungsprozesse**
  - Handlungen zum Navigieren des Fahrzeugs: Geschwindigkeit, Wahl des Fahrstreifens
- **Fahrzeugbedienung**
  - Regelung der Fahrzeugbewegungen
  - Bedienung Licht, Scheibenwischer, Radio

Durch die Unterstützung der Fahrerassistenzsysteme bei diesen Aufgaben soll der Fahrkomfort und die Sicherheit erhöht werden. Es können kritische Fahrsituationen entschärft werden. Ist der Fahrer beispielsweise durch das Radio oder Navigationssystem abgelenkt und das Vorderfahrzeug bremst, kann es zu einem Auffahrunfall kommen. Fahrerassistenzsysteme können ohne Beteiligung des Fahrers einen ausreichenden Sicherheitsabstand zum Vorderfahrzeug einhalten. Auch nach Eintritt einer kritischen Situation unterstützen Fahrerassistenzsysteme den Fahrer. Beispielsweise bei einer Vollbremsung durch ESP und ABS ([WHLS15], Seite 112). Im Falle höherer Automatisierungsstufen (siehe Tabelle 1.1) wird der Fahrkomfort deutlich erhöht, da der Fahrer das Fahrzeug nicht mehr ständig überwachen muss und andere Tätigkeiten ausführen kann.

Ein weiterer positiver Faktor des automatisierten Fahrens ist die steigende Ökonomie des Fahrens und die Erhöhung der Verkehrsflussdichte bzw. Nutzung der Infrastruktur. Ein automatisiertes Fahrzeug kann den Fahrzustand so wählen, dass ein optimaler Kraftstoffverbrauch bzw. Energieverbrauch erzielt wird. Betreffend der besseren Nutzung der Infrastruktur unterscheidet Laurgeau in [Lau12] zwischen Quer- und Längsnutzung der Straße. In Europa ist die standardisierte Fahrstreifenbreite 3,5 m [EC08], die Breite eines Fahrzeugs nur durchschnittlich 1,75 m. Der große sicherheitsbedingte Spielraum ist auf die begrenzte menschliche Fähigkeit die Spur zu halten zurückzuführen. Autonome Fahrzeuge können die Spur besser halten. Daher könnten auf einer 2-spurigen Fahrbahn 3 Fahrzeuge nebeneinander fahren, was einer Steigerung der Straßennutzung von 50 % entsprechen würde. Laurgeau [Lau12] berücksichtigt bei dieser Theorie die durchschnittliche Fahrzeugbreite von 1,75 m. Die gesetzliche Maximalbreite eines Kraftfahrzeugs darf in Österreich allerdings 2,55m betragen [Bun17a]. Ein Nebeneinaderfahren von 3 Fahrzeugen mit Maximalbreite auf einer 2-spurigen Fahrbahn wäre somit nicht möglich. Der Sicherheitsabstand zwischen zwei Fahrzeugen in Längsrichtung muss aufgrund der Reaktionszeit des menschlichen Fahrers und möglichen unterschiedlichen Bremsleistungen mindestens 2 Sekunden betragen. Bei automatisierten Fahrzeugen und zusätzlicher V2V-Kommunikation zwischen den Fahrzeugen entfällt die Reaktionszeit. Dadurch kann der Sicherheitsabstand wesentlich verringert werden und so die Verkehrsflussdichte erhöht werden. Nach [22109]kann der Mindestzeitabstand zwischen zwei Fahrzeugen auf 1 s reduziert werden. Dieses Potential soll auch im Fall der in dieser Arbeit thematisierten Platooningfunktion genutzt werden.

Kategorie A: Informierende und warnende Funktionen	Kategorie B: Kontinuierlich automatisierende Funktionen	Kategorie C: Eingreifende Notfallfunktionen (unfallgeneigte Situation)
Wirken ausschließlich „mittelbar" über den Fahrer auf die Fahrzeugführung	Haben unmittelbaren Einfluss auf die Fahrzeugsteuerung (bewusste Übertragung durch den Fahrer – arbeitsteilige Ausführung). Immer übersteuerbar, i.d.R. Komfortfunktionen	Haben unmittelbaren Einfluss auf die Fahrzeugsteuerung in unfallgeneigten Situationen, die der Fahrer faktisch nicht mehr kontrollieren kann (i.d.R: Sicherheitsfunktionen)
Gestaltungsbeispiele: • Verkehrszeichenassistenz (bspw. Anzeige der Geschwindigkeitsbegrenzung) • Spurverlassenswarnung (bspw. Vibration am Lenkrad)	Gestaltungsbeispiele: • Adaptive Geschwindigkeitsregelung (ACC) • Spurhalteassistenz (über Lenkeingriffe)	Gestaltungsbeispiele: • Automatisches Notbremssystem (systeminitiiert) • Ausweichsystem • Nothaltesystem (Fahrer handlungsunfähig)

Abbildung 1.3.: Kategorien der Faherassitenzsysteme nach [WHLS15] (Seite 29)

Fahrerassistenzsysteme können in 3 Kategorien eingeteilt werden (Abbildung 1.3). Diese Einteilung erfolgt über die Wirkungsweise der Systeme und ist auch in rechtlicher Hinsicht relevant. Kategorie A übernimmt nur eine informative Funktion. Die Information wird dem Fahrer optisch, akustisch oder haptisch mitgeteilt. Systeme der Kategorie B übernehmen teilweise Aufgaben der Fahrzeugführung. Der Status des Systems wird dem Fahrer bereitgestellt und der Fahrer kann das Assistenzsystem jederzeit übersteuern. Die Besonderheit der Systeme der Kategorie C und Unterscheidung zur Kategorie B liegen darin, dass sie in Notsituationen vorübergehend die Kontrolle über das Fahrzeug übernehmen, ohne dass der Fahrer eingreifen kann.

Die Kategorie B kann in weitere Automatisierungsgraden unterteilt werden. Die auch in [WHLS15] zitierte Projektgruppe Bundesanstalt für Straßenwesen (BASt) definiert 5 (0 bis 4) unterschiedliche Automatisierungsgrade (siehe Tabelle 1.1). Die niedrigste Stufe 0 hat keine Automatisierung. In den höheren Stufen steigt der Automatisierungsgrad bis zur Stufe 4 (Vollautomatisiert). Der in dieser Arbeit simulierte Stauassistent bewegt sich auf Stufe 3 nach SAE und BASt. Eine weitere Einteilung der Kategorie B aus Abbildung 1.3 wurde von SAE (Society of Automotive Engineers) vorgenommen. In Abbildung 1.4

sind die SAE Stufen angeführt. Es wird auch der Vergleich zu den entsprechenden BASt Stufen angeführt.

Tabelle 1.1.: Grade der Automatisierung nach BASt [Bun17b]

Stufe	Nomenklatur	Fahraufgaben des Fahrers
0	Driver only	Der Fahrer führt dauerhaft die Längs- und Querführung des Fahrzeugs aus.
1	Assistiert	Der Fahrer führt entweder die Längs- oder die Querführung aus. Die jeweils andere Aufgabe wird in Grenzen vom Fahrerassistenzsystem ausgeführt
2	Teilautomatisiert	Das Fahrerassistenzsystem übernimmt Quer- und Längsführung für einen gewissen Zeitraum und/oder in spezifischen Situationen. Der Fahrer muss die Situation dauernd überwachen und die Steuerung gegebenenfalls übernehmen können.
3	Hochautomatisiert	Das Fahrerassistenzsystem übernimmt Quer- und Längsführung für einen gewissen Zeitraum und/oder in spezifischen Situationen. Der Fahrer muss die Situation nicht dauernd überwachen und die Steuerung nach Aufforderung und mit einer Zeitreserve übernehmen können.
4	Vollautomatisiert	Das Fahrerassistenzsystem übernimmt Quer- und Längsführung in einem spezifischen Anwendungsfall. Wird das Fahrzeug nach Aufforderung nicht vom Fahrer übernommen, kehrt das Fahrzeug selbstständig in den risikominimalen Systemzustand zurück.

Level	Name	Narrative definition	Execution of steering and acceleration/ deceleration	Monitoring of driving environment	Fallback performance of *dynamic driving task*	System capability (*driving modes*)	BASt level	NHTSA level
Human driver monitors the driving environment								
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a	Driver only	0
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes	Assisted	1
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes	Partially automated	2
Automated driving system ("system") monitors the driving environment								
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task* with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes	Highly automated	3
4	**High Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes	Fully automated	3/4
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**	-	

Abbildung 1.4.: Kategorien der Faherassitenzsysteme nach SAE [Sta17]

1.2. Stand der Technik

1.2.1. Adaptive Cruise Control (ACC)

Auf Seite 852 in [WHLS15] definiert Winner ACC als eine Fahrgeschwindigkeitsregelung, die sich an die Verkehrssituation anpasst. ACC ist eine Weiterenticklung der CC (Cruise Control), auch Tempomat genannt. Mitsubishi brachte 1995 als erster Automobilhersteller ein ACC-System auf den Markt. Europäische Hersteller folgten 1999 mit einem System welches im Gegensatz zu Mitsubishi auch Bremseingriffe durchführte. Ist der ACC-Modus aktiviert, kann eine gewünschte Fahrgeschwindigkeit eingestellt werden. Fährt das mit ACC ausgestattete Fahrzeug auf ein langsameres, vorausfahrendes Fahrzeug auf, wird die Geschwindigkeit mit Eingriffen in die Aktuatoren Drosselklappe bzw. Bremszylinder dem vorausfahrenden Fahrzeug angepasst und das ACC-Fahrzeug folgt ihm mit einem gewünschtem geschwindigkeitsabhängigem Zeitabstand. Beschleunigt das vorausfahrende Fahrzeug oder wechselt es den Fahrstreifen, nimmt das ACC-Fahrzeug seine gewünschte Geschwindigkeit wieder auf. Die Erfassung des Verkehrs erfolgt dabei entweder mit Radar-, Lidar-, oder Kamerasensoren oder einer Kombination dieser Sensoren. Die Funktionsgrenzen des Systems sind in ISO 15622 festgelegt und auf den Seiten 854-855 in [WHLS15] aufgelistet. Die wichtigsten Eckpunkte dabei sind:

- Das System übergibt bei Unterschreitung von einer Geschwindigkeit von 5 m/s wieder an den Fahrer
- Die Zeitlücke zum Vorderfahrzeug darf 1 s nicht unterschreiten.
- Die Beschleunigungen dürfen minimal -3,5 m/s^2 und maximal 2,5 m/s^2 betragen.

1.2.2. Lane Keeping Assistant (LKA)

Der Lane Keeping Assistant ist ein System zur Querführung des Fahrzeugs. Zusätzlich zu LDW (Lane Departure Warning) Systemen, welche den Fahrer vor dem Verlassen der Fahrspur warnen, greift der LKA durch aktive Lenkunterstützung direkt in die Querführung des Fahrzeugs ein [WHLS15]. Dazu werden Fahrbahnmarkierungen von einem Kamerasensor detektiert. Mit dieser Information über die Fahrbahn wird das Fahrzeug mit Eingriffen über die Aktuatoren der Servolenkung auf der gewählten Spur gehalten. Bei einer sicherheitsbetonten Auslegung wird erst eingegriffen, wenn sich das Fahrzeug bereits nahe an den Fahrbahnmarkierungen befindet. Eine komfortbetonte Auslegung versucht das Fahrzeug mittig auf der Fahrspur zu halten. Das System muss vom Fahrer aktiviert werden und kann laut [Ver17] üblicherweise in einem Geschwindigkeitsbereich von 65 bis 180 km/h eingesetzt werden. Hauptsächlich wird das System auf Autobahnen eingesetzt, da der minimale mögliche Kurvenradius von 230 Metern (laut [Ver17]) für Stadtgebiete nicht geeignet ist.

1.2.3. Traffic Jam Assistant (TJA)

Der TJA kombiniert die Systeme ACC und LKA. Der ACC-Regler muss allerdings auch für den Stop-and-Go-Betrieb erweitert sein, um auch in Stausituationen die Längsführung des Fahrzeugs ausführen zu können. Die Querführung des Fahrzeugs übernimmt der LKA Regler. Die Regelung beim LKA erfolgt üblicherweise mit Fahrbahnmarkierungen, welche von Kameras detektiert werden. In Stausituationen sind diese oft verdeckt, daher erfolgt beim TJA die Querführung des Fahrzeugs auch über die Position des Vorderfahrzeugs. Verlässt das Vorderfahrzeug die Fahrspur bestehen die zwei folgenden Optionen. Entweder kann die Führung des Fahrzeugs wieder an den menschlichen Fahrer übergeben werden oder das Ego-Fahrzeug (Fahrzeug mit dem TJA) folgt dem Vorderfahrzeug. Dazu sind weitere Sensoren nötig, um zu erkennen, ob die Verkehrssituation das Folgen des Vorderfahrzeugs erlaubt. Es können auf Informationen der Rückfahrkamera und Einparkassistenten zurückgegriffen werden. Dieses System ist aufgrund von Einschränkungen der Verlässlichkeit von Sensoren noch auf SAE Level bzw. BASt-Stufe 2 (siehe Abbildung 1.4) ausgeprägt. [WHLS15] (Seiten 996 bis 1003)

1.2.4. Platooning

Bergenhem definiert in [BSC+12] einen Platoon als eine Ansammlung mehrerer Fahrzeuge, welche durch aktive Koordinierung in einer Formation fahren. Im Unterschied zum TJA werden beim Platooning nicht nur Sensordaten sondern auch Daten einer Fahrzeug-zu-Fahrzeug (engl. Vehicle-to-Vehicle, V2V) Kommunikation verwendet. Alle Fahrzeuge folgen autonom dem ersten Fahrzeug. Durch die ständige V2V-Kommunikation entfällt die Reaktionszeit des Menschen oder eines Sensors, welcher den Verkehr bei Systemen wie ACC überwacht. Bremst beispielsweise das erste Fahrzeug im Platoon, wird diese Information in Echtzeit an die folgenden Fahrzeuge übermittelt, welche sofort auf die Verzögerung reagieren können. Dadurch können Fahrzeuge mit geringerem Sicherheitsabstand hintereinanderfahren. Dies erhöht wiederum den Verkehrsfluss und den Verkehrsdurchsatz. Laut Daimler [Dai17] ist dadurch bei LKWs ein Kraftstoffverbrauch von 0,66l/100km pro 1000 kg möglich. Durch die aktive Koordination der Fahrzeuge sollen außerdem die Sicherheit und der Komfort für den Fahrer erhöht werden [BSC+12].

Platooning wurde bisher weitgehend für LKW getestet und ist noch nicht in Serie zu finden. Bislang wurden mehrere Platooning-Projekte erfolgreich durchgeführt [BSC+12]. 2016 haben 6 europäische LKW Hersteller an der European Truck Platooning Challenge [Eck16] teilgenommen. Ziel dieses Projekts war es unter anderem, die unterschiedlichen gesetzlichen Voraussetzungen für Platooning verschiedener EU-Staaten zu vereinheitlichen um Platooning einen Schritt näher zur Implementierung in der EU zu bringen. Dazu wurden 19 Ausnahmegenehmigungen der teilnehmenden EU-Staaten eingeholt. Mit diesen Ausnahmegenehmigungen war es einem Teilnehmer der European Truck Platooning Challenge möglich 5 EU-Staaten im Platoon zu durchfahren. Die größten Risiken im Vergleich zu normalem Verkehr sind laut Behörden die Länge des Platoons sowie die Distanz und Kommunikation zwischen den Fahrzeugen im Platoon.

Potentielle Gefahren des Platoonings werden in [Axe17] diskutiert. Als größte Gefahr beim Platooning wird hier das Auffahren auf ein vorausfahrendes Fahrzeug im Platoon gesehen. Auch Risiken wie etwa die Aufsplittung des Platoons durch umgebende Verkehrsfahrzeuge spielen dabei ebenso eine Rolle wie die möglichen technischen Fehler betreffend Kommunikation, Software, Sensoren, Aktoren oder Ungenauigkeiten der Positionsmessung. Um die Sicherheit des Platoonings zu erhöhen, verlässt sich das einzelne Fahrzeug nicht nur auf V2V-Kommunikation, sondern überwacht selbstständig mit Sensoren den Verkehr, um potentielle Gefahrensituationen besser erkennen zu können. Weitere Herausforderungen in der Entwicklung des Platoonings sind Situationen in denen Fahrzeuge den Platoon verlassen oder beitreten wollen oder der Platoon durch weitere Verkehrsfahrzeuge unterbrochen werden muss. Fährt beispielsweise ein Fahrzeug auf einer Autobahnauffahrt und ein Platoon bewegt sich auf der ersten Fahrspur, so verhindert der Platoon möglicherweise das sichere Auffahren des Fahrzeugs auf die Autobahn. Es muss also der Platoon kurzfristig geteilt und anschließend wieder zusammengeführt werden, um keine weiteren Verkehrsteilnehmer zu behindern und die Platooningfunktion aufrecht zu erhalten.

1.3. Simulation

Aufgrund des steigenden Entwicklungsaufwands und höheren Komplexität von Gesamtsystemen werden experimentelle Testmethoden zunehmend durch virtuelle Simulation ersetzt. Mit einer geeigneten Simulationssoftware ist es möglich, viele Testfälle bereits früh im Entwicklungsprozess zu validieren. In diesem Abschnitt wird die Simulation im Bereich der Entwicklung von Fahrerassistenzsystemen betrachtet. Payerl hat in [Lau12] verschiedene Anforderungen an Simulationsprogramme, die dabei verwendet werden, diskutiert. Einige davon sind:

- **Anwenderfreundlichkeit**
  - Visuelle Aus- und Eingabe
  - Gestaltung der Szenarien
  - Bedienung
- **Funktionale Verkehrsumgebung**
  - Parametrierbarkeit der statischen und dynamischen Objekte
  - Gegenseitige Beeinflussung der Verkehrspartner
- **Konfigurierbarkeit der Fahrzeugeigenschaften**
  - Validierte Fahrzeugparameter vorhanden
  - Modifizierbarkeit der Fahrzeugparameter
  - Möglichkeit des Imports von Fahrzeugmodellen aus anderen Simulationsprogrammen

- **Konfigurierbarkeit des Fahrers**
  - Validierte Fahrermodelle vorhanden
  - Parametrierbarkeit des Fahrermodells
- **Konfigurierbarkeit Fahrmanöver**
  - Standardmanöver vorhanden
  - Visuelle Unterstützung beim Szenarienaufbau
  - Modifizierbarkeit der Szenarien
- **Sensoren**
  - Standardsensoren vorhanden
  - Konfigurierbarkeit der Sensorik
  - eigene Sensormodelle implementierbar
- **Analysemöglichkeiten**
  - Auswertung über Diagramme
  - Visualisierung des Szenarios
- **Echtzeitfähigkeit**
- **Qualität**
  - Softwarestabilität
  - Technischer Support
  - vollständige Dokumentation

Ein weiteres Kriterium ist die Grafikqualität der Simulationssoftware. Dies ist vor allem bei der Verwendung der Software in einem Fahrsimulator relevant, um eine möglichst realistische Situation für den Fahrer zu schaffen. Auch bei der der Objektklassifizierung durch implementierte Kamerasensormodelle ist die Grafikqualität entscheidend. Simulationsprogramme wie CarSim, dSpace ASM, IPG Carmaker, TESIS DYNAWare Dyna4 oder VTD erfüllen den Großteil dieser Anforderungen.

1.3.1. Co-Simulation

In dieser Arbeit wird die Simulationssoftware VTD in einer Co-Simulation verwendet. In [Vir17d] ist die Zielsetzung der Co-Simulation als die Zusammenführung unterschiedlicher Fachbereiche im Entwicklungsprozess definiert. Dazu werden die verschiedenen Simulationsprogramme der Fachbereiche miteinander gekoppelt, um das Verhalten des Gesamtsystems simulieren zu können. Die Co-Simulation hat sowohl Vor- als auch Nachteile gegenüber der Simulation mit einem eigenständigem Simulationsprogramm. Ein

wesentlicher Vorteil der Co-Simulation ist, dass verschiedene Programme parallel simuliert, und bereits vorhandene Simulationsmodelle verwendet werden können. Dabei können die Stärken jedes Programms genutzt werden. Im Fall dieser Arbeit wird beispielsweise die Visualisierung und die mikroskopische Verkehrssimulation (Simulation der einzelner Verkehrsfahrzeuge) von VTD [VIR17a] verwendet. Das Verhalten der Verkehrsfahrzeuge wird intern in VTD berechnet. Die Simulation der Fahrzeugdynamik der Fahrzeuge mit den zu entwickelnden ADAS-Systemen (Advanced Driver Assistance Systems z.B. LKA, ACC, TJA), dem sogenannten Ego-Fahrzeug, wird in dieser Arbeit mit VSM [AVL17a] durchgeführt. In VSM können die Fahrzeugparameter genauer und einfacher in einer grafischen Oberfläche eingestellt werden. Die ADAS-Systeme werden mit MATLAB/Simulink [Mat17] erstellt. Durch die grafische Oberfläche von Simulink ist es einfach übersichtliche Modelle zu erstellen. Alle eigenständigen Programme werden mit einem Co-Simulationsprogramm gekoppelt. Die Kopplung der einzelnen Simulationsprogramme ist eine Fehlerquelle der Co-Simulation, da die einzelnen Programm eigene Koordinatensysteme, Einheitssysteme und Größenbezeichnungen verwenden. Wird eine Ausgangsgröße eines Programms einem anderen übergeben, müssen diese Systeme und Größen übereinstimmen. Die Co-Simulationssoftware unterstützt den Anwender dabei.

In dieser Arbeit wird dazu das Programm Model.CONNECT verwendet. Die folgenden Ausführungen sind auf das Co-Simulationsprogramm Model.CONNECT [Doc17] und dessen User Manual (Kapitel 3) bezogen. Bei der Co-Simulation tauschen die Elemente (eigenständig laufende Simulationsprogramme) Daten zwischen den Simulationsschritten aus. Diese Simulationsschritte werden Makrosteps ΔT genannt. Sie bringen die Simulation vom Zeitpunkt T zum Zeitpunkt $T + \Delta T$. Vor jedem Makrostep werden die Eingangswerte und nach jedem Makrostep die Ausgangswerte jedes Elements aktualisiert.

Für die Co-Simulation ist jedes Element eine sogenannte Blackbox. Das heißt, dass keine Information über die interne Struktur der Elemente bekannt ist, sondern lediglich die Ein- und Ausgangswerte mit den anderen Elementen über Ports ausgetauscht wird. Wird die Simulationsroutine eines Elements als DoStep() definiert, so kann der Ausgangswert y_{t+1} jedes Elements nach Formel 1.1 definiert werden. Jedes Element benötigt eine bestimmte Zeit für die interne Berechnung. Diese Zeitspanne wird als Mikrostep Δt und t als Startzeit des Mikrosteps bezeichnet. Der Eingangswert zum Zeitpunkt t wird u_t genannt. Der interne Status $\boldsymbol{s}$ der Elemente beinhaltet beliebige Informationen wie zum Beispiel frühere Ein- und Ausgangswerte.

$$y_{t+1} = \text{DoStep}(u_t, t, \Delta t, \boldsymbol{s}) \tag{1.1}$$

Abbildung 1.6 verdeutlicht den Unterschied zwischen den Markosteps ΔT und den Mikrosteps Δt_i der einzelnen Elemente, hier FMUs (Funktional Mockup Units). Eine FMU ist ein standardisierte Komponente, die die Schnittstelle FMI (Functional Mockup Interface) implementiert. Ziel von FMI ist eine Standardisierte Schnittstelle zwischen Elementen einer Co-Simulation. Die Beschreibung der Schnittstellendaten erfolgt über eine .xml Datei, der Inhalt einer FMU ist ein C-Code. Simulationsmodelle verschiedener Programme

können zu einer FMU konvertiert werden. Beispielsweise kann ein OEM (Original Equipment Manufacturer) Simulationsmodelle von Zulieferern einfach über eine FMI in sein Simulationsprogramm einbinden. In Abbildung 1.5 ist dieser Vorgang dargestellt. Der Inhalt einer FMU ist für den Anwender nicht einsehbar, [FMI17].

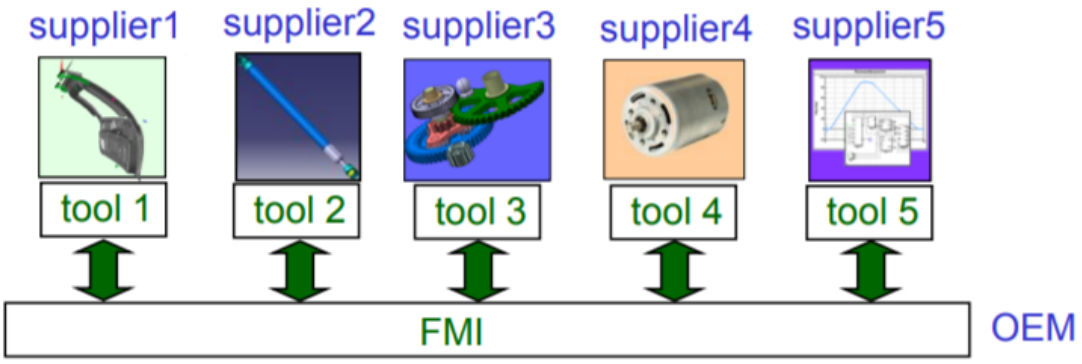

Abbildung 1.5.: Vereinheitlichung der Schnittstellen von Simulationsprogrammen mit FMI, [FMI17]

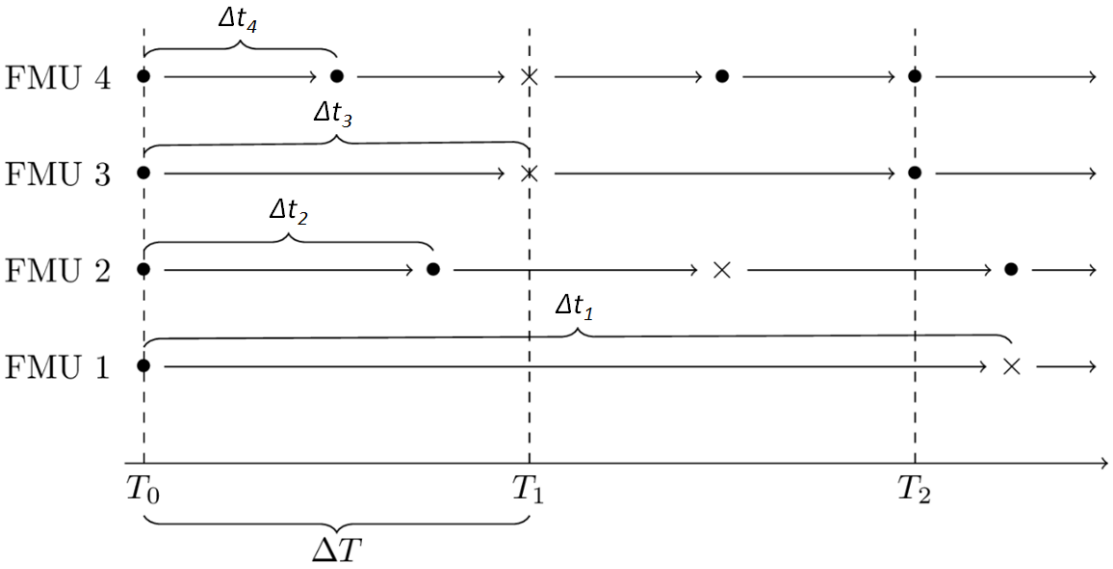

Abbildung 1.6.: Vergleich Makro- (ΔT) und Mikrosteps (Δt_i), [Doc17] (Kapitel 3.2)

Das Element FMU 1 in Abbildung 1.6 hat einen größeren Mikrostep Δt_1 als der definierte Makrostep ΔT. Wird der Zeitschritt zwischen T_1 und T_2 simuliert, befindet sich FMU 1 bereits bei einer Zeit $> T_2$ und rechnet nicht mehr weiter, bis die anderen Elemente aufgeholt haben.

Die Auswertereihenfolge definiert die Abfolge, in der die Elemente der Co-Simulation berechnet werden. Die Auswertereihenfolge kann sequentiell oder parallel erfolgen. Der

Unterschied wird in den folgenden Unterkapiteln beschrieben. Dazu wird eine einfaches Modell (Abbildung 1.7) herangezogen. Die Elemente sind wiederum FMUs.

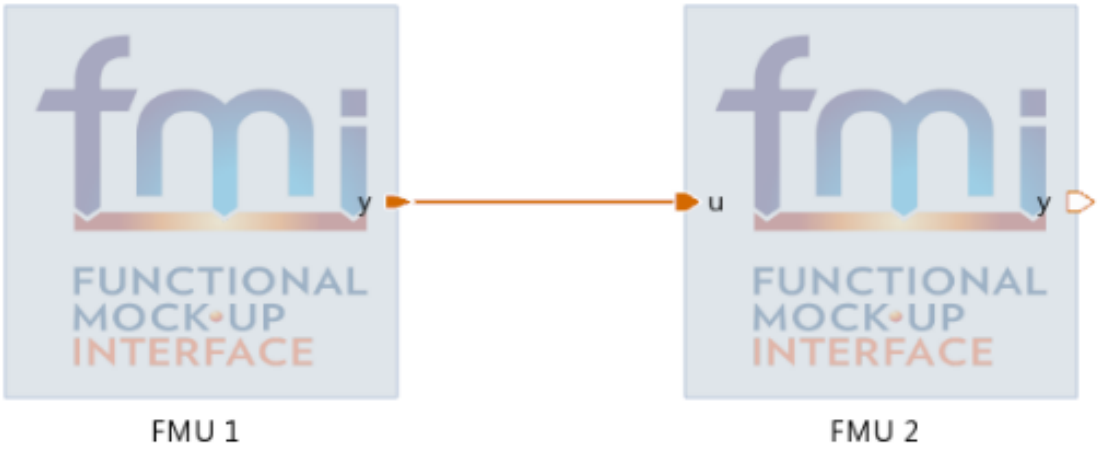

Abbildung 1.7.: Einfaches Modell zur Erklärung der Auswertereihenfolgen, [Doc17] (Kapitel 3.3.2)

Sequentielle Co-Simulation

Bei der sequentiellen Auswertung werden die Ausgangswerte eines Elements als Eingangswert für die folgenden Elemente verwendet. Daher ist die Auswertereihenfolge entscheidend für die Ergebnisse. Sind zwei Elemente verbunden, muss das vorhergehende Element vor dem nächsten Element berechnet werden. Wenn die Topologie des Modells eine Schleife enthält, müssen eine oder mehrere Verbindungen zwischen Elementen getrennt werden um eine Reihenfolge zu erhalten. Die Verbindung wird nur zur Bestimmung der Auswertereihenfolge getrennt. Es werden trotzdem Ergebnisse in der dadurch definierten Reihenfolge zwischen den Elementen ausgetauscht.

In Abbildung 1.8 ist sequentielle Auswertung des Modells in Abbildung 1.7 dargestellt. Die Makrosteps sind hier gleich groß wie die Mikrosteps ($\Delta T = \Delta t$,vgl. FMU 3 in Abbildung 1.6). Die Ausgangswerte y der FMU 1 zum Zeitpunkt k-ten Zeitschritt werden als $y_k^{(1)}$ bezeichnet. Die der FMU 2 als $y_k^{(2)}$.

1. Als erstes berechnen die FMU 1 nach Formel 1.1 ihren Simulationsschritt von t_0 nach t_1 mit $y_0^{(1)}$ als Ausgangswert beim Zeitschritt t_1.
2. Der Ausgangswert $y_0^{(1)}$ wird dann als Eingangswert für die FMU2 verwendet, welche ebenfalls ihren Zeitschritt von t_0 nach t_1 berechnet.
3. Dies wird für alle weiteren Zeitschritte t_k nach t_{k+1} wiederholt.

Zusammengefasst bedeutet das, dass die FMU 2 ihren ersten Zeitschritt bereits mit dem neuen Ausgangswert der FMU 1 berechnet.

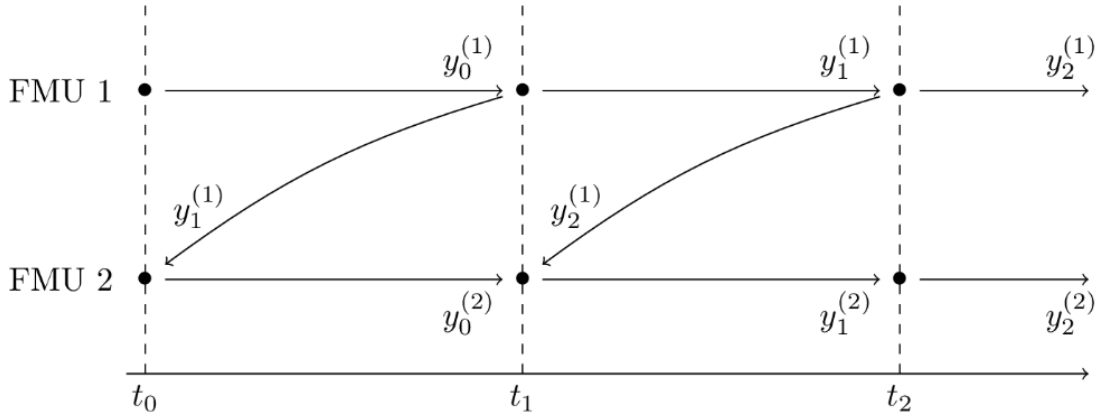

Abbildung 1.8.: Sequentielle Auswertereihenfolge, [Doc17] (Kapitel 3.3.2)

Parallele Co-Simulation

Im Gegensatz zur sequentiellen Simulation berechnen bei der parallelen Auswertereihenfolge alle Elemente ihre Ausgangswerte nur mit den Werten des letzten Kommunikationspunkts (Makrosteps). Werden neuere Werte benötigt (z.B. wenn ein Element einen kleineren Mikrostep als ein folgendes Element aufweist, werden die alten Werte extrapoliert. Die Topologie des Modells beeinflusst die Auswertereihenfolge nicht. In 1.9 ist die parallele Auswertung des Modells in Abbildung 1.7 beschrieben.

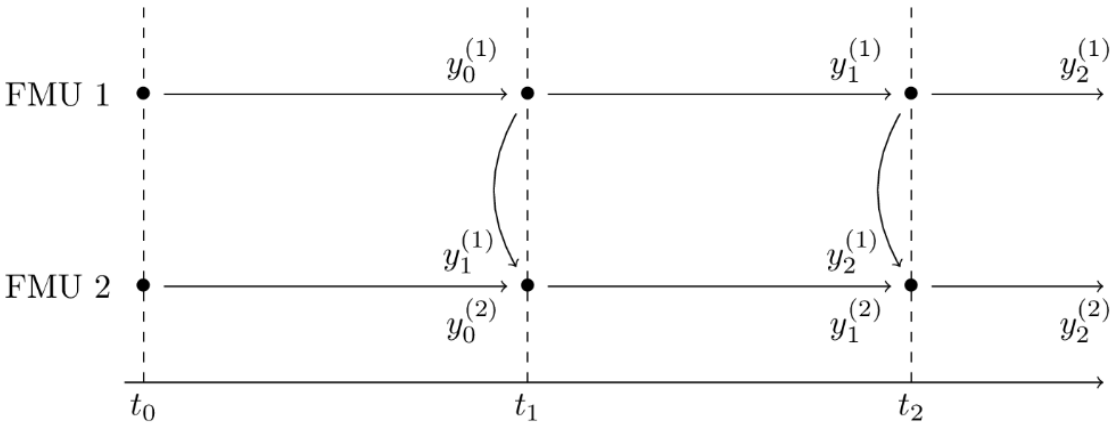

Abbildung 1.9.: Parallele Auswertereihenfolge, [Doc17] (Kapitel 3.3.2)

1. Als erstes berechnen die FMU 1 und FMU 2 parallel nach Formel 1.1 ihren Simulationsschritt von t_0 nach t_1 mit den Ausgangswerten $y_0^{(1)}$ für FMU 1 und $y_0^{(2)}$ für FMU 2 beim Zeitschritt t_1.
2. Der Ausgangswert $y_0^{(1)}$ wird an FMU 2 übergeben.

3. Dies wird für alle weiteren Zeitschritte t_k nach t_{k+1} wiederholt.

Die parallele Auswertung hat den Effekt, dass das Ergebnis der FMU 1 erst im zweiten Zeitschritt der FMU 2 verwendet wird. Es werden nur Eingangswerte verwendet, die beim Start des nächsten Zeitschritts vorhanden sind. Dies führt zu einer Verzögerung der Simulation, welche durch Extrapolationsverfahren minimiert werden kann.

Ablaufkoordination der Co-Simulation

Um die Simulationszeit zu reduzieren und die Genauigkeit der Simulation zu erhöhen, müssen Regeln zur effizienten Ablaufkoordination einer Co-Simulation eingehalten werden. Besonders wichtig ist dies, wenn die Topologie des Modells Schleifen enthält, da in diesem Fall Extrapolationsschritte durchgeführt werden müssen. Ein Beispiel für eine Schleife im Modell ist in Abbildung 1.10 dargestellt. Es handelt sich um einen einfachen Spurhalteassistenten. Das dynamische Fahrzeugmodell (VSM-Element) berechnet die Position des Fahrzeugs auf dem Fahrstreifen, diese wird an einen PID-Regler weitergegeben. Dieser berechnet den nötigen Lenkradwinkel um das Fahrzeug mittig auf dem Fahrstreifen zu halten.

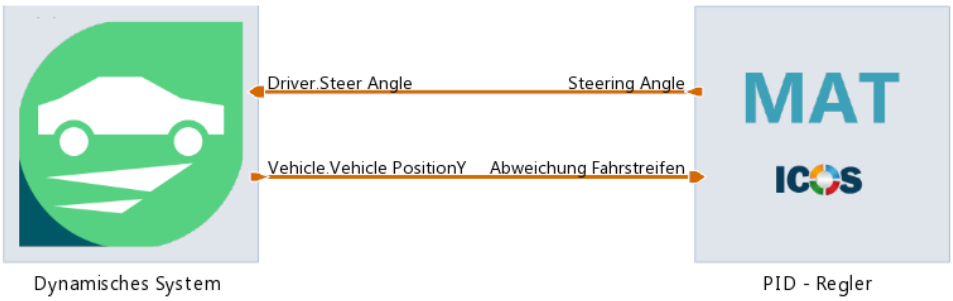

Abbildung 1.10.: Beispiel einer Schleife anhand eines vereinfachten LKA-Modells

Angenommen ein Modell besteht aus einem dynamischen System und einem PID-Regler (Abbildung 1.11), welche gegenseitig Daten austauschen. Bei der sequentiellen Simulationsreihenfolge muss mindestens eine Extrapolation durchgeführt werden um die Schleife zu lösen. In Abbildung 1.11 a) wird zuerst das dynamische System gelöst. Die Eingangssignale sind zu diesem Zeitpunkt noch unbekannt und müssen daher aus den vorherigen Werten extrapoliert werden. Wird die Schleife parallel (b) gelöst, muss zweimal extrapoliert werden. Der Extrapolationsfehler ist also größer, allerdings ist die Simulation bei der parallelen Simulation schneller. Es ist also in jedem Anwendungsfall zu entscheiden welche Herangehensweise (sequentiell oder parallel) gewählt wird bzw. ob Schnelligkeit oder Genauigkeit wichtiger ist.

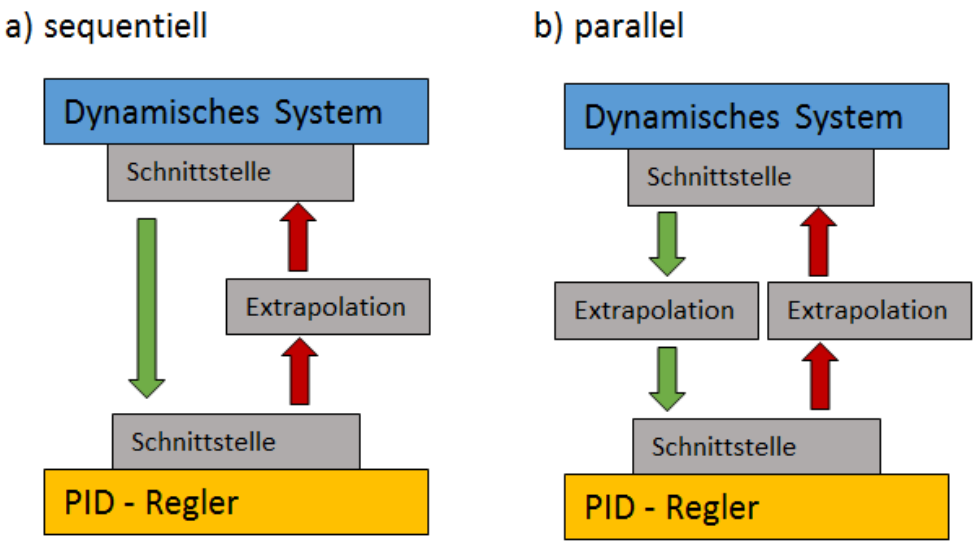

Abbildung 1.11.: Beispiel Ablaufkoordination a) sequentiell b) parallel, [Doc17] (Kapitel 3.3.4.2.3.)

1.3.2. Simulationsstandards

In einem Projekt mit mehreren Teilnehmern ist die Austauschbarkeit von Daten ein wichtiger Punkt. Dies gilt auch für die in VTD generierten Szenarien. Dazu wurde 2014 von der VIRES Simulationstechnologie GmbH und Projektpartnern aus der Automobilbranche das europäische Forschungsprojekt OpenSCENARIO® ins Leben gerufen [VIR17c]. Ziel ist die Standardisierung des dynamischen Inhalts wie zum Beispiel Geschwindigkeitsänderungen oder Fahrstreifenwechsel in Fahrsimulationen. Der statische Inhalt wird mit OpenDRIVE® und OpenCRG® definiert. Mit OpenDRIVE® wird der statische Teil der Fahrsimulation wie z.B. das Straßennetzwerk modelliert. OpenCRG® beschreibt die detaillierten Eigenschaften der Straßenoberfläche [VIR17b].

1.3.3. Sensormodell

Die Fahrumgebungserfassung konventioneller Radarsensoren erfolgt nach Abbildung 1.12. Im ersten Schritt werden im Empfangselement des Radarsensors Reflexionssignale (Energie) empfangen und in Rohsingale (z.B. Spannung) umgewandelt. Mit diesen Rohdaten werden über Annahmen und bekannten Referenzwerten Objekte gebildet. Beispielsweise werden nahe genug beieinander gelegene Reflexionspunkte zu einem Objekt zusammengefasst und zeitlich verfolgt. Mittels Kalmanfiltern kann das Bewegungsverhalten von Objekten vorausberechnet werden. Diese berechneten Daten werden mit neuen Messungen verglichen und korrigiert. Liegen keine neuen Messungen vor, wird der berechnete Zustand des Objekts übernommen. Dies ist z.B. der Fall wenn Objekte vorübergehend

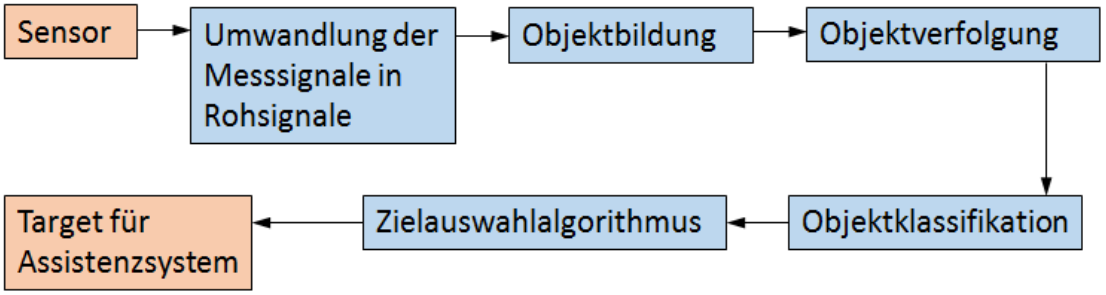

Abbildung 1.12.: Perzeptionsreihenfolge eines Radarsensors, [WHLS15] (Seiten 442 bis 452)

durch andere Objekte verdeckt werden. Als nächstes werden die Objekte klassifiziert, d.h. die Messignale werden mit einer bestehenden Datenbank verglichen und einer Klasse wie, LKW, PKW, Zweiräder oder Fußgänger zugeordnet und in eine Objektliste eingetragen. Auswahlkriterien sind dabei die Reflektionsstärke, die Geschwindigkeit und die Größe der Objekte. Mit dieser Objektliste selektiert der Zielauswahlalgorithmus das Target-Objekt für das Assistenzsystem, [WHLS15] (Seiten 442 bis 452) und [Inv17].

2. Methoden

Ziel der Masterarbeit ist es, ein funktionsfähiges Modell zur Simulation eines Stauassistenten zur Verfügung zu stellen. Dazu wird das Modell mit einfachen Reglern, mit denen die Funktionsfähigkeit der Simulationsumgebung und des Modells nachgewiesen wird, aufgebaut. Ist die Funktionsfähigkeit gegeben, können diese Regler beliebig verändert bzw. weiterentwickelt und in anderen Modellen verwendet werden.

2.1. Aufbau der Co-Simulation

Die Co-Simulation wird in Model.CONNECT aufgebaut. In diesem Programm werden VTD, VSM und Simulink miteinander gekoppelt. VTD ist auf einem eigenen Rechner mit LINUX-Betriebssystem installiert und über ein Netzwerk mit dem Simulations-PC, auf dem Model.CONNECT installiert ist, verbunden. Der Datenaustausch zwischen den Rechnern erfolgt über TCP-Ports (Transmission Control Protocol - Ports). In Abbildung 2.1 ist die Simulationsumgebung und deren Datenaustausch vereinfacht dargestellt. Der Makrostep der Simulation beträgt 0.01 s. Dies ist die Standardeinstellung und wird auch in den Schulungsunterlagen [Gmb17] empfohlen und später durch Testszenarien bestätigt, (siehe Kapitel 3.2). Die einzelnen Blöcke werden in den folgenden Kapiteln erläutert. Die Sensormodelle sind momentan noch in VTD integriert, in Zukunft soll aber ein eigenständiges, externes Modell implementiert werden.

2.1.1. Co-Simulation Platform Model.CONNECT

Model.CONNECT [AVL17b] ist die Co-Simulationsplattform. Es können Elemente verschiedener Simulationssoftwares eingefügt werden. Im Modell dieser Masterarbeit sind das die Elemente VTD, VSM, MATLAB und FMU. Jedem dieser Elemente kann ein Modell der jeweiligen Simulationssoftware zugeordnet werden. Anschließend werden die Schnittstelle der Elemente verbunden. Diese Schnittstellen sind Ein- oder Ausgänge, sogenannte „Ports“. Beispielsweise ist ein Eingangs-Port der Fahrzeugsimulation VSM die Gaspedalstellung und ein Ausgangs-Port die Fahrzeuggeschwindigkeit. Jedem dieser Ports sind Name, Einheit und Datentyp zugeordnet. Es können Ausgangs-Ports mit Eingangs-Ports anderer Schnittstellen verbunden werden. Wie Abbildung 2.2 zeigt kann z.B. die berechnete Gaspedalstellung des TJA-Reglers mit dem Eingangs-Port von VSM gekoppelt werden. Zur besseren Übersicht der Benutzeroberfläche können Ports auch zu so genannten Bundles zusammengefasst werden. So werden im Modell alle Ports

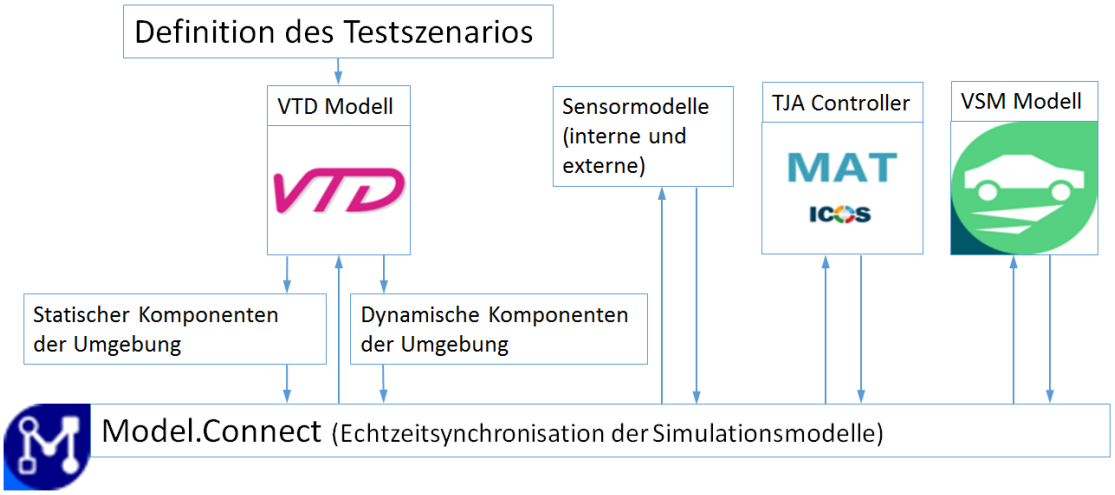

Abbildung 2.1.: Simulationsumgebung

eines Bundles zu einer Verbindung im Modell zusammengefasst. Ein Bundle im VSM-Element ist zum Beispiel „Vehicle_out“ welches mit dem „Ego.Vehicle“-Bundle des VTD-Elements gekoppelt wird. In diesem Bundle sind Informationen wie Fahrzeugposition, -geschwindigkeit und -beschleunigung zusammengefasst.

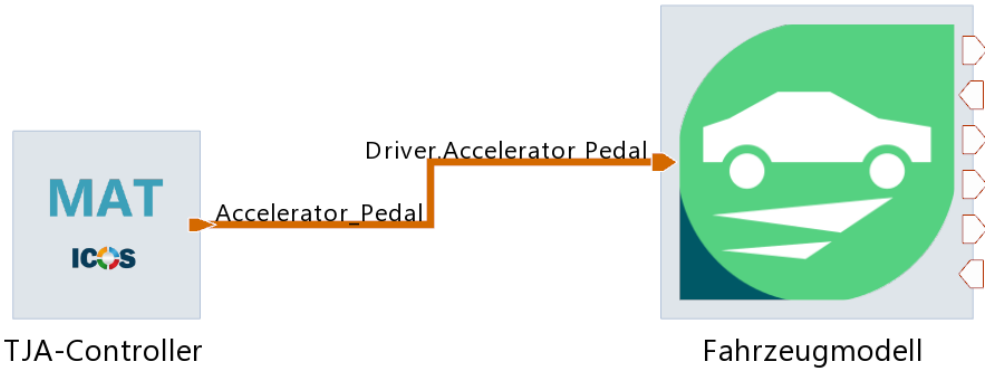

Abbildung 2.2.: Beispiel Port-Verbindungen

Dabei ist zu beachten, dass die Datentypen der Ports übereinstimmen. Sofern Ports die gleiche Größe (z.B. Geschwindigkeit) besitzen, sind auch unterschiedliche Einheiten der verbundenen Ports erlaubt (z.B. m/s und km/h). Model.CONNECT rechnet die Einheiten intern auf SI-Einheiten um. Es ist darauf zu achten, dass jeder definierte Eingangs-Port verbunden ist, da die Simulation sonst nicht gestartet werden kann. Ausgangs-Ports müssen nicht verbunden sein. Jeder momentane Wert der definierten Ports kann während der Simulation in Diagrammen über der Simulationszeit aufgezeichnet werden. Die Simulationsergebnisse eines Model.CONNECT Projekts werden im Projektordner als

Dateien im .csv Format gespeichert. Somit ist eine Weiterverarbeitung der Daten z.B. mit Matlab möglich. Die Datenspeicherfrequenz kann frei gewählt werden. Die Start- und Endzeit der Datenspeicherung kann ebenfalls beliebig eingestellt werden. Dies ist von Vorteil, wenn sich Fahrzeuge zu Beginn eines Szenarios in einem stabilen Zustand befinden sollen, welcher erst nach einer gewissen Zeit nach dem Start der Simulation erreicht werden kann.

2.1.2. Vehicle Dynamics Simulation (VSM)

Vehicle Dynamics Solution (VSM) [AVL17a] ist ein eigenständiges Fahrsimulationsprogramm. Alle Ego-Fahrzeuge der Szenarien werden mit VSM berechnet. VSM besitzt ein eigenes Fahrer- und-Fahrzeugmodell. Für den hier beschriebenen Anwendungsfall kommt allerdings nur das Fahrzeugmodell zum Einsatz und das Fahrermodell wird deaktiviert. Das Fahrermodell wird nicht verwendet, da sich die Ego-Fahrzeuge autonom im Straßenverkehr bewegen und das Fahrzeug durch die ACC-, LKA- und TJA-Regler gesteuert wird. In Abhängigkeit der Eingangsparameter Gas- und Bremspedalstellung, dem Lenkradwinkel und der Fahrbahneigenschaften wie Reibwert zwischen Straße und Reifen und den geometrischen Straßeneigenschaften werden alle Ausgangsgrößen berechnet. Die berechnete Fahrzeugposition und –geschwindigkeit wird über eine Koordinatentransformation der Visualisierung in VTD übergeben.

2.1.3. Umgebungsmodellierung Virtual Test Drive (VTD)

Für die Definition der Straße und des statischen Inhalts der Szenarien wird der in VTD [VIR17a] integrierte RoadDesigner verwendet. Das definierte Straßennetzwerk wird in eine OpenDrive®-Datei exportiert. Für die visuelle Beschreibung wird eine eigene Datei erstellt. Diese beiden Dateien werden im OpenDrive Scenario Editor geladen. Dort wird der dynamische Teil des Szenarios wie Fahrzeugmanöver definiert. Es können Fahrzeuge, Personen und andere Objekte eingefügt werden. Den Fahrzeugen kann ein internes oder externes Fahrermodell zugewiesen werden. Das interne Fahrermodell wird von VTD bereitgestellt und kann modifiziert werden. Die Fahrzeuge bewegen sich mit den Fahrermodellen selbstständig im Straßenverkehr. Es ist möglich, diesen Fahrzeugen Aktionen wie Geschwindigkeitsänderungen oder Fahrstreifenwechsel ausführen zu lassen. Diese können zeit- oder positionsabhängig ausgelöst werden. Wird einem internen Fahrermodell eine Aktion zugewiesen, fährt es nicht mehr selbstständig, sondern führt diese Aktion solange aus, bis dem Fahrermodell ein anderer Befehl erteilt wird. Dem externen Fahrermodell können über Model.CONNECT Gas- und Bremspedalstellung und Lenkradwinkel vorgegeben werden. Die internen Fahrermodelle berücksichtigen auch extern gesteuerte Fahrzeuge im Verkehr. Das erstellte Szenario wird in einer .xml Datei gespeichert und kann in der GUI (Graphical User Interface) von VTD geladen und simuliert werden. Dies ist keine wie eingangs (siehe Kapitel 1.3.2 Simulationsstandards, Seite 18) erwähnte OpenScenario®-Datei. Laut Vires (Herausgeber von VTD) wird es aber in zukünftigen

Versionen möglich sein, aus dem OpenDrive Scenario Editor OpenScenario®-Dateien zu exportieren. In VTD wird die Simulation (Abbildung 2.3) visualisiert. Der Blickwinkel auf die Simulation ist frei wählbar und kann relativ zu einem beliebigen Fahrzeug eingestellt werden.

Abbildung 2.3.: Visualisierung der Co-Simulation mit VTD

In VTD können auch ideale Sensoren erstellt und beliebigen Fahrzeugen zugewiesen werden. Diese lesen während der Simulation bestimmte Daten aus dem Datenbus (Runtime Data Bus, RDB). So können beispielsweise der Relativabstand und –geschwindigkeit zu Fahrzeugen, welche sich im Sensorkegel befinden, ausgelesen werden. Die Öffnungswinkel des Sensorkegels und die Position des Sensors am Fahrzeug können frei gewählt werden. Die Sensorinformationen aus VTD dienen als Eingangsgrößen für den ACC- und TJA-Regler und werden diesen in Model.CONNECT als Vektor übergeben. Jeder Eintrag im Vektor steht für eine Größe eines Fahrzeug. Die Reihenfolge der Fahrzeuge ist nach Abstand zum Fahrzeug mit dem Sensor geordnet und kann sich im laufe der Simulation ändern. Dies unterscheidet das Sensormodell von VTD von einem konventionellen Sensormodell beschrieben in Kapitel 1.3.3. Die Objektliste entspricht in VTD dem Vektor mit den Größen der detektierten Objekten. Den Objekten ist keine gleichbleibende Stelle im Vektor zugeordnet, vielmehr ändert sich die Reihenfolge der Objekte im Vektor je nach Abstand zum Fahrzeug mit dem Sensor. Dies muss dies beim Zielauswahlalgorithmus berücksichtigt werden. Das Sensormodell ist nach Abbildung 2.1 ein internes Sensormodell,

welches in VTD integriert ist.

2.1.4. MATLAB Simulink

Mit MATLAB/Simulink werden alle Regler des TJA programmiert. Das Simulink-Modell wird über eine ICOS (Independent Co-Simulation) Schnittstelle in Model.CONNECT eingebunden. Mit den Eingangsdaten aus VTD und VSM werden Gas- und Bremspedalstellung bzw. der Lenkradwinkel errechnet und wiederum an die Co-Simulation weitergegeben.

2.1.5. Koordinatensysteme

In Abbildung ist das Koordinatensystem nach ISO 8855 abgebildet [HES11].

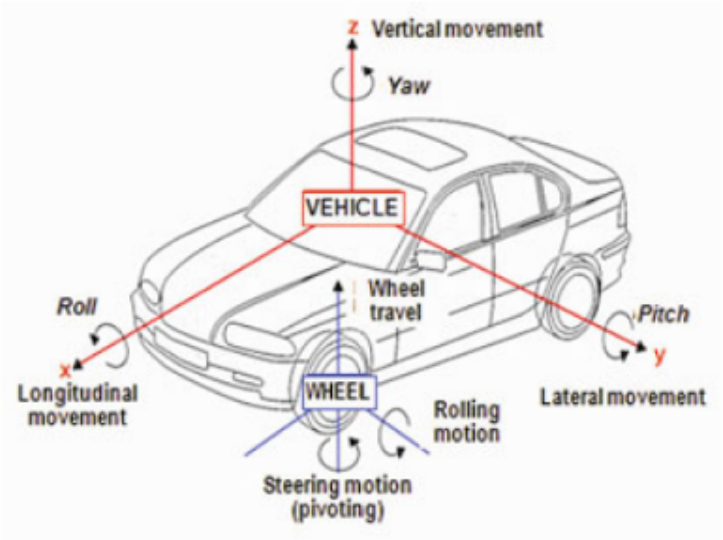

Abbildung 2.4.: ISO 8855 / DIN 70000 Koordinatensystem, [HES11] (Seite 18)

Im Vergleich dazu zeigen Abbildung 2.5 und Abbildung 2.6 die Koordinatensysteme von VTD bzw. VSM. In beiden Koordinatensystemen zeigt die x-Achse in die Längsrichtung des Fahrzeugs sowie positiv nach vorne. Die y-Achse zeigt in Fahrtrichtung positiv nach links. In VTD ist der Ursprung des Koordinatensystems auf Höhe der Straßenebene unter dem Zentrum der Hinterachse. Das Programm VSM legt den Ursprung des Koordinatensystems im Gegensatz zu VTD in das Zentrum der Hinterachse.

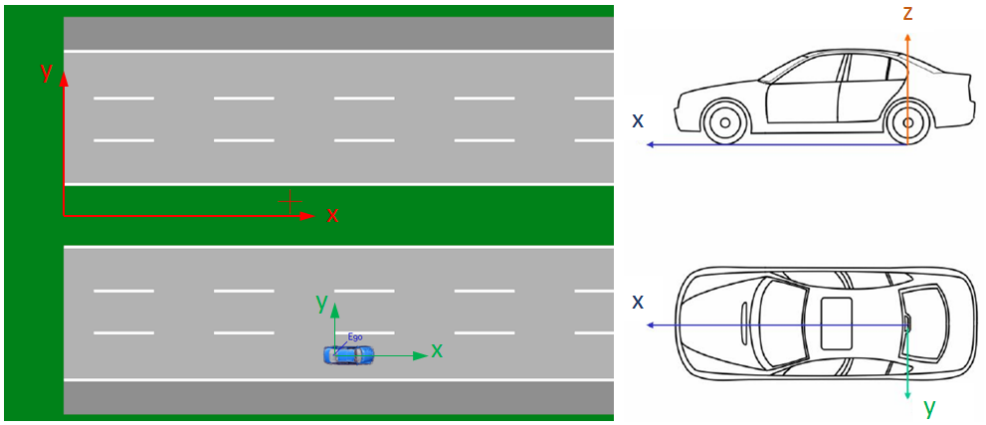

Abbildung 2.5.: Koordinatensystem in VTD, [Dup15] (Kapitel 3) links in Rot die Straßenkoordinaten und in grün die Fahrzeugkoordinaten, rechts die Fahrzeugkoordinaten

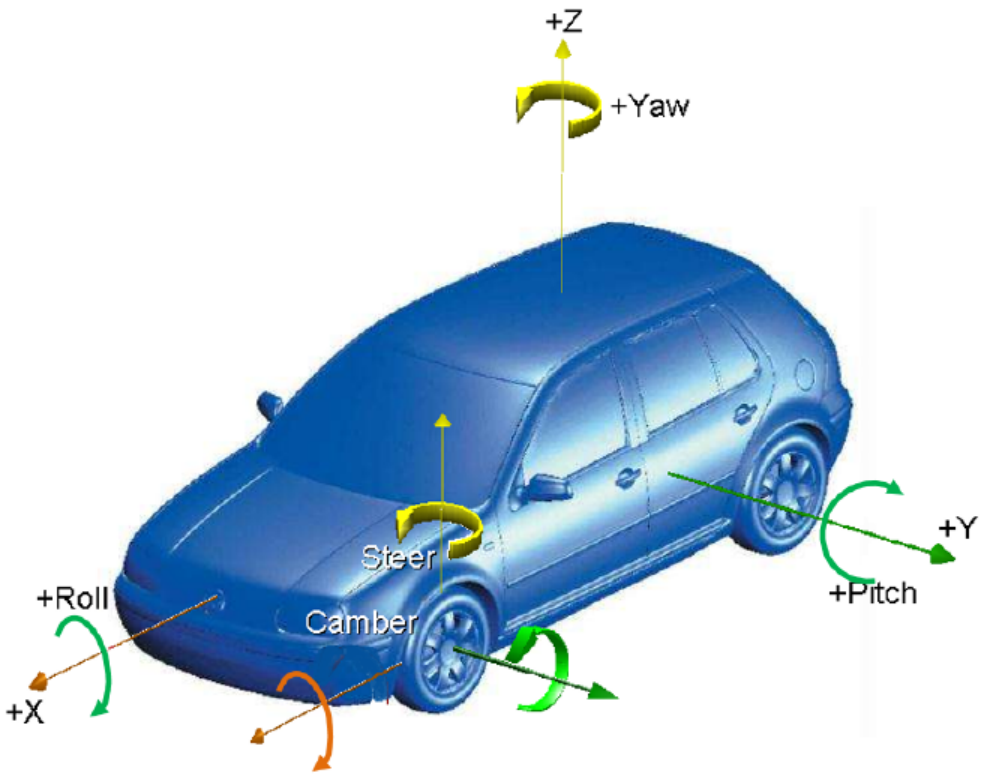

Abbildung 2.6.: Fahrzeugkoordinatensystem in VSM, [Dok17] (Kapitel 3)

Die Wank-, Nick- und Gierrichtungen sind, wie in Abbildung 2.7 ersichtlich in VSM

und VTD unterschiedlich. Das VTD Koordinatensystem entspricht der ISO 8855 Norm. VSM kehrt die Drehrichtungen der Roll-, Nick- und Wankbewegungen um. Die Unterschiede der Koordinatensysteme in VTD und VSM müssen bei der Modellbildung in Model.CONNECT berücksichtigt werden. Die AVL List GmbH (Projektpartner und Herausgeber von Model.CONNECT und VSM) hat bereits eine Koordinatentransformation realisiert und als Blackbox-FMU Modell zur Verfügung gestellt.

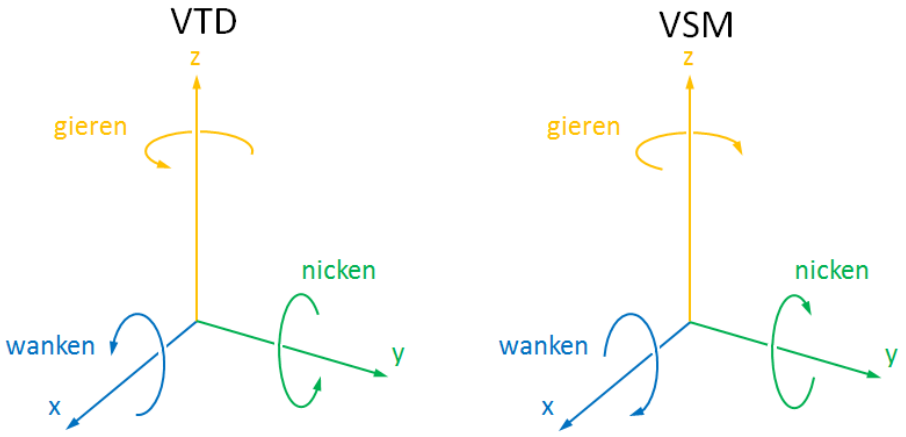

Abbildung 2.7.: Vergleich der Koordinatensysteme in VTD und VSM

2.2. Reglermodelle der Co-Simulation

In diesem Kapitel werden alle in der Co-simulation verwendeten Regler beschrieben. Sie werden bis auf die Koordinatentransformation alle in MATLAB/Simulink erstellt.

2.2.1. Adaptive Cruise Control (ACC)

Der ACC-Regler wurde von IPG CarMaker [IPG17] übernommen und an die Anforderungen der Co-Simulation angepasst. Er besteht im wesentlichen aus drei Bestandteilen:

- Verarbeitung Sensordaten
- Berechnung der Sollbeschleunigung
- PID-Regelung der Gas- bzw. Bremspedalstellung

Details zu diesem Regler können in Programmers Guide vom IPG CarMaker [Aut15] (ab Seite 131) nachgelesen werden.

Verarbeitung der Sensordaten

Wird vom ACC-System kein Target-Fahrzeug erkannt, fährt das System mit einer einstellbaren Wunschgeschwindigkeit. Ein Target-Fahrzeug ist ein vorherfahrendes Fahrzeug auf der gleichen Fahrstreifen wie das Ego-Fahrzeug, welches sich im Sensorkegel befindet, weniger als einen einstellbaren Abstand entfernt ist und langsamer fährt als die vom Fahrer des Ego-Fahrzeugs eingestellte Wunschgeschwindigkeit im ACC-Regler. Der ACC-Regler passt die eigene Geschwindigkeit an die Geschwindigkeit des Ego-Fahrzeugs an und folgt diesem in einem einstellbaren Zeitabstand. Dieser Zeitabstand darf nach ISO 15622 im eingeschwungenen Zustand nicht kleiner als 1 s sein [WHLS15] (Seite 855).

Berechnung der Sollbeschleunigung

Grundsätzlich werden zwei Beschleunigungen berechnet, wenn ein Target-Fahrzeug detektiert wurde. Die benötigte Beschleunigung zum Erreichen der gewünschten Geschwindigkeit ohne Target-Fahrzeug und die benötigte Beschleunigung um auf einen gewünschten Abstand $\Delta s_{x,t}$ zum Target-Fahrzeug zu kommen. Schlussendlich wird die geringere dieser zwei Beschleunigungen als Eingangswert für den PID-Regler der Gas- bzw. Bremspedalstellung übernommen. Dem Regler muss eine gewünschte Zeitlücke zum Target-Fahrzeug Δt_T vorgegeben werden, über diese berechnet das System mit der Geschwindigkeit des Target-Fahrzeugs $v_{x,t}$ den Abstand zum Target-Fahrzeug (siehe 2.1). Die gewünschte Zeitlücke als Berechnungsgrundlage gilt nur bis zu einem einstellbaren Mindestabstand Δs_{xmin} zum Target-Fahrzeug. Dies ist sinnvoll, um bei geringen Geschwindigkeiten oder beim „Stop-and-Go“ keine zu kleinen Abstände zu erhalten. Dieser Mindestabstand wird vom IPG-Modell mit 6 m übernommen.

$$\Delta s_{x,t} = v_{x,t} * \Delta t_T \tag{2.1}$$

PID-Regelung der Gas- bzw. Bremspedalstellung

Der PID-Regler berechnet mit der gewünschten Beschleunigung die Gas- bzw. Bremspedalstellung. Diese werden dann der Fahrzeugdynamikberechnung in VSM übergeben.

2.2.2. Lane Keeping Assistant

Der Lane Keeping Assistant verwendet die Spurabweichung des Ego-Fahrzeugs als Regelgröße. Die Spurabweichung ist die Abweichung des Koordinatensystemursprungs des Ego-Fahrzeugs in VTD zur Fahrstreifenmitte. Dieser Wert wird von VTD an den Regler übergeben. In dieser Arbeit wird mit PID-Reglern über die Ausgangsgröße Lenkradwinkel die Spurabweichung auf Null geregelt und der berechnete Lenkradwinkel an VSM übergeben. Grundsätzlich sind auch andere Reglertypen möglich. Das System besteht aus zwei PID-Reglern. Einen für Geschwindigkeiten unter 8 m/s und einen für darüber. Dies verbessert die Performance des Reglers, da bei unterschiedlichen Geschwindigkeiten verschiedene PID-Reglereinstellungen günstiger sind. Der Wert von 8 m/s wurde durch Testszenarien ermittelt, in denen das Fahrzeug mit verschiedenen Geschwindigkeiten auf einer kurvigen Straße fährt. Detaillierte Informationen zu LKA-Reglern können in [Nes13] nachgelesen werden.

2.2.3. Traffic Jam Assistant und Platooning

Der Traffic Jam Assistant unterscheidet sich vom ACC-Regler durch die Platooningfunktion bei geringen Geschwindigkeiten. Diese Geschwindigkeit wird beim ENABLE-S3 Projekt als 20 km/h angegeben [PHdC17]. Unterschreitet die Verkehrsgeschwindigkeit 20 km/h wird die Platooningfunktion aktiviert.

Platooning Längssteuerung

Der hier beschriebene Regler ist ein erster Versuch eine Platooningfunktion zu realisieren. Es soll damit die Fähigkeit des Modells zu Platooningfunktion nachgewiesen werden. Die Erweiterung des ACC-Modells mit der Platooningfunktion ist in Abbildung 2.8 dargestellt. Ist die Platooningfunktion aktiviert, erfolgt die Regelung des Abstands zum Vorderfahrzeug nicht mehr mit den Sensordaten des Vorderfahrzeugs. Stattdessen wird über V2V-Kommunikation jedem Fahrzeug im Platoon die Geschwindigkeit des ersten Fahrzeugs übermittelt. Damit werden Latenzen bei der ACC Regelung von Fahrzeugen zu Fahrzeugen vermieden (siehe „String Stability“ von konventionellen ACC Systemen, [Ber15], Seite 79). Diese V2V-Kommunikation wird realisiert, indem die Geschwindigkeitsdaten aus VTD direkt an die TJA-Regler der Ego-Fahrzeuge ohne den Umweg über die Sensormodelle übergeben werden. Jedes Fahrzeug im Platoon berechnet seine Beschleunigung zum Einhalten des gewünschten Abstandes $s_{x,t}$ nun mit der Geschwindigkeit des ersten Fahrzeugs (Platooning-Leader). Die neue Target-Fahrzeuggeschwindigkeit $v_{x,t}$ in Formel 2.1 ist also nun jene des Platooning-Leaders. Außerdem wird die Zeitlücke zwischen den Fahrzeugen auf den Minimalwert von 1 s (nach [22109]) gesetzt. Der Mindestabstand von 6 m zum Target-Fahrzeug bleibt gleich. Das Fahrzeug überwacht trotz V2V-Kommunikation weiterhin den Abstand zum direkten Vorderfahrzeug. Verlässt das Fahrzeug einen Grenzabstand von 4 bis 18 m zum Vorderfahrzeug, so wird von der Platooningfunktion wieder auf den ACC-Regler umgeschaltet, bis sich das Fahrzeug wieder in den Grenzabständen befindet. Dieser Sicherheitskorridor wird eingeführt, um auch bei fehlerhafter V2V-Kommunikation eine Kollision der Fahrzeuge zu verhindern. Die Fahrzeuge im Platoon sollen durch die neue Regelung homogener beschleunigen und bremsen, wodurch ein höherer Verkehrsdurchsatz erreicht werden soll. Der Abstand zwischen den Fahrzeugen kann verringert werden, da die Fahrzeuge im Platoon schneller auf Geschwindigkeitsänderungen des Platooning-Leaders (erstes Fahrzeugs im Platoon) reagieren können als über Sensorinformationen herkömmlicher ACC Systeme.

Platooning Quersteuerung

Die Quersteuerung während aktiver Platooningfunktion erfolgt über einen PID-Controller und ist in Abbildung 2.9 dargestellt. Die Regelgröße ist die y-Position des Platooning-Leaders. Alle folgenden Fahrzeuge im Platoon werden zeitversetzt über die Stellgröße Lenkradwinkel auf die y-Position des ersten Fahrzeugs (Platooning-Leader) geregelt und sollen damit dem zurückgelegten Pfad des Platooning-Leaders folgen. Der Zeitabstand in dem die Fahrzeuge dem Platooning-Leader folgen Δt_{PL} wird mit der Geschwindigkeit des Ego-Fahrzeugs $v_{x,e}$ und dem Abstand zum Platooning-Leader $s_{x,PL}$ nach Formel 2.2 berechnet. Außerdem wird die Fahrzeuglänge berücksichtigt.

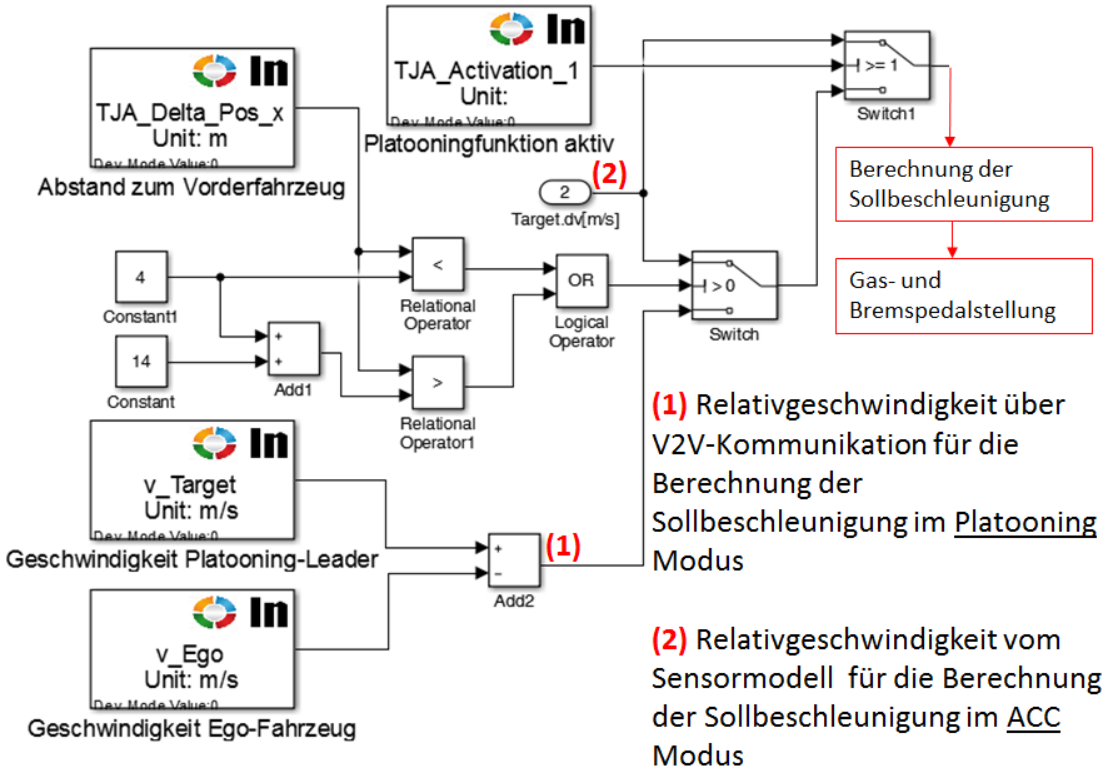

Abbildung 2.8.: Erweiterung des ACC-Simulink Modells mit der Platooningfunktion

$$\Delta t_{PL} = \frac{\Delta s_{x,PL}}{v_{x,e}} \tag{2.2}$$

Fährt beispielsweise das zweite, 5 m lange Fahrzeug im Platoon mit einer Geschwindigkeit von 5 m/s und der Zeitabstand zwischen den Fahrzeugen im Platoon ist 1 s, dann folgt das zweite Fahrzeug im Platoon dem Platooning-Leader nach 2 s. Sind die Zustände des dritten und vierten Fahrzeug im Platoon gleich, so folgen diese mit Abständen von 4 bzw. 6 s. Befindet sich ein Hindernis auf dem Fahrstreifen und der Platooning-Leader weicht diesem durch einen Fahrstreifenwechsel aus, folgen alle Fahrzeuge rechtzeitig. Es wird mit diesem Regler nicht garantiert, dass für jedes Fahrzeug ein sicherer Fahrstreifenwechsel möglich ist. Wenn beispielsweise ein Verkehrsfahrzeug den Platoon überholt, kann ein sicherer Fahrstreifenwechsel für alle Platooningfahrzeuge nicht mehr garantiert werden.

2.2.4. Sensormodell

Ein weiteres Simulink-Modell wird für die Auswahl der Sensordaten benötigt, da diese wie in Kapitel 2.1.3 beschrieben, als Vektoren übergeben werden. Dieses Simulink Modell

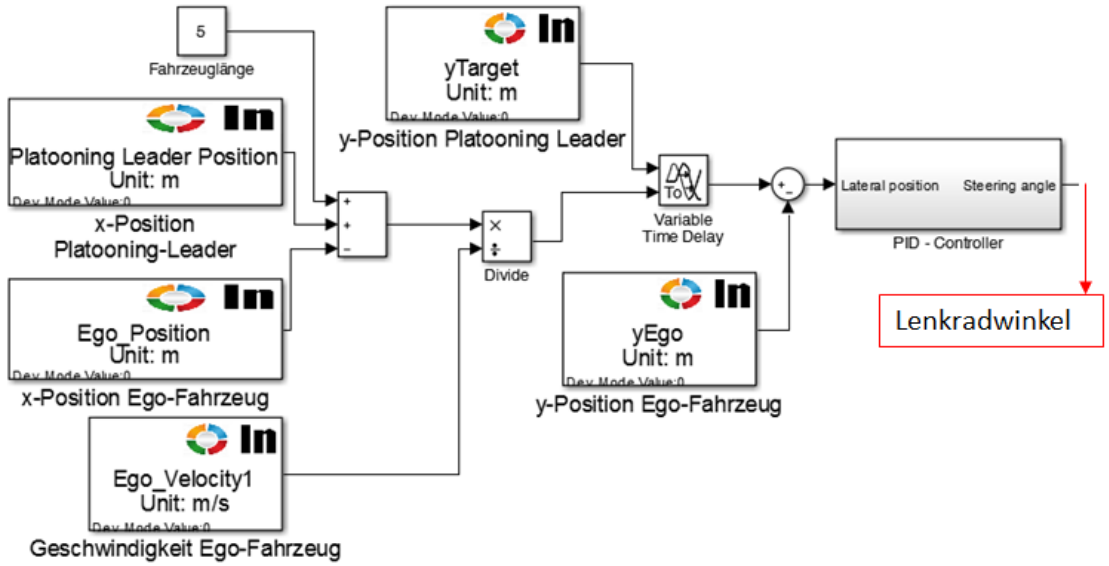

Abbildung 2.9.: Erweiterung des LKA-Simulink Modells mit der Platooningfunktion

entspricht dem Block Zielauswahlalgorithmus aus Abbildung 1.12 in Kapitel 1.3.3. VTD ordnet die Vektorsensordaten Abstand und Relativgeschwindigkeit der Verkehrsteilnehmer nach dem Abstand zum Fahrzeug dem der Sensor zugeordnet ist. Die Reihenfolge der Vektordaten kann sich im laufe der Simulation ändern. Die Eingänge der Simulink-Einheit Sensormodell sind zwei Vektoren und die Ausgänge zwei Skalare. Diese Skalare sind die Relativgeschwindigkeit $\Delta v_{x,t}$ und der Relativabstand $\Delta s_{x,t}$ des nächsten Fahrzeugs, welches sich im Sensorkegel befindet. Das Element Sensormodell kann für beliebig viele Sensoren bzw. Fahrzeuge mit Sensoren erweitert werden.

2.2.5. Fahrstreifenabhängiges Sensormodell

Eine Alternative des im vorherigen Unterkapitel beschriebenen Sensormodells ist das fahrstreifenabhängige Sensormodell. Die Auswahl der Sensordaten erfolgt hier nicht mehr nur nach dem Abstand zu den Verkehrsteilnehmern, sondern auch nach dem Fahrstreifen, auf der sich diese befinden. Die Ausgangsskalare des fahrstreifenabhängigen Sensormodells sind nun jene des Verkehrsteilnehmers, welcher sich auf der gleichen Fahrstreifen und sich am nächsten zum Fahrzeug dem der Sensor zugeordnet ist, befindet. Dazu wird der Simulink-Einheit ein weiterer Sensordatenvektor, nämlich der Fahrstreifen, auf der sich die Verkehrsteilnehmer befinden, übergeben. Die Reihenfolge der Vektordaten ist wiederum nach dem Abstand zum Fahrzeug mit dem Sensor geordnet. Das fahrstreifenabhängige Sensormodell ist vorteilhaft, wenn das Ego-Fahrzeug (Fahrzeug dem der Sensor zugewiesen ist) langsamere Fahrzeuge auf einer anderen Fahrstreifen überholt oder selbst von schnelleren Fahrzeugen überholt wird und diese Verkehrsteilnehmer in den

Sensorkegel eindringen, da in solchen Fällen beim ursprünglichen Sensormodell aus Kapitel 2.2.4 die Ausgänge der Simulink-Einheit auf die Werte dieser Fahrzeuge gewechselt werden. Dadurch folgt das Ego-Fahrzeug dem zu überholenden bzw. dem überholenden Fahrzeug und nicht mehr dem vorgesehenen Target-Fahrzeug, bis das Störfahrzeug den Sensorkegel wieder verlassen hat oder das ursprüngliche Target-Fahrzeug wieder näher am Ego-Fahrzeug ist. Durch das fahrstreifenabhängige Sensormodell wird dies verhindert und das Ego-Fahrzeug folgt nur Target-Fahrzeugen, welche sich auf dem gleichen Fahrstreifen befinden. Auch das Element fahrstreifenabhängiges Sensormodell kann für beliebig viele Sensoren bzw. Ego-Fahrzeuge mit Sensoren erweitert werden. In Kapitel 3 wird ein ACC-Regler, welcher mit fahrstreifenabhängigen Sensordaten arbeitet als LDACC (Lane Dependant Adaptive Cruise Control) bezeichnet.

2.2.6. TJA - Controller

Dieses Simulink-Element dient dem manuellen Ein- und Ausschalten des TJA-Reglers. Wie zu Beginn des Kapitels 1 beschrieben und in [PHdC17] spezifiziert. soll der TJA-Controller manuell auf Wunsch des Fahrers eingeschaltet werden. Dazu wird die Geschwindigkeit des Ego-Fahrzeugs überwacht. Unter einer Geschwindigkeit von 60km/h wird die Aktivierung des TJA-Controllers empfohlen. Durch einen manuellen Schalter im TJA-Controller kann dann die TJA-Funktion aktiviert werden. Fällt die mittlere Geschwindigkeit des ersten Platooning-Leaders (erstes Fahrzeug im Platoon) unter 20km/h, wird automatisch die Platooningfunktion aktiviert. Die mittlere Geschwindigkeit wird aus den Werten der letzten 3 Sekunden ermittelt. Durch die Aktivierung der Platooningfunktion wird der Mindestzeitabstand zwischen den Fahrzeugen auf 1 s gesetzt und das fahrstreifenabhängige Sensormodell. deaktiviert um dem Platooning-Leader bei Querbewegungen zu folgen.

2.2.7. Koordinatentransformation

Die Koordinatentransformation zwischen VTD und VSM wird mit einer FMU (Functional Mockup Unit) wie in Kapitel 2.1.5 beschrieben durchgeführt. Da diese FMU als „Blackbox“-Modell ausgeführt, ist wird es hier nicht näher beschrieben. Alle Ausgänge Ausgangswerte die von VSM auf VTD übertragen werden durch diese FMU umgewandelt.

2.3. Modelle

Es werden zwei verschiedene Modelle simuliert. Einmal das ACC- und LKA-Modell sowie das TJA-Modell mit Platooningfunktion. Das ACC- und LKA-Modell wird zuerst simuliert, um die korrekte Ausführung und Einbettung der Controller in die Co-Simulation

zu testen. Diese Regler sind die Basis für den TJA-Regler, welcher im TJA-Modell verwendet wird.

2.3.1. ACC- und LKA-Modell

Dieses Modell (Abbildung 2.10) regelt ein Ego-Fahrzeug. Es besteht aus dem ACC-Regler, dem LKA-Regler und der Einheit Sensormodell, welche in MATLAB/Simulink erstellt und in Kapitel 2.2 beschrieben wurden, sowie den Komponenten VTD und VSM. Eine weiterer Einheit ist die Koordinatentransformations-FMU zwischen VTD und VSM. Das Fahrzeugmodell in VSM entspricht jenem einer Kompaktklasse. Es wird ein vordefiniertes Modell von VSM verwendet. In VTD können erstellte Szenarien zur Verifizierung des Modells geladen werden. Jedem Element können Ports hinzugefügt werden. Die Ports der Regler wurden bereits bei der Erstellung der Regler definiert. Die genaue Vorgangsweise zum Aufbau eines Modells kann im Anhang A.2 nachgelesen werden.

Abbildung 2.10 zeigt das Modell, wie es in Model.CONNECT aufgebaut ist. Eine bessere Übersicht der Ein- und Ausgänge zeigt Abbildung 2.11. Der Regelkreis besteht aus den 6 Einheiten VTD, VSM, ACC, LKA, Sensormodell und der Koordinatentransformation. Die Sensordaten (Relativabstand und Relativgeschwindigkeit) des Verkehrs werden in der Simulink-Einheit Sensormodell ausgewertet und dem ACC-Regler übergeben. Dieser ermittelt die benötigte Gas- und Bremspedalstellung. Gleichzeitig werden mit der Spurabweichung aus VTD in der Simulink-Einheit LKA der Lenkradwinkel, um mittig auf dem Fahrstreifen zu bleiben, berechnet. Gas- und Bremspedalstellung sowie der Lenkradwinkel sind die Eingänge der VSM-Einheit. Diese berechnet die Position, Geschwindigkeit und Beschleunigung des Ego-Fahrzeugs und gibt die berechnete Position und Geschwindigkeit an die Koordinatentransformation weiter. Von dort werden Position und Geschwindigkeit des Ego-Fahrzeugs an VTD übergeben. An den ACC- bzw. LKA-Regler werden benötigte Werte aus Position, Geschwindigkeit und Beschleunigung des Ego-Fahrzeugs übergeben. Der LKA-Regler benötigt die Geschwindigkeit, um zwischen den geschwindigkeitsabhängigen PID-Einstellungen zu wechseln. Dieses Modell dient als Ausgangsmodell, um die Simulationsumgebung für den TJA-Regler zu testen, da dieser auf den Einzelfunktionen ACC und LKA aufbaut. Die Testszenarios und Ergebnisse sind in Kapitel 3.3 erläutert. Die Test werden einmal mit und einmal ohne dem fahrstreifenabhängigen Sensormodell (später als LDACC, Lane Dependent ACC bezeichnet) ausgeführt.

2.3.2. TJA-Modell mit Platooningfunktion

Das TJA Modell erweitert die Grundfunktionen des ACC- und LKA-Modells durch gleichzeitige Regelung von 3 Ego-Fahrzeugen. Dazu wird das Grundmodell aus Kapitel 2.3.1 erweitert, indem die Elemente ACC, LKA, VSM und die Koordinatentransformation für 2 weitere Ego-Fahrzeuge erstellt werden. Das Element Sensormodell wird für 2 weitere Ego-Fahrzeuge erweitert, ebenso wird die Simulink-Element TJA-Controller

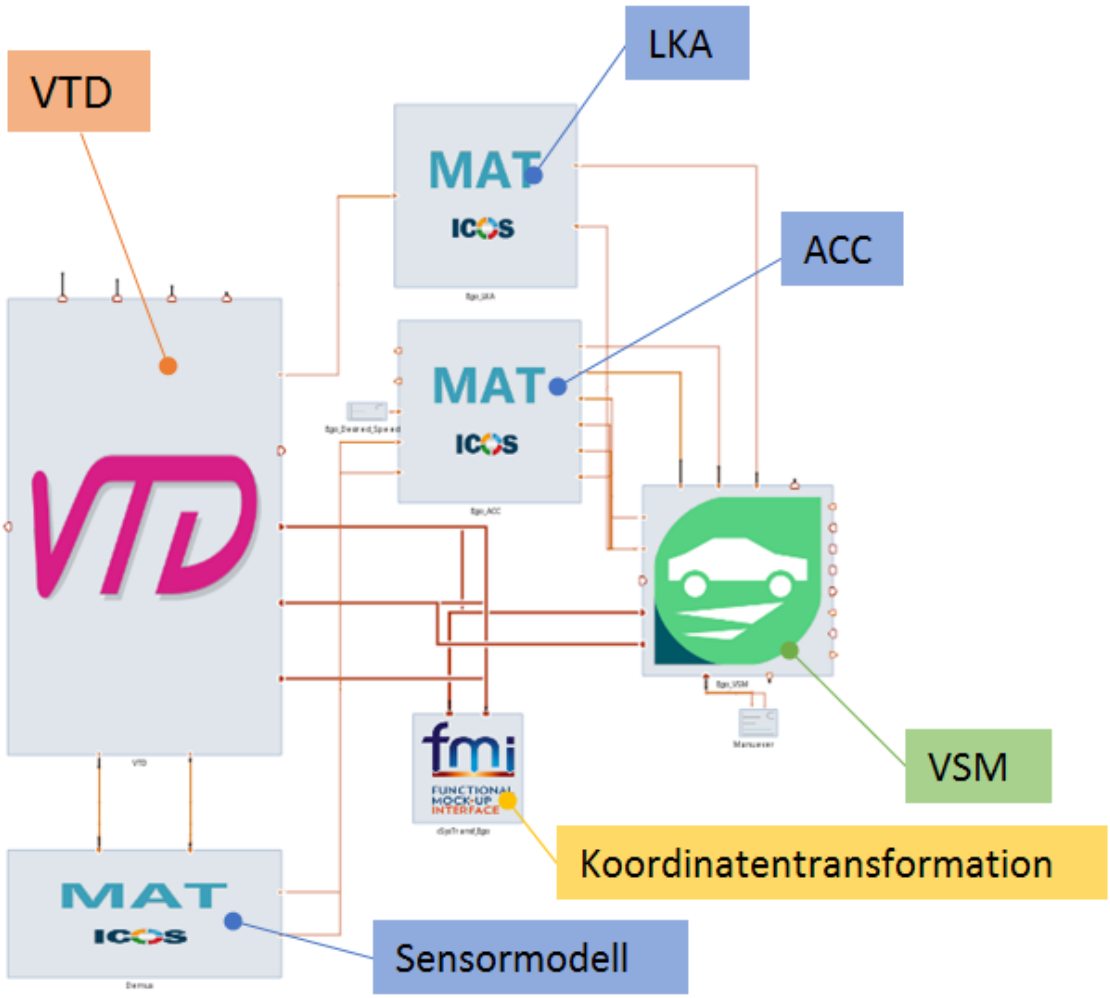

Abbildung 2.10.: ACC- und LKA-Modell in Model.CONNECT

hinzugefügt. Abbildung 2.12 zeigt das erweiterte Modell, wie es in Model.CONNECT aufgebaut ist. Die roten Blöcke der 3 Ego-Fahrzeuge sind analog zum Ego-Fahrzeug im ACC- und LKA-Modell aus Abbildung 2.10 aufgebaut. Die ACC- und LKA-Regler sind um die Platooningfunktionen des TJA erweitert. Das heißt, der ACC-Regler wurde um die Längssteuerung der Platooningfunktion erweitert und der LKA-Regler mit der Quersteuerung der Platooningfunktion. Durch den TJA-Controller wird zwischen den Funktionen umgeschaltet. Für dieses Modell wird das fahrstreifenabhängige Sensormodell gewählt. Die Ein- und Ausgänge der Blöcke des TJA-Modells sind zur besseren Übersicht in Abbildung 2.13 dargestellt. Alle Ein- und Ausgangskonfigurationen dieses Modells, bis auf die der Koordinatentransformation, sind im Anhang A.1 aufgelistet. Dieses Modell dient als Ausgangsmodell um den TJA-Regler zu testen. Die Testszenarios

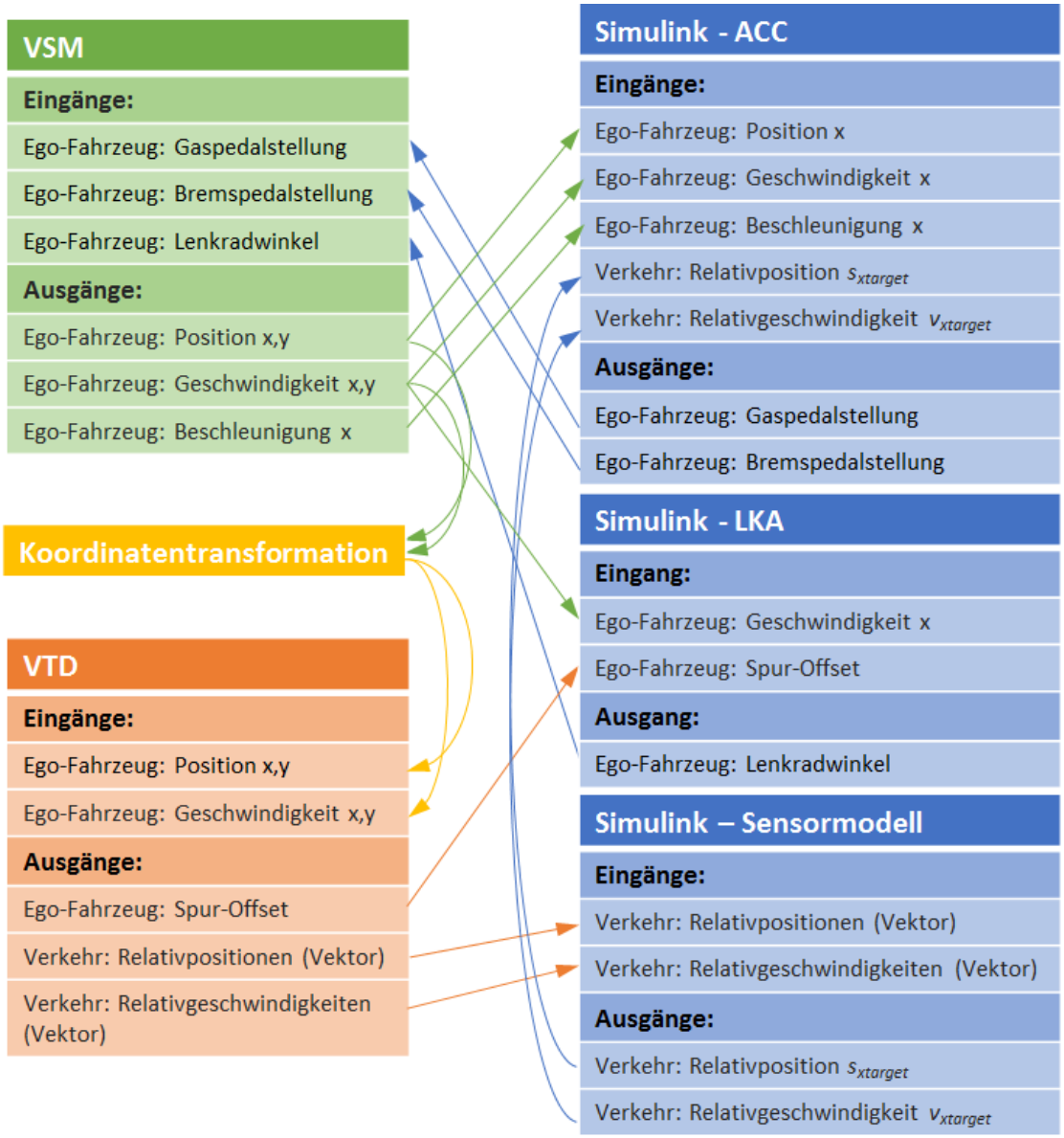

Abbildung 2.11.: Ein- und Ausgänge der Elemente im ACC- und LKA-Modell
Der Regelkreis besteht aus den 6 Einheiten VTD, VSM, ACC, LKA, Sensormodell und der Koordinatentransformation. VSM berechnet die dynamischen Zustände des Fahrzeugs, welche über eine Koordinatentransformation an VTD weitergeleitet werden. In VTD werden Sensoren definiert. Die Sensordaten werden vom Sensormodell aufbereitet und von den Reglern verwendet. Die Regler berechnen wiederum Gas- und Bremspedalstellungen sowie den Lenkradwinkel, mit denen VSM die dynamischen Zustände des Fahrzeugs simuliert.

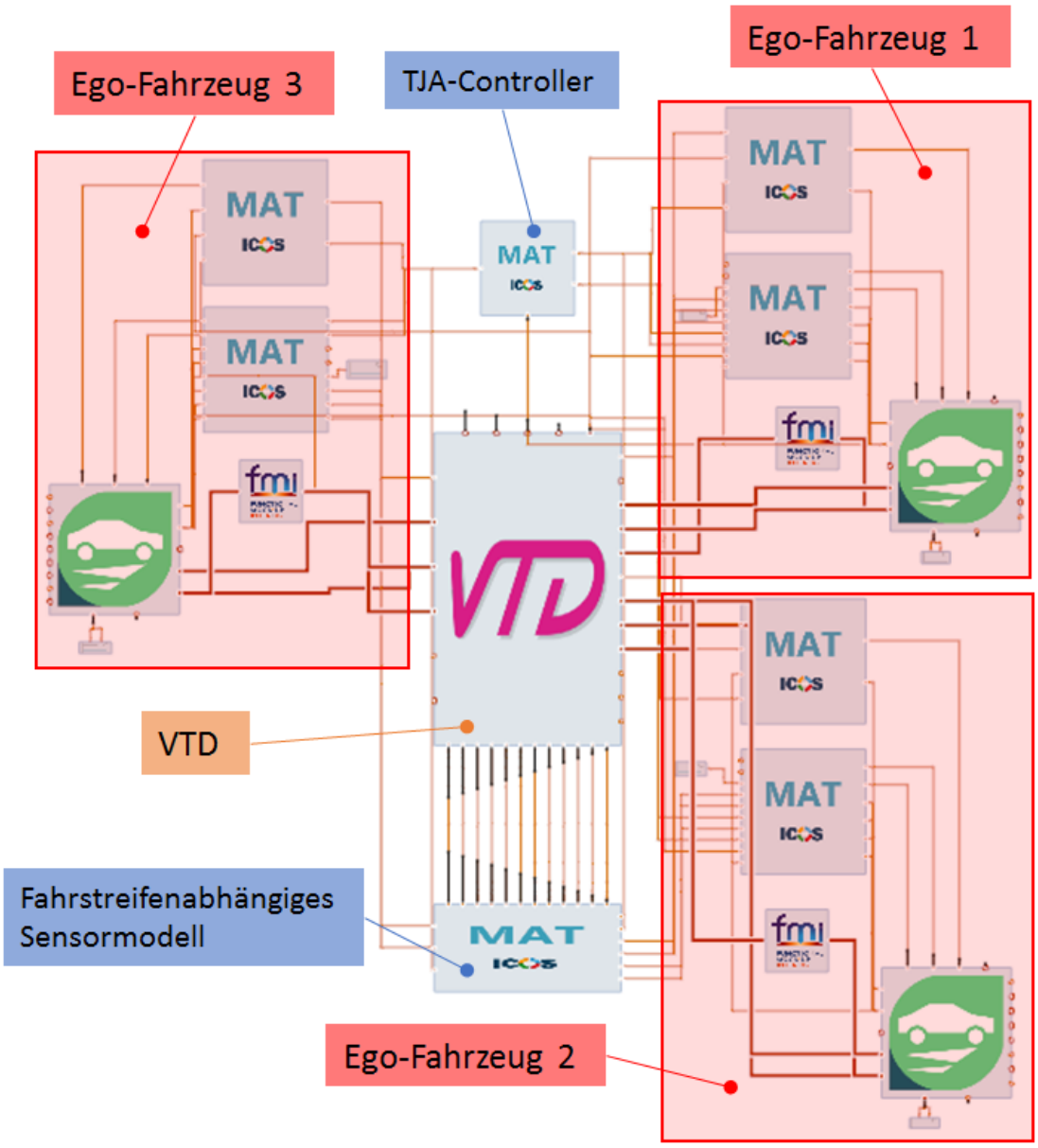

Abbildung 2.12.: TJA-Modell in Model.CONNECT

und Ergebnisse sind in Kapitel und erläutert. Die Ergebnisse des ENABLE-S3 Szenarios sind in Kapitel 3.5 zu finden.

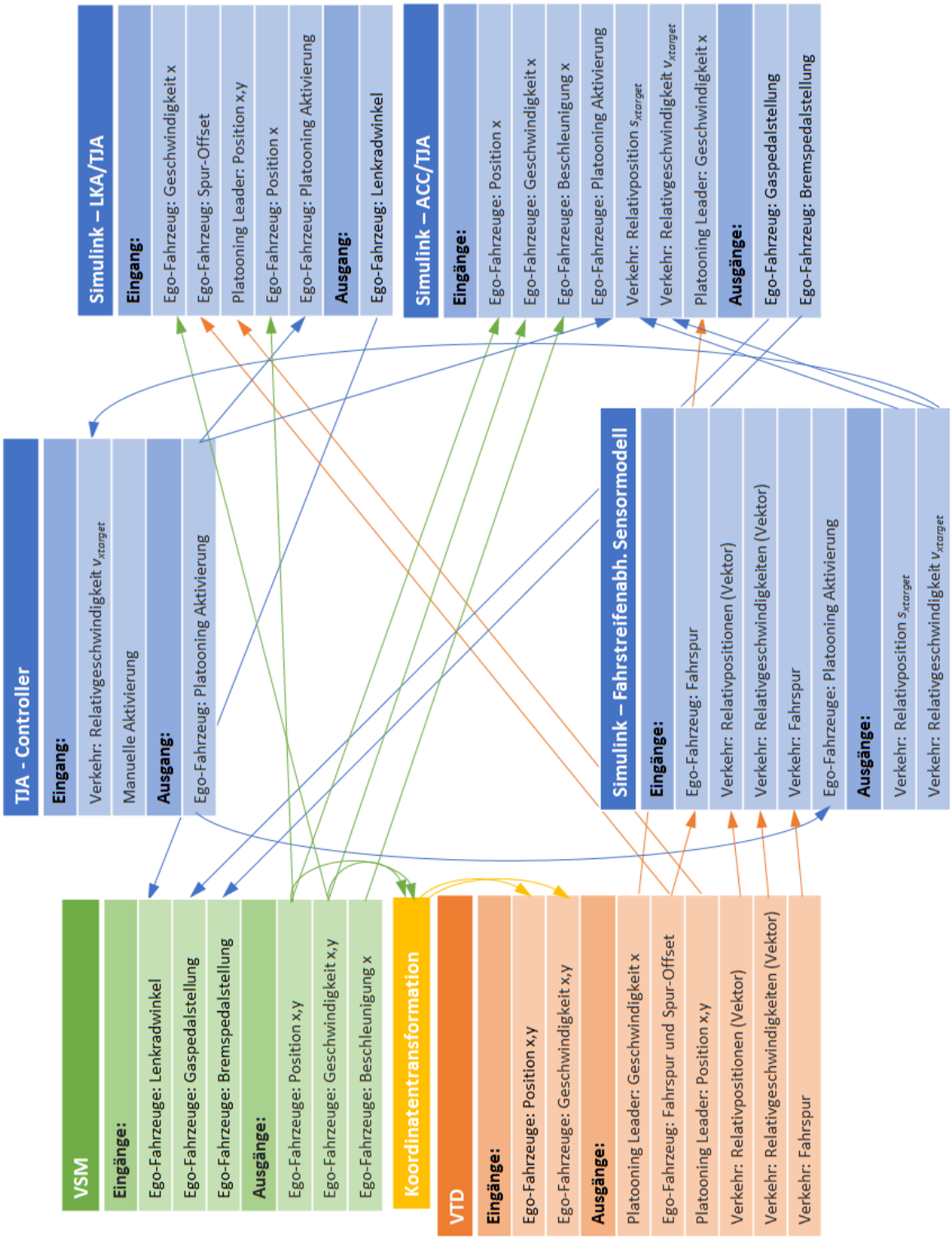

Abbildung 2.13.: Ein- und Ausgänge der Elemente im TJA-Modell

3. Szenarien und Resultate

Ziel der in diesem Kapitel durchgeführten Simulationen ist der Nachweis der Funktionsfähigkeit der aufgebauten Co-Simulationsumgebung und der darin enthaltenen Modelle und Regler.

3.1. Vergleich parallele und sequentielle Co-Simulation

Zu Beginn wird das Verhalten der Co-Simulation bei paralleler bzw. sequentieller Kopplung der Simulationselemente untersucht. Wie in Kapitel 1.3.1 Co-Simulation beschrieben, gibt es bei der sequentiellen Kopplung eine festgesetzte Reihenfolge der Simulationselemente. Die berechneten Daten werden in dieser Reihenfolge bei jedem Makrostep an das folgende Element weitergegeben. Das heißt Ausgangswerte eines Elements des Zeitschritts $T + \Delta T$ werden als Eingangswerte des folgenden Elements zum Zeitpunkt T verwendet. Bei der parallelen Kopplung berechnen alle Elemente ihren Simulationsschritt mit den Werten des letzten Datenaustauschs und leiten zu Beginn des nächsten Zeitschritts die errechneten Größen weiter.

Im folgenden Szenario wird das in Kapitel 2.3.1 beschriebene ACC- und LKA-Modell einmal sequentiell und einmal parallel simuliert. Das Szenario besteht aus einem Ego-Fahrzeug und einem Target-Fahrzeug welches auf einer geraden Fahrbahn auf 100 km/h beschleunigt, danach auf 50 km/h abbremst und anschließend wieder auf 70 km/h beschleunigt.Das Target-Fahrzeug wird von VTD berechnet. Das Ego-Fahrzeug folgt dem Target-Fahrzeug mit den ACC- und LKA Reglern. Nach dem Start der Co-Simulation werden alle beteiligten Simulationsprogramme gestartet. Diese Startzeit ist abhängig von der Größe des Modells und Anzahl der Elemente. Das Modell in diesem Versuch startet bei paralleler Simulation nach 45 Sekunden und bei sequentieller Simulation nach 42 Sekunden mit der eigentlichen Simulation des Szenarios. Die Simulation ist so eingestellt, dass das Szenario nach 130 Sekunden gestoppt wird. Die parallele Simulation dauert insgesamt 85 Sekunden und die sequentielle 88 Sekunden. Das heißt die Simulation läuft beschleunigt ab und erlaubt hiermit eine spätere Echtzeitanwendung in einem Fahrsimulator.

Der Vergleich der Geschwindigkeit des Ego-Fahrzeugs bei paralleler und sequentieller Co-Simulation sind in Abbildung 3.1 über der Zeit dargestellt. Die Differenz der Geschwindigkeit zwischen paralleler und sequentieller Simulation ist maximal 0,142 km/h und bleibt ab Sekunde 25 in einem Rahmen von -0,01 bis 0,01 km/h.

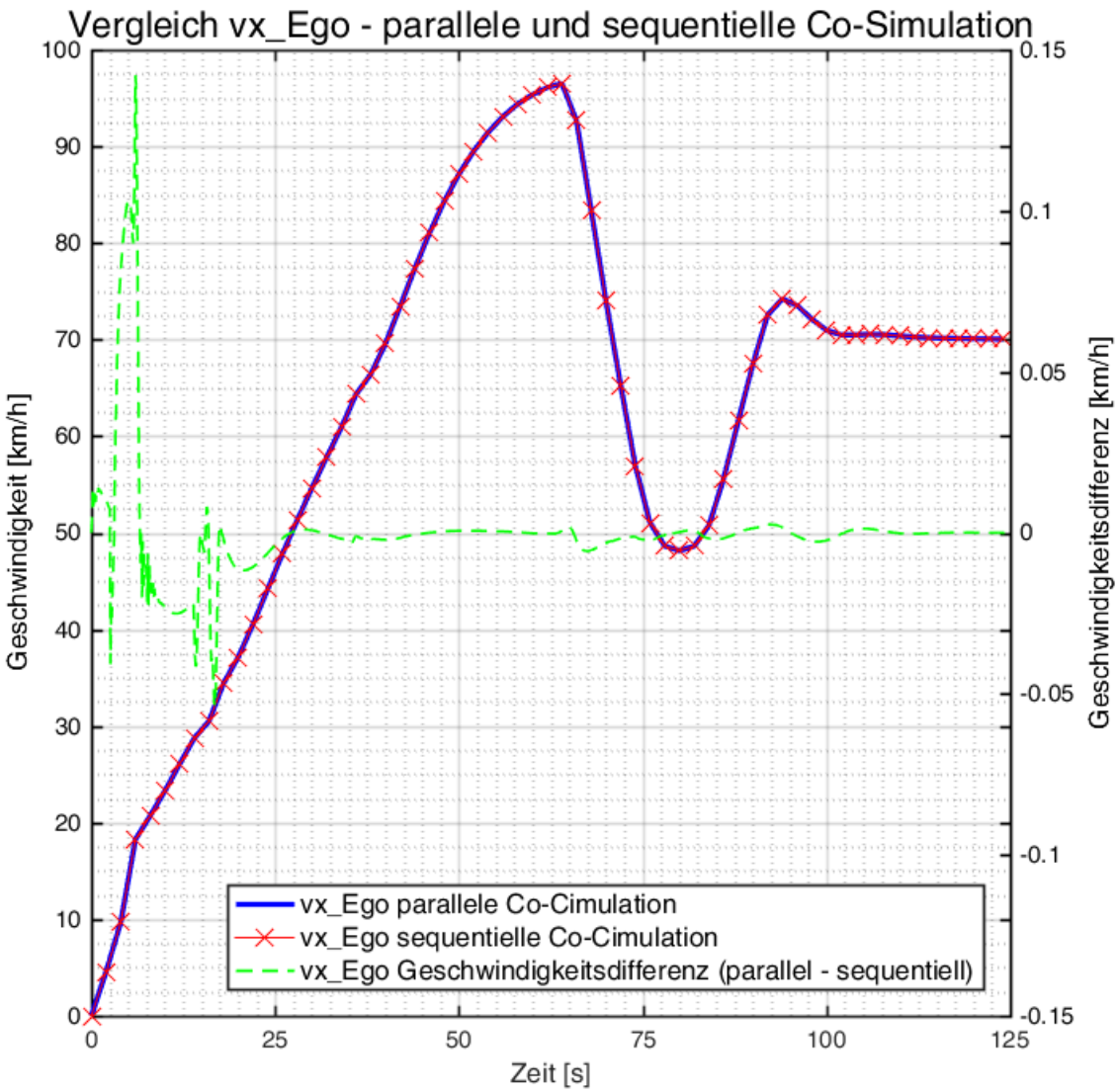

Abbildung 3.1.: Vergleich parallele und sequentielle Co-Simulation

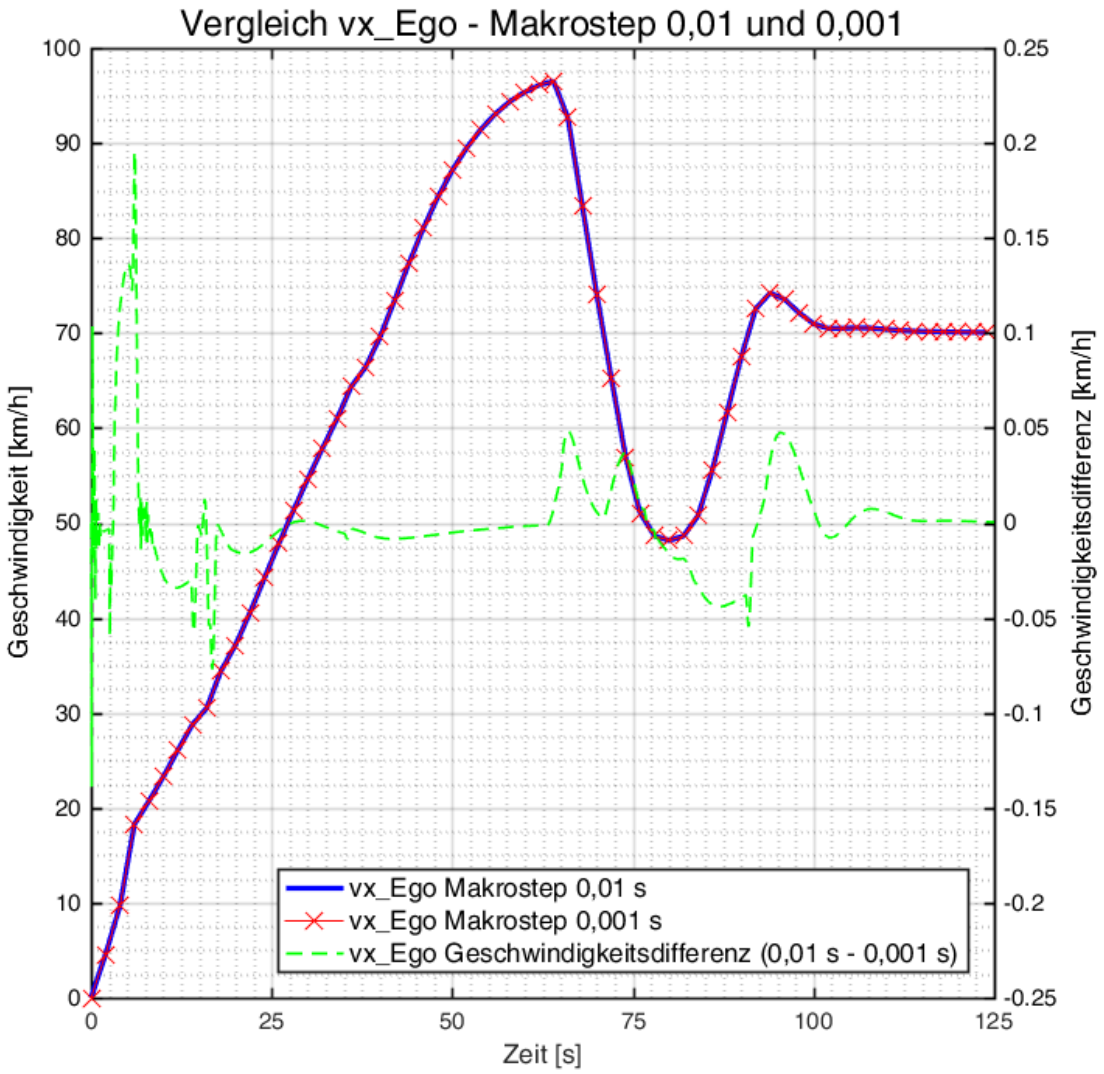

Abbildung 3.2.: Vergleich Makrostep 0,01 s und 0,001 s

3.2. Variation des Makrosteps

Nach den Simulaitonsversuchen in Kapitel 3.1 wird das gleiche Szenario aus diesem Kapitel mit einer Variation des Makrosteps simuliert. Es werden die Makrosteps von 0,01 s auf 0,1 s und 0,001 s variiert. Bei einem Makrostep von 0,1 s startet zwar die Simulation, aber das Target-Fahrzeug führt die vorgegebene Szenariendefinition nicht aus und bleibt stehen. Möglicherweise ist der Makrostep von 0,1 s für VTD zu groß gewählt. In Abbildung 3.2 ist daher nur der Vergleich zwischen den Makrosteps 0,01 und 0,001 dargestellt.

Abbildung 3.2 zeigt die Geschwindigkeitsverläufe der Ego-Fahrzeugs mit dem Szenario aus Kapitel 3.1 mit den Makrosteps von 0,01 s und 0,001 s. Die Geschwindigkeitsverläufe unterscheiden sich um maximal 0,2 km/h. Der wesentliche Unterschied zwischen den beiden Simulationen ist die Simulationszeit. Die Simulation mit einem Makrostep von 0,01 s dauert 85 Sekunden, während die Simulation mit dem Makrostep von 0,001 s 185 Sekunden dauert. Die Peaks im Verlauf der Geschwindigkeitsdifferenz beim Start der Simulation sind auf die genaueren Extrapolationswerte der Variante mit einem Makrostep von 0,001 s zurückzuführen. Weitere Peaks sind ab Sekunde 60 zu erkennen. Diese fallen in den Bereich der Änderung der Beschleunigungswerte des Target-Fahrzeugs.

3.3. ACC/LKA - Szenarien

Das ACC- und LKA-Modell aus Kapitel 2.3.1 wird mit den Szenarien „Target Changes Lane“ 3.3 und „Cut-into-Lane Test“ 3.7 verifiziert. Messdaten eines Fahrzeugs mit ACC System zu diesen Fahrmanövern wurden vom Institut für Fahrzeugtechnik der TU Graz bei Benchmarktests gesammelt, [Nov16]. Das Verhalten des Ego-Fahrzeugs wird von VSM mit den Ausgangswerten der ACC- und LKA-Reglern berechnet. Den anderen Fahrzeugen (Target1 und Target2) sind VTD Fahrermodelle zugewiesen. Sie führen die im Szenario-Editor vorgegebenen Manöver wie Geschwindigkeitsänderungen oder Fahrstreifenwechsel aus.

3.3.1. Target Changes Lane - Test (TCL)

Bei diesem Fahrmanöver fährt das Ego-Fahrzeug hinter dem Target1-Fahrzeug, siehe Abbildung 3.3. Target1 fährt hinter dem Target2-Fahrzeug. Anschließend verringert Target2 seine Geschwindigkeit und Target1 überholt das Target2-Fahrzeug. Es wird getestet, ob und wie der ACC-Regler des Ego-Fahrzeugs sich der Geschwindigkeit des neuen Target-Fahrzeugs (Target2) anpasst. Dieses Szenario wird einmal ohne und einmal mit dem fahrstreifenabhängigen Sensormodell des ACC-Reglers (LDACC, Lane Dependant Adaptive Cruise Control) simuliert.

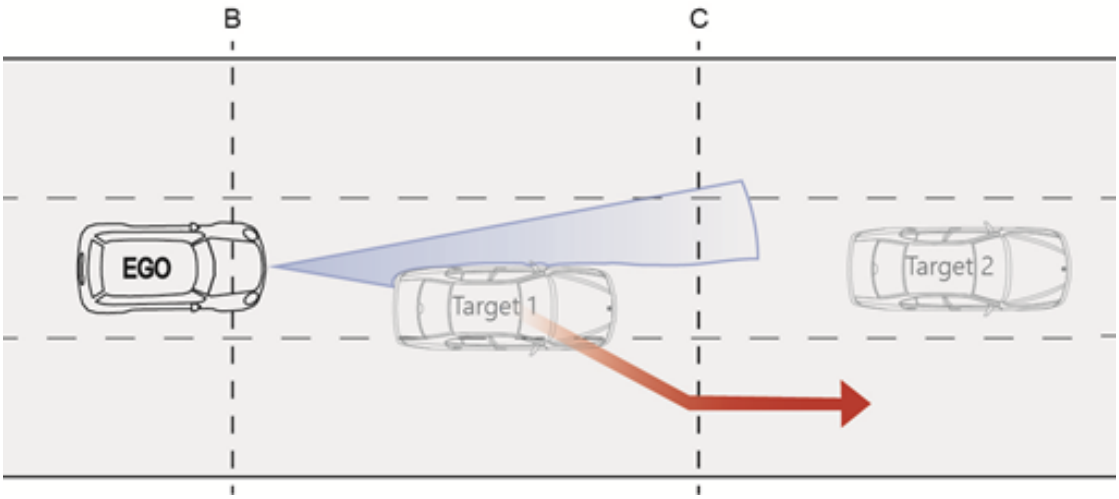

Abbildung 3.3.: Target Changes Lane - Test, [Nov16]

Bei den Messfahrten wurde das Manöver mit folgenden Parametern ausgeführt:

- **Ausgangssituation**
  - Ego — folgt Target1 mit ACC
  - Geschwindigkeit Target1: — 50 km/h
  - Geschwindigkeit Target2: — 50 km/h
  - Abstand Target2 zu Target1: — 24,3 m
- **ACC-Einstellungen**
  - ACC Zeitabstand zum Target: — 2,25 s
  - ACC Mindestabstand zum Target: — 12 m
- **Verzögerung Target2**
  - Mittlere Verzögerungsrate Target2: — 1,85 m/s^2
  - Endgeschwindigkeit Target2: — 10 km/h

Diese Parameter wurden auf das Simulationsszenario übertragen. Zeitlich werden die Messfahrt und die Simulation durch den gleichen Startpunkt der Verzögerung des Target2-Fahrzeugs synchronisiert. Der horizontale Öffnungswinkel des Radarsensors im Simulationsversuch beträgt 20°. In Abbildung 3.4 und Abbildung 3.5 sind die Ergebnisse des Szenarios dargestellt. Abbildung 3.4 vergleicht die Geschwindigkeiten der Messfahrt (a), der Simulation ohne LDACC (fahrstreifenabhängiges Sensormodell) (b) und der Simulation mit LDACC (c). Abbildung 3.5 zeigt die Beschleunigung der beteiligten Fahrzeuge der Messfahrt (a), der Simulation ohne LDACC (b) und der Simulation mit LDACC (c).

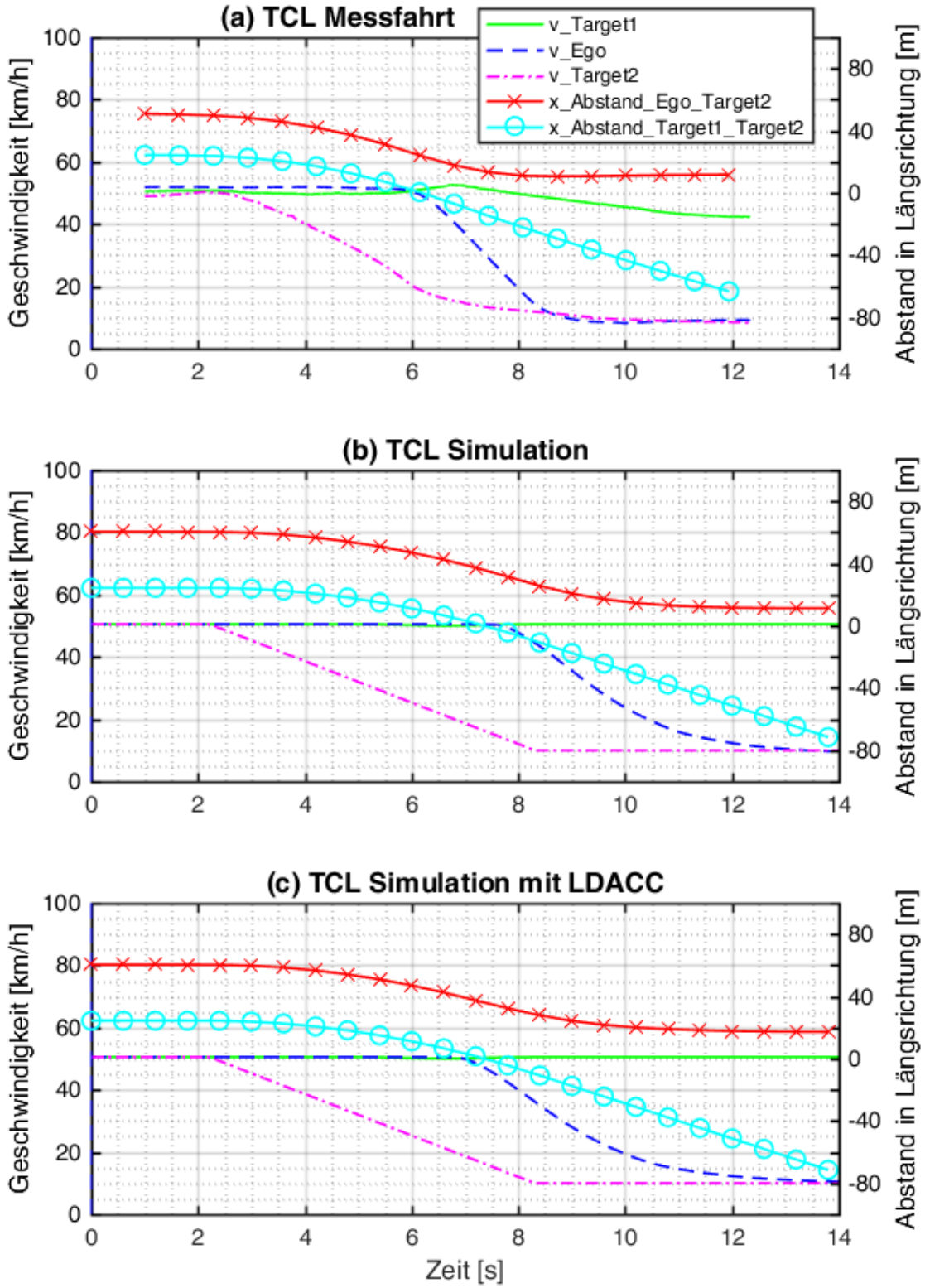

Abbildung 3.4.: Vergleich der Geschwindigkeiten und Abstände über der Zeit (TCL)

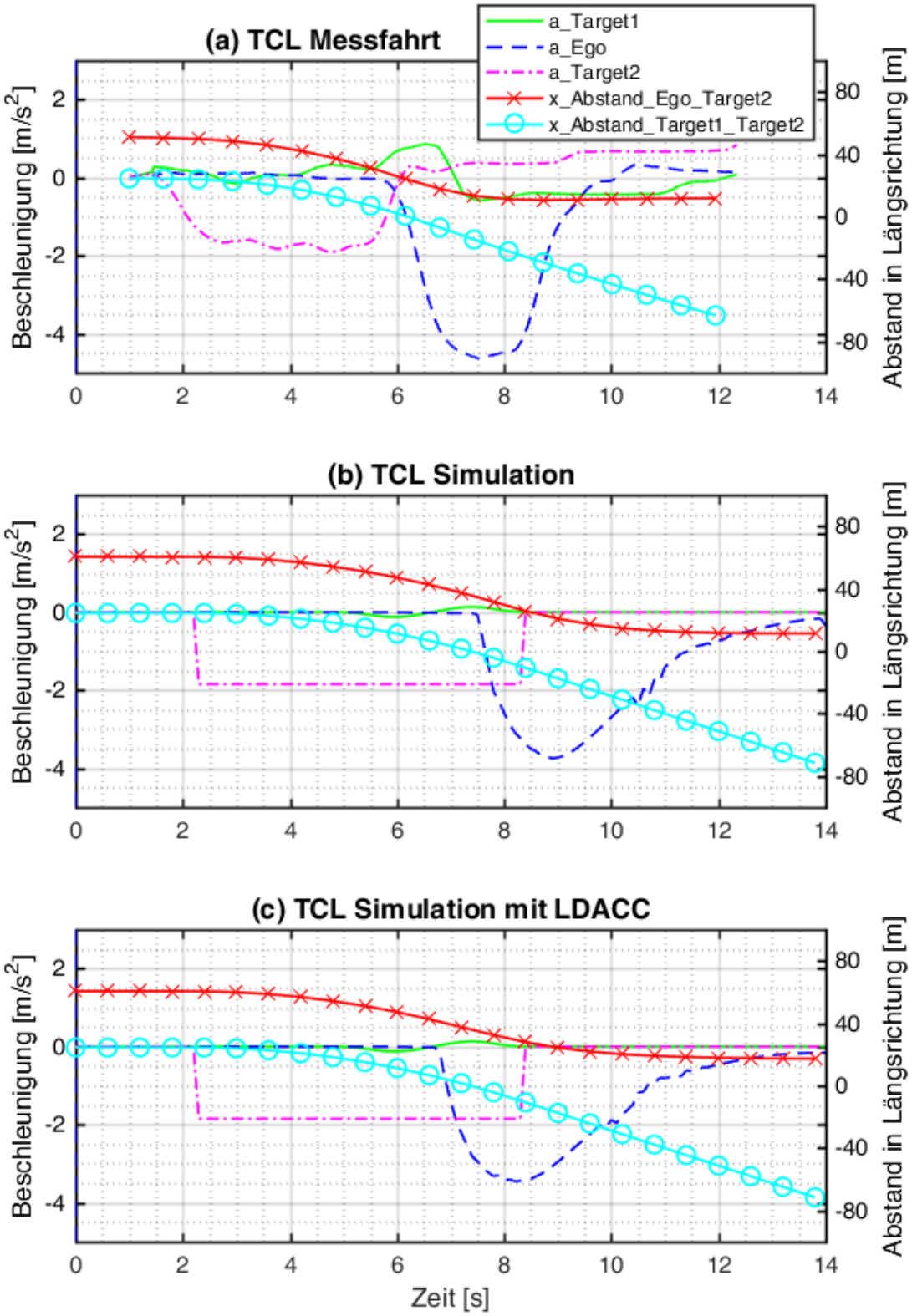

Abbildung 3.5.: Vergleich der Beschleunigungen und Abstände über der Zeit (TCL)

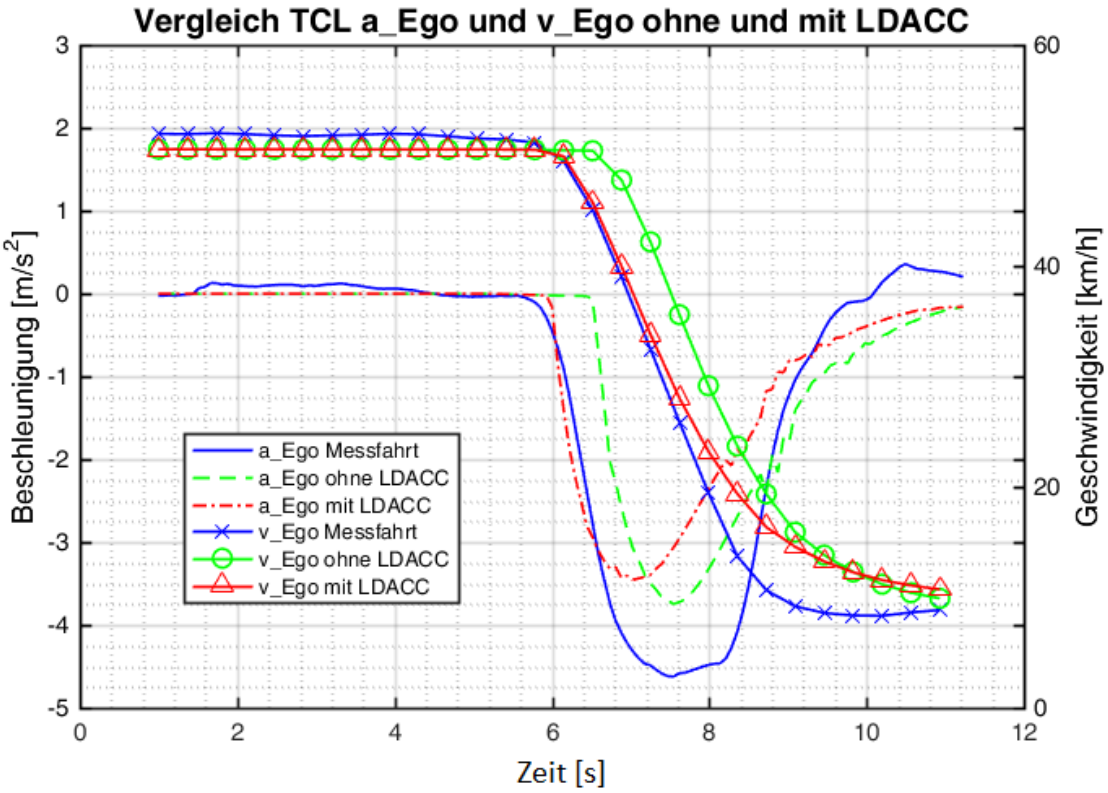

Abbildung 3.6.: Vergleich der Beschleunigungen und Geschwindigkeiten (TCL)

Ein Vergleich der Beschleunigung und Geschwindigkeiten der Messfahrt, der Simulation ohne LDACC und der Simulation mit LDACC der Ego-Fahrzeuge des TCL-Tests ist in Abbildung 3.6 dargestellt. Die Unterschiede erklären sich durch die unterschiedlichen Reglereinstellungen zwischen dem Fahrzeug der Messfahrt und dem ACC-Regler der Simulation. Die zeitliche Versetzung der Beschleunigungsverläufe der Simulation sind in der Verwendung der unterschiedlichen Sensormodelle begründet. Das Ego-Fahrzeug bei der Simulation mit LDACC bremst früher als das Ego-Fahrzeug bei der Simulation ohne LDACC, da vom Zielauswahlalgorithmus früher auf das neue Target2-Fahrzeug umgeschaltet wird.

3.3.2. Cut into Lane - Test (CIL)

In diesem Szenario (Abbildung 3.7) fährt das Ego-Fahrzeug dem Target1-Fahrzeug auf dem mittleren Fahrstreifen hinterher. Target1 überholt ein langsameres Target2-Fahrzeug, welches auf dem rechten Fahrstreifen fährt. Hat Target1 Target2 überholt, wechselt Target2 den Fahrstreifen und positioniert sich zwischen dem Ego-Fahrzeug und Target1. Es wird getestet, ob und wie der ACC-Regler des Ego-Fahrzeugs sich der Geschwindigkeit

des neuen Target-Fahrzeugs (Target2) anpasst.

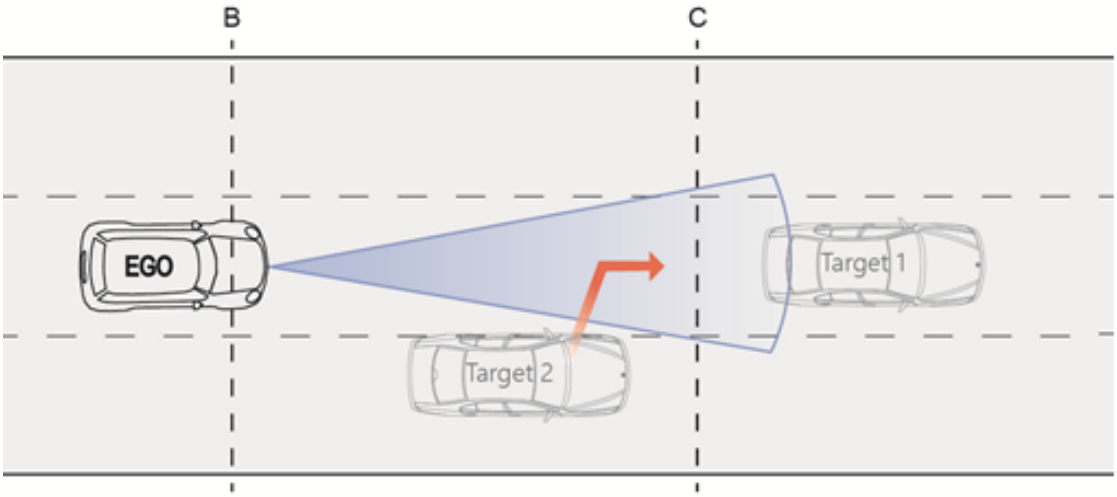

Abbildung 3.7.: Cut into Lane - Test, [Nov16]

Bei den Messfahrten wurde das Manöver mit folgenden Parametern ausgeführt:

- **Ausgangssituation**
  - Ego — folgt Target1 mit ACC
  - Geschwindigkeit Target1: — 60,5 km/h
  - Geschwindigkeit Target2: — 40 km/h
- **ACC-Einstellungen**
  - ACC Zeitabstand zum Target: — 2,25 s
  - ACC Mindestabstand zum Target: — 12 m

Diese Parameter wurden wieder auf das Simulationsszenario übertragen. Der horizontale Öffnungswinkel des Radarsensors im Simulationsversuch beträgt 20°. Zur Synchronisation der Messfahrt mit der Simulation wird jener Zeitpunkt verwendet, wo Target1 und Target2 die gleiche x-Position besitzen und somit auf gleicher Höhe sind. In Abbildung 3.8 und Abbildung 3.9 sind die Ergebnisse des Szenarios dargestellt. Abbildung 3.8 vergleicht die Geschwindigkeiten der Messfahrt (a), der Simulation ohne LDACC (fahrstreifenabhängiges Sensormodell) (b) und der Simulation mit LDACC (c). Abbildung 3.9 zeigt die Beschleunigung der beteiligten Fahrzeuge der Messfahrt (a), der Simulation ohne LDACC (fahrstreifenabhängiges Sensormodell) (b) und der Simulation mit LDACC (c).

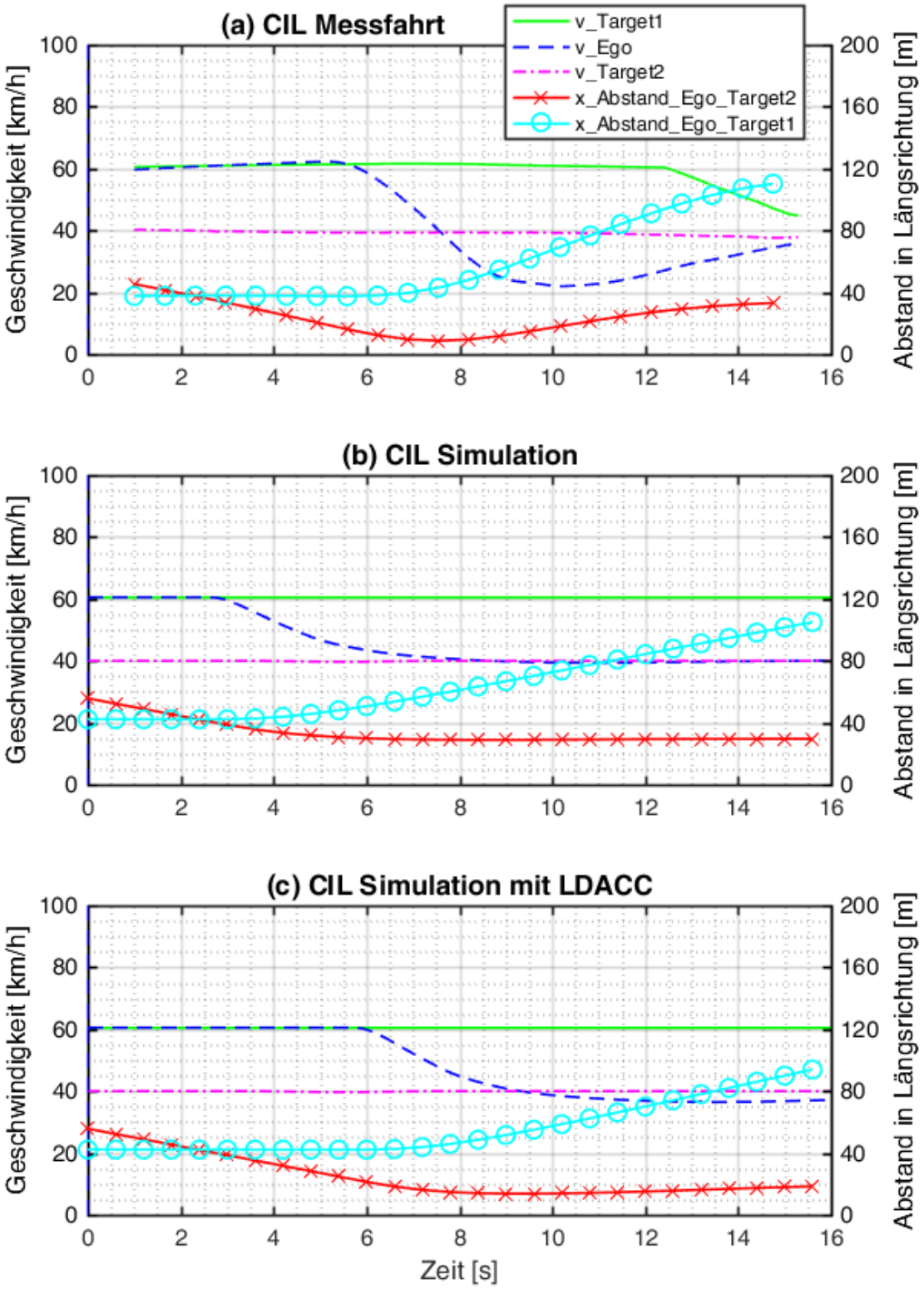

Abbildung 3.8.: Vergleich der Geschwindigkeiten und Abstände über der Zeit (CIL)

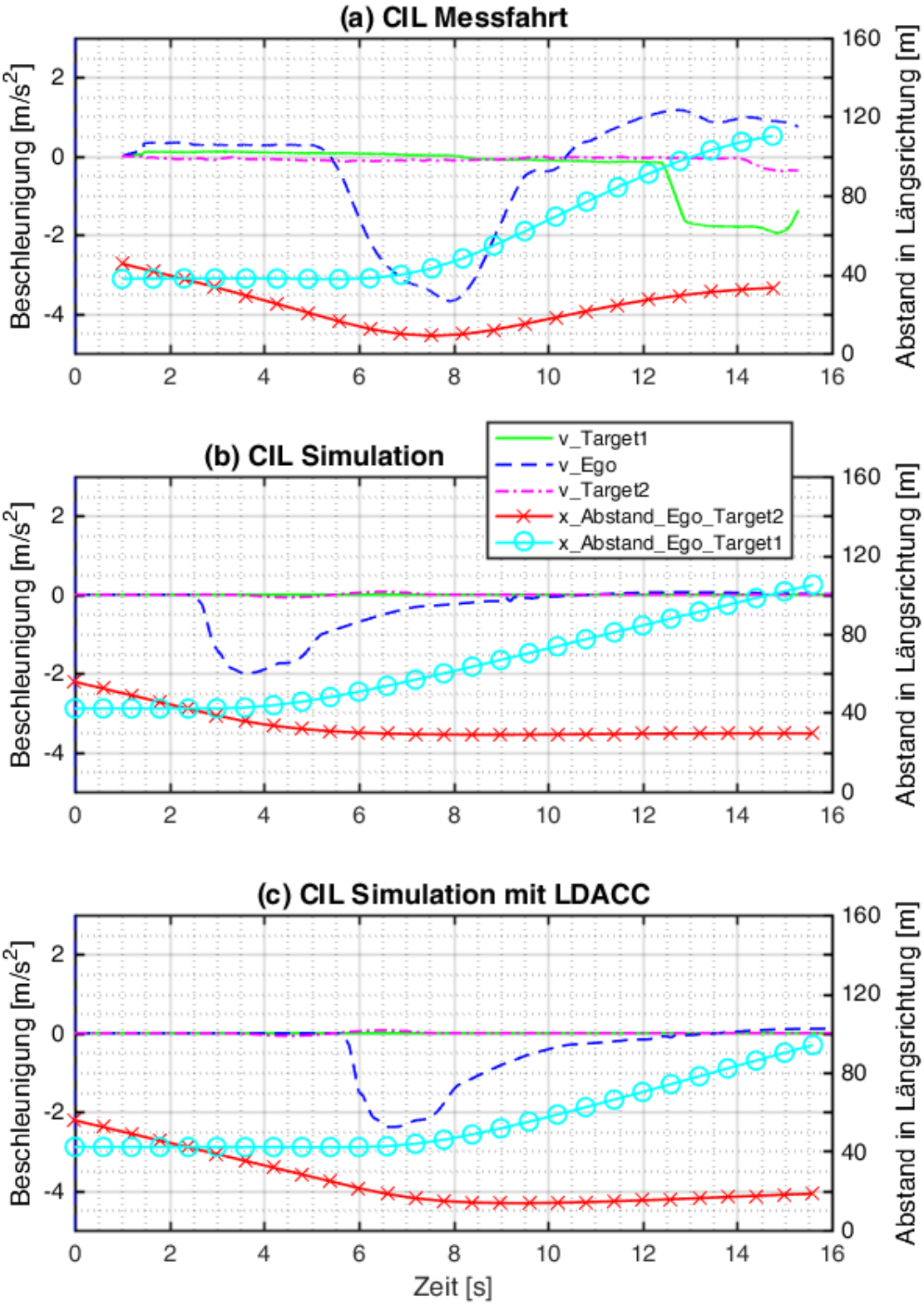

Abbildung 3.9.: Vergleich der Beschleunigungen und Abstände über der Zeit (CIL)

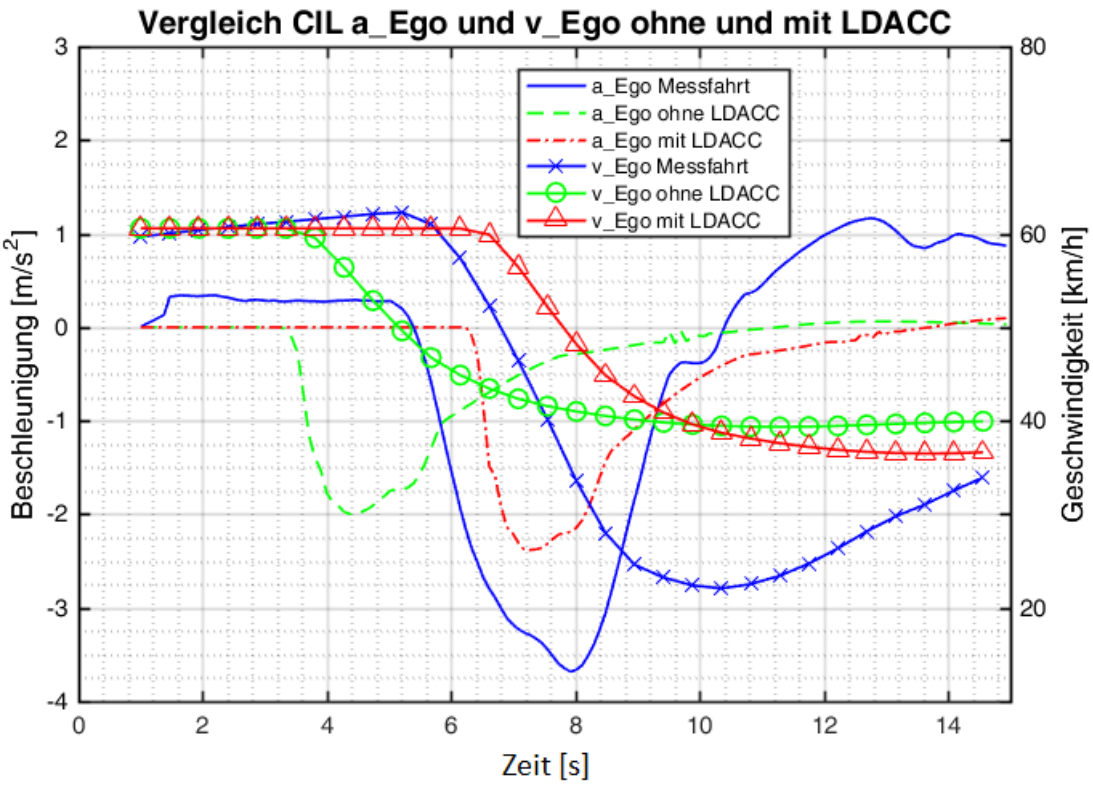

Abbildung 3.10.: Vergleich der Beschleunigungen und Geschwindigkeiten (CIL)

Ein Vergleich der Beschleunigung und Geschwindigkeit der Messfahrt, der Simulation ohne LDACC und der Simulation mit LDACC der Ego-Fahrzeuge des CIL-Tests ist in Abbildung 3.10 dargestellt. Das Fahrzeug in der Simulation ohne fahrstreifenabhängigem Sensormodell bremst wesentlich früher, da der Zielauswahlalgorithmus dem ACC-Regler die Daten des Target2-Fahrzeugs übergibt, sobald die Target-Fahrzeuge auf gleicher Höhe sind.

3.4. TJA mit Platooning

Dieses Kapitel zeigt die Ergebnisse der Testszenarien des in Kapitel 2.3.2 beschriebenen Modell TJA-Modells mit Platooningfunktion. Dieses Modell basiert auf dem ACC- und LKA-Modell und wurde um die Platooningfunktion und dem TJA-Controller erweitert. Für die Auswahl des Targets für den ACC-Regler wurde das fahrstreifenabhängige Sensormodell verwendet.

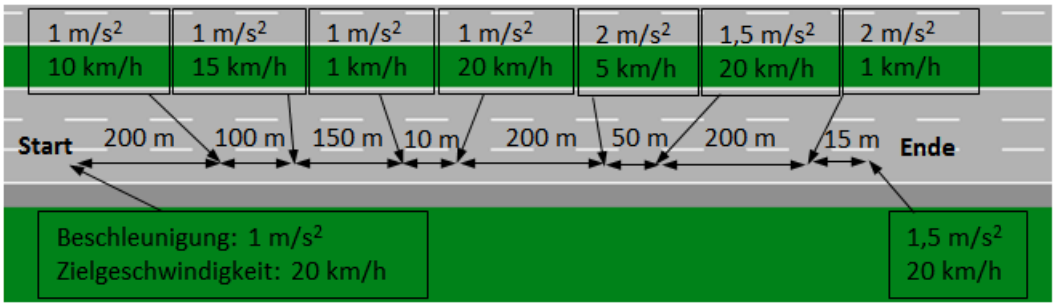

Abbildung 3.11.: Szenarioabfolge Platooning Längsdynamik

3.4.1. Platooning Längsdynamik

In diesem Kapitel wird die Längsdynamik des TJA mit Platooningfunktion untersucht. Bei aktiver Platooningfunktion, d.h. der TJA-Controller wurde manuell aktiviert. Ist die Platooning-Leader Geschwindigkeit $\leq$ 20 km/h, berechnet der ACC-Regler den Abstand zum Vorderfahrzeug nicht mehr mit den Sensordaten des Ego-Fahrzeugs, sondern mit den Daten der V2V-Kommunikation. Durch gleichmäßigere Geschwindigkeiten und geringere Abstände soll ein höherer Verkehrsdurchsatz erreicht werden. Es wird ein Szenario (Abbildung 3.11 und Abbildung 3.12) erstellt, in dem der Platooning-Leader mehrere Geschwindigkeitsänderungen in den Grenzen der Platooningfunktion (0 bis 20 km/h) durchführt. Dieses Szenario wird einmal ohne (a) und einmal mit (b) aktiver Platooningfunktion durchgeführt, um die erwarteten Vorteile der Platooningfunktion zu verifizieren. Die Zeitlücke wird bei aktiver Platooningfunktion von 3 auf 1 s (nach [22109]) heruntergesetzt. Der Mindestabstand zwischen den Fahrzeugen ist mit 10 m eingestellt. Wird der Abstandskorridor zum direkten Vorderfahrzeug im Platoon von 4 bis 18 m verlassen (siehe Kapitel 2.2.3), wird nicht mehr die Geschwindigkeit des Platooning-Leaders zur Abstandsregelung verwendet, sondern wieder die des direkten Vorderfahrzeugs. Da heißt die Platooningfunktion in Längsrichtung wird kurzzeitig deaktiviert und das ACC-System wieder aktiviert.

Die Ergebnisse der beiden Szenariovarianten (a) und (b) sind in Abbildung 3.13 dargestellt. Es wird die Längsgeschwindigkeit jedes Fahrzeugs im Platoon dargestellt. Sie zeigen, dass den Geschwindigkeiten des ersten Fahrzeug im Platoon bei aktiver Platooningfunktion wesentlich exakter gefolgt wird als ohne Platooningfunktion. Die Namensgebung der Fahrzeuge im Platoon ist aus Abbildung 3.12 ersichtlich.

Für den Verkehrsfluss ist ein möglichst konstanter Abstand zwischen den Fahrzeugen im Platoon entscheidend. Dazu wird in Abbildung 3.14 der Abstand vom Platooning Leader zum Ego3 Fahrzeug (siehe Abbildung 3.12) der beiden Szenarienvarianten (a) und (b) gegenübergestellt. Beim Platooning (b) sind die Abstände zwischen Platooning-Leader und Ego3-Fahrzeug wesentlich geringer und konstanter als bei Variante (a) im ACC-Modus.

Abbildung 3.12.: Fahrzeuge im Platoon

Abbildung 3.14.: Abstand Platooning-Leader zu Ego3 ohne und mit Platooningfunktion
Die Vorteile der Platooningfunktion werden in dieser Abbildung deutlich. Durch den konstanteren Abstand zwischen Platooning-Leader und Ego-Fahrzeug werden die in der Einleitung beschriebenen Vorteile der Platooningfunktion ermöglicht.

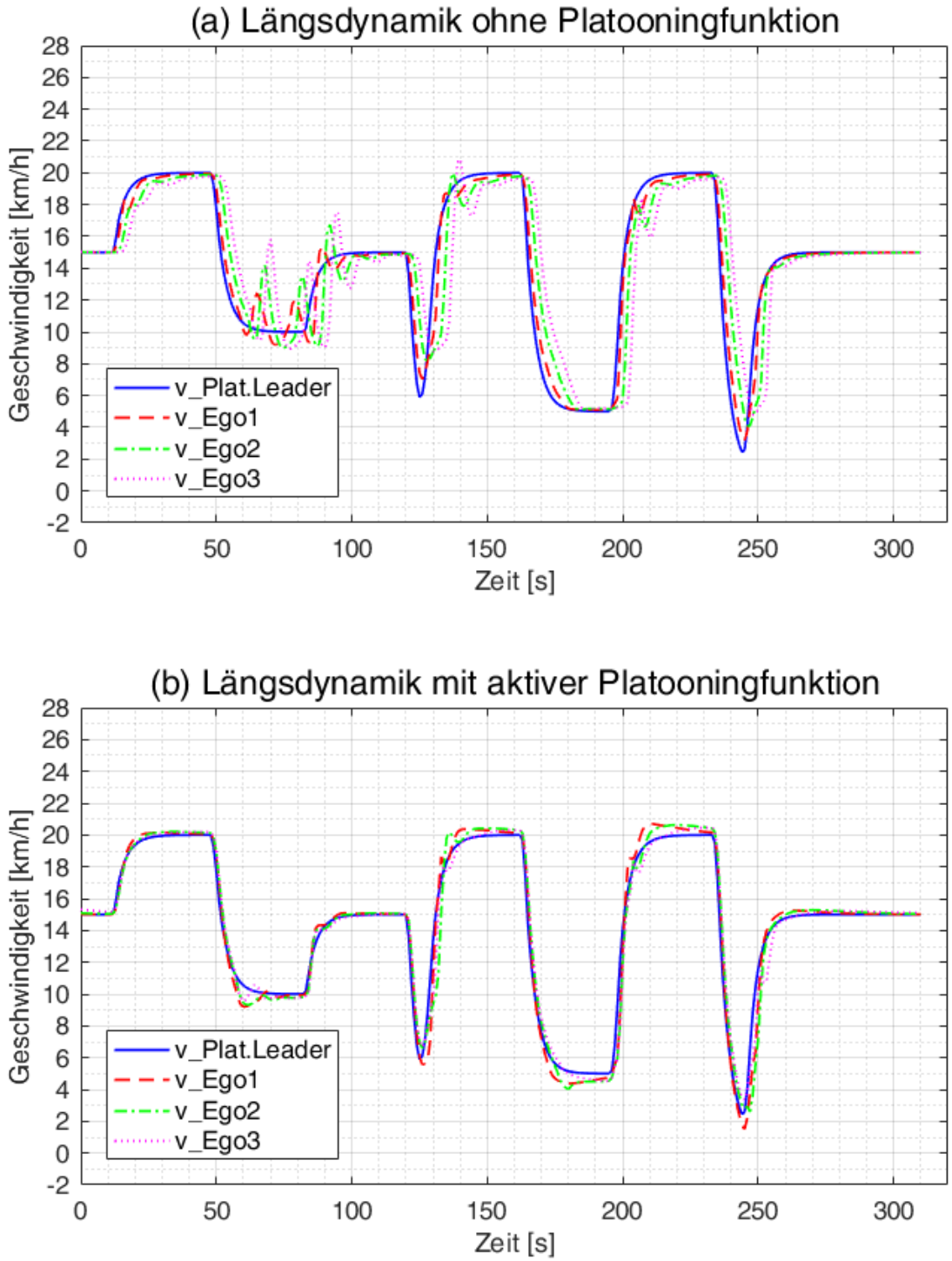

Abbildung 3.13.: Vergleich der Geschwindigkeiten im Szenario Platooning Längsdynamik: (a) ohne Platooningfunktion, (b) mit aktiver Platooningfunktion

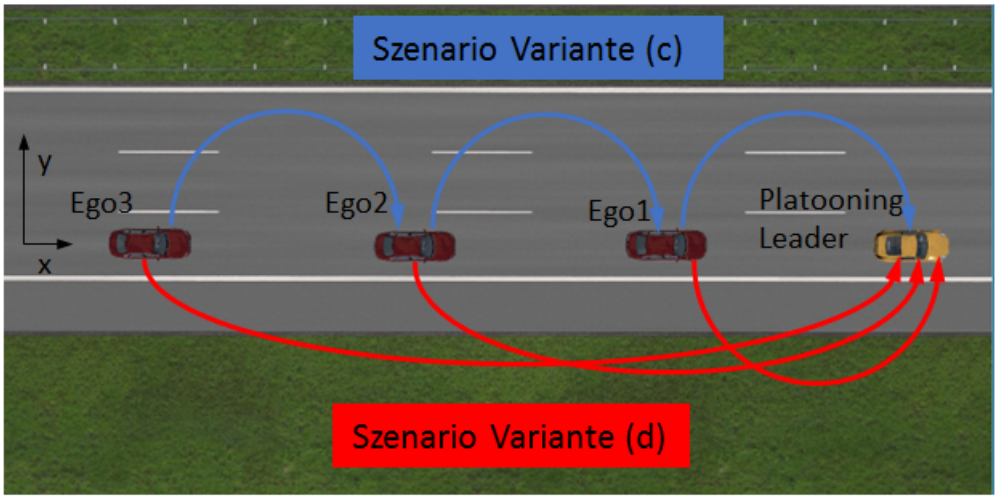

Abbildung 3.15.: Szenariovarianten Platooning Querdynamik: (c) vorderes Fahrzeug ist Target, (d) Platooning Leader ist Target

3.4.2. Platooning Querdynamik

In diesem Kapitel wird die Querdynamik des in Kapitel 2.3.2 beschriebenen TJA-Modells mit Platooningfunktion untersucht. Ziel ist es, dass bei aktiver Platooningfunktion die Platooning-Fahrzeuge möglichst genau der y-Position (siehe Abbildung 3.15) des Target-Fahrzeugs folgen. Dazu wird ein Szenario erstellt in dem der Platooning-Leader bei konstanter Geschwindigkeit von 15 km/h mehrere Fahrstreifenwechsel durchführt. Die Platooningfunktion ist aktiv, d.h. der TJA-Controller wurde manuell aktiviert und die Platooning-Leader Geschwindigkeit ist $\leq$ 20 km/h. Die Zeitlücke zwischen den Fahrzeugen beträgt 1 s bei einem Mindestabstand von 10 m. Das Szenario wird, wie in Abbildung 3.15 gezeigt, auf zwei unterschiedliche Varianten getestet:

- Das Target-Fahrzeug jedes Ego-Fahrzeugs ist das vor ihm fahrende Fahrzeug (c)
- Das Target-Fahrzeug jedes Ego-Fahrzeugs ist der Platooning-Leader (d)

Das Szenario ist eine 3-spurige Fahrbahn. Der Platooning-Leader ist ein von VTD berechneter Fahrer, welcher die im Szenario-Editor bestimmten Fahrmanöver ausführt. Die drei Ego-Fahrzeuge (Ego1, Ego2 und Ego3) folgen diesem Fahrzeug. Der Platoon startet auf dem rechten Fahrstreifen mit einer konstanten Geschwindigkeit von 15 km/h. Es folgen die Fahrstreifenwechsel in der Reihenfolge links, rechts, links, links in einem Abstand von jeweils 200 m gefolgt von einer 300 m langen Fahrt auf dem linken Fahrstreifen. Danach folgen weitere Fahrstreifenwechsel in der Reihenfolge rechts, links, rechts und rechts in einem Abstand von jeweils 100 m (vgl. Abbildung 3.16). Am Ende befinden sich die Fahrzeuge wieder auf dem rechten Fahrstreifen. Die Fahrstreifenwechsel des Platooning-Leaders dauern 4.3 Sekunden. Dieser Wert stammt aus [SAKE16], Seite 96.

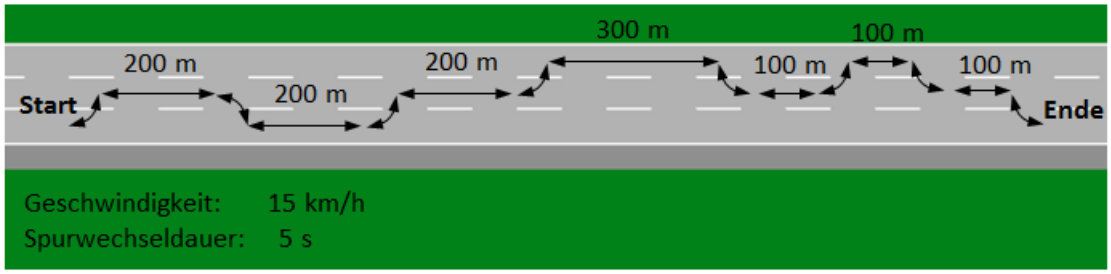

Abbildung 3.16.: Fahrstreifenwechselabfolge im Szenario Platooning Querdynamik

Die Ergebnisse der beiden Szenarienvarianten (c) und (d) sind in Abbildung 3.17 dargestellt. Es wird die y-Position jedes Fahrzeugs im Platoon dargestellt. Die y-Position 0 wird in die Mitte des rechten Fahrstreifens gelegt. Es ist zu sehen, dass die Fahrzeug Ego 1-3 , welche mit dem TJA-Regler ausgestattet sind, dem Platooning-Leader erst nach der im Regler vorgegebenen Zeitabstand folgen. Sie folgen allerdings nicht exakt der vorgegebenen Spur des Platooning-Leaders. Dies ist auf Verzögerungen des PID-Reglers zurückzuführen. Der Verzögerungseffekt ist in Variante (c) stärker ausgeprägt als in Variante (d).

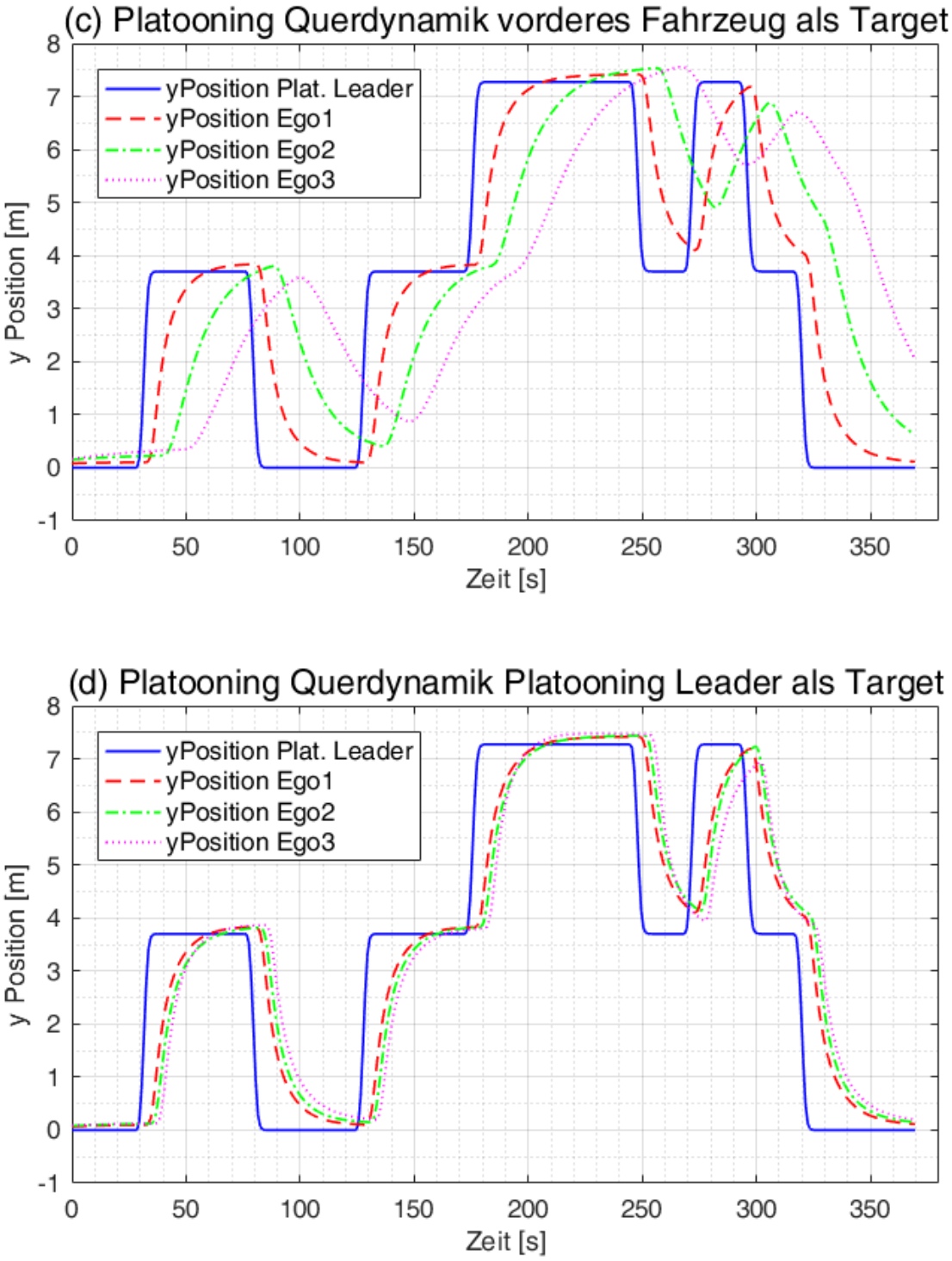

Abbildung 3.17.: Vergleich der Ergebnisse Platooning Querdynamik: (c) vorderes Fahrzeug ist Target, (d) Platooning Leader ist Target

3.5. ENABLE-S3 UC4 - Szenario

Die ENABLE-S3 UC4 Szenarien sind in [PHdC17] projektintern definiert. UC4 (Anwendungsfall 4, engl. Use Case 4) steht dabei für den Anwendungsfall Stauassistent. Diese Szenarien entsprechen den in Kapitel 1 beschriebenen Anforderungen an den TJA. Sie werden mit dem in Kapitel 2.3.2 beschriebenen TJA-Modell simuliert. Im Umfang dieser Masterarbeit werden die ersten 3 Szenarien dieses Szenarienkatalogs, so genau wie es das momentane Modell und die Regler erlauben, abgedeckt. Diese 3 Szenarien werden zu einem zusammenhängenden Szenario mit folgendem Ablauf zusammengefasst:

1. Drei Ego-Fahrzeuge fahren mit einer Geschwindigkeit von über 60 km/h auf dem rechten Fahrstreifen einer dreispurigen Autobahn mit geringem Verkehrsaufkommen. ACC und LKA sind in allen Ego-Fahrzeugen aktiviert. Die Geschwindigkeit variiert nur gering und es kommt selten zu Situationen, wo die Ego-Fahrzeuge bremsen müssen. Die Ego-Fahrzeuge sind mit einem TJA ausgestattet, welcher die Verkehrsumgebung (hier die Geschwindigkeit des Vorderfahrzeugs) überwacht. In diesem Szenario wird den Fahrern der Ego-Fahrzeuge nicht empfohlen, den TJA zu aktivieren.

2. Die Ego-Fahrzeuge müssen ihre Geschwindigkeit von 60 km/h auf 20 bis 60 km/h aufgrund mittleren Verkehrsaufkommens verringern. Die Geschwindigkeit variiert stark und es kommt zu Situationen, wo die Ego-Fahrzeuge bremsen müssen. Der TJA erkennt die neue Situation und schlägt den Fahrern die Aktivierung des TJA vor. Die Fahrer aktivieren daraufhin den TJA. Dies wird mit einem manuellen Schalter im Simulink-Element TJA-Controller (siehe Kapitel 2.2.6) während der Simulation umgesetzt. Die Ego-Fahrzeuge fahren weiterhin mit den ACC- und LKA-Reglern.

3. Die Ego-Fahrzeuge müssen ihre Geschwindigkeit aufgrund starken Verkehrsaufkommens auf unter 20 km/h verringern. Die Geschwindigkeit variiert stark. Der TJA erkennt die neue Situation und aktiviert automatisch die Platooningfunktion. Gleichzeitig werden die Systeme ACC und LKA deaktiviert. Wurde im zweiten Abschnitt des Szenarios der TJA nicht manuell aktiviert, so fahren die Fahrzeuge weiterhin mit den ACC- und LKA-Reglern.

Während der Platooningphase des Szenarios überholt der Platooning-Leader einen Verkehrsteilnehmer. Die Ego-Fahrzeuge folgen dem Platooning-Leader wie im Szenario aus Kapitel 3.4.2.

Dieses Szenario dient vor allem zur Bewertung des TJA-Controllers (Kapitel 2.2.6). Das heißt, ob er im richtigen Moment seine Aktivierung empfiehlt und wenn manuell aktiviert, selbstständig auf die Platooningfunktion umschaltet. Die Ergebnisse des oben beschriebenen Szenarios sind in Abbildung 3.18 dargestellt.

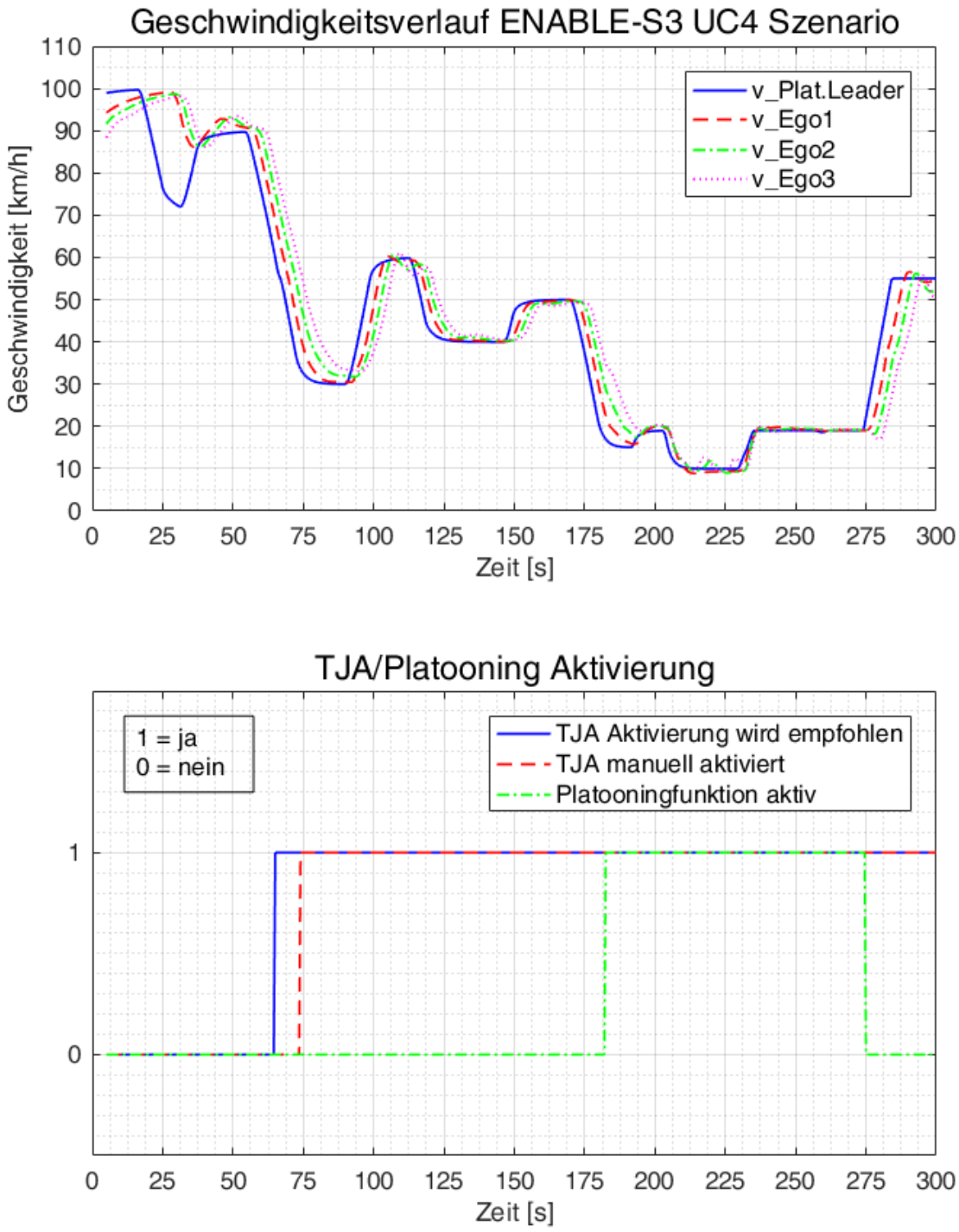

Abbildung 3.18.: Ergebnisse des ENABLE - S3 Szenarios mit TJA/Platooning Aktivierung

4. Diskussion

In diesem Kapitel werden die Ergebnisse aus Kapitel 3 Szenarien und Resultate diskutiert.

4.1. Vergleich parallel und sequenitelle Co-Simulation

Die Ergebnisse des Vergleichs zwischen paralleler und sequentieller Co-Simulation (siehe Abbildung 3.1) sind sehr ähnlich. Die maximale Differenz der Geschwindigkeit des Ego-Fahrzeugs zwischen paralleler und sequentieller Simulation ist 0,142 km/h, der Mittelwert ist 0,0002 km/h. Die Peaks zu Beginn der Simulation sind auf die größeren Extrapolationsfehler der numerischen Berechnung bei der parallelen Simulation zurückzuführen. Die mittlere Geschwindigkeit des Ego-Fahrzeugs während der gesamten Simulation beträgt 61,2 km/h. Die in Kapitel 1.3.1 beschriebenen größeren Extrapolationsfehler der parallelen Simulation sind also relativ klein. Zu Beginn der Simulation schwankt die Geschwindigkeitsdifferenz am größten, ab Sekunde 25 bleibt sie relativ konstant am Wert 0. Der Makrostep wurde mit 0,01 s also klein genug gewählt, sodass die Extrapolationsfehler der parallelen Simulation vernachlässigbar klein werden. Die benötigte Simulationszeit der parallelen Simulation ist um 3 Sekunden kürzer als die der sequentielle Co-Simulation. Dies ist relativ zur Gesamtzeit von 85 Sekunden bei der parallelen Co-Simulation nicht entscheidend. Später wird das verwendete ACC und LKA-Modell auf 3 Ego-Fahrzeuge erweitert, wodurch die Simulationszeit verlängert wird und der Unterschied zur sequentiellen Simulation deutlicher wird. Durch diese Erkenntnisse werden die folgenden Szenarien parallel simuliert. Es sollte für weitere Szenarien bedacht werden, dass die Fehler der numerischen Berechnung erst nach einer Zeit von (hier) 25 s geringer werden und Daten für Szenarien erst ab diesem Zeitpunkt gesammelt werden sollten. Im Anhang wird A.2 beschrieben, wie die Datenspeicherung der Simulation eingestellt werden kann.

4.2. Variation des Makrosteps

Um den Einfluss des Makrosteps auf die Simulation zu bestimmen, wurden Simulationen durchgeführt, in denen der Makrostep zwischen 0,01 s, 0,01 s und 0,001 s variiert. Bei einem Makrostep von 0,1 Sekunden wird das Szenario nicht korrekt durchgeführt, da die Vorgaben für Targetfahrzeuge in VTD nicht umgesetzt werden. Abbildung 3.2 zeigt,

dass die Unterschiede der Geschwindigkeit des Ego-Fahrzeugs im Szenairo zwischen einem Makrostep von 0,01 und 0,001 maximal 0,2 km/h betragen. Die Peaks im Verlauf der Geschwindigkeitsdifferenz beim Start der Simulation sind auf die genaueren Extrapolationswerte der Variante mit einem Makrostep von 0,001 s zurückzuführen. Ab Sekunde 60 sind weitere Peaks im Geschwindigkeitsverlauf, welche mit den Manöveränderungen im Szenario zusammenhängen. Es kommt zu größeren Änderungen der Sollbeschleunigung im Regler, wodurch die Extrapolationfehler steigen. Es ist anzunehmen, dass die Extrapolationsfehler bei der Simulation mit einem Makrostep von 0,001 s geringer sind, als die der Simulation mit einem Makrostep von 0,01 s. Dadurch entstehen bei Manöveränderungen Peaks im Verlauf der Geschwindigkeitsdifferenz. Da die Simulationszeiten der Variante mit dem Makrostep von 0,001 s 100 s länger dauert, wird in zukünftigen Szenarien ein Makrostep von 0,01 s gewählt.

4.3. Target Changes Lane - Test (TCL)

Die Ergebnisse des TCL Tests (Abbildung 3.4 und 3.6) zeigen, dass das Ego-Fahrzeug bei der Messfahrt früher verzögert als bei den beiden Simulationsvarianten. Das Ego-Fahrzeug bei der Simulation mit dem fahrstreifenabhängigen Sensormodell (LDACC) verzögert 0,6 s früher als ohne LDACC. Der Grund dafür ist, dass der ACC-Regler des Ego-Fahrzeugs mit LDACC das Target-Fahrzeug wechselt, sobald Target1 den Fahrstreifen wechselt. Durch das frühere Umschalten auf Target2 wird die benötigte Bremsbeschleunigung um 0,3 m/s^2 verringert. Der ACC-Regler des Ego-Fahrzeugs der Messfahrt reagiert noch schneller auf die neue Situation. Allerdings erreicht er auch wesentlich größere Beschleunigungen (-4,67 m/s^2) als die Ego-Fahrzeuge der Simulation (-3,45 m/s^2 mit LDACC bzw. -3,75 m/s^2 ohne LDACC). Das heißt der ACC-Regler der Simulation bremst komfortabler, mit dem Nachteil, dass der Abstand zwischen Ego-Fahrzeug und Target2 beim Test ohne LDACC kleiner wird als bei der Messfahrt.

4.4. Cut into Lane - Test (CIL)

Beim CIL-Test ergeben sich deutlich größere Unterschiede der Reaktionszeiten zwischen den Varianten ohne und mit LDACC. Abbildung 3.10 zeigt, dass das Ego-Fahrzeug in der Variante mit LDACC fast zum gleichen Zeitpunkt zu verzögern beginnt wie bei der Messfahrt. Das Ego-Fahrzeug in der Variante ohne LDACC bremst um 3,2 s früher. Dies ist damit zu erklären, dass das Ego-Fahrzeug ohne LDACC auf das neue Target (Target2-Fahrzeug) umschaltet, sobald dies näher am Ego-Fahrzeug ist, obwohl es sich noch nicht auf dem gleichen Fahrstreifen wie das Ego-Fahrzeug befindet. Damit ergeben sich um 0,45 m/s^2 geringere Verzögerungswerte als bei der Variante mit LDACC. Bei der Messfahrt werden wie beim TCL-Test wesentlich größere Beschleunigungswerte (-3,7 m/s^2) als bei der Simulation erreicht. Außerdem beschleunigt das Ego-Fahrzeug der Messfahrt nach der Verzögerung (ab Sekunde 10). Dies ist bei den Ego-Fahrzeugen

der Simulation nicht der Fall. Da auch der Minimalabstand zwischen Ego-Fahrzeug und Target2 bei der Simulationsvariante ohne LDACC größer und bei der Simulationsvariante mit LDACC annähernd gleich ist wie bei der Messfahrt (siehe Abbildung 3.8), ist der ACC-Regler der Simulation komfortabler und ausgelegt als jener des Ego-Fahrzeugs bei der Messfahrt.

Damit hat das fahrstreifenabhängige Sensormodell beim TCL-Test einen positiven und beim CIL-Test einen negativen Einfluss auf die Sicherheits- und Komfortwerte des ACC-Reglers, da sich größere Beschleunigungswerte negativ auf den Komfort auswirken. Die Unterschiede der ACC-Regler zwischen Messfahrt und Simulation erklären sich durch verschiedene Regelstrategien zur Berechnung der Sollbeschleunigung. Um die ACC-Regler der Simulation genau auf die des Messfahrzeugs abzustimmen, müsste eine Parameteridentifikation des ACC-Reglers des Benchmarkfahrzeugs der Messfahrt durchgeführt werden. Dies war nicht Teil der gestellten Aufgabe, daher wurden die Parameter des ACC-Reglers der Simulation bei den Originalwerten belassen. Es wird beim Vergleich von Simulation und Messfahrt jedoch klar, dass das ACC System des Ego-Fahrzeugs bei der Messfahrt bei den Tests ähnlich reagiert wie die Variante mit LDACC. Daher wird dieses System weiterverwendet und dient als Grundlage für den TJA.

4.5. Platooning Längsdynamik

Die in Kapitel 3.4.1 simulierten Ergebnisse zeigen die Auswirkungen der Platooningfunktion in Bezug auf die Längsdynamik der Fahrzeuge. In Abbildung 3.13 wird bereits deutlich, dass sich die Fahrzeuge bei aktiver Platooningfunktion schneller an die Geschwindigkeit des Platooning-Leaders anpassen als ohne Platooningfunktion. Der geringe zeitliche Verzug der Ego-Fahrzeug zum Platooning-Leader in (b) ist auf das unterschiedliche Fahrzeugmodell von VSM und VTD zurückzuführen. Das Fahrermodell des Platooning-Leaders wird von VTD berechnet und verwendet nicht das gleiche VSM-Fahrzeugmodell wie die Platooningfahrzeuge. Dem Platooning-Leader werden im Szenario die Zielgeschwindigkeit und Beschleunigung vorgegeben, welche bei der Simulation umgesetzt werden und mit den Reglern der Ego-Fahrzeuge nicht exakt reproduzierbar sind. Die mit VSM berechneten Ego-Fahrzeuge weisen daher einen unterschiedlichen Beschleunigungsverlauf auf. Sie reagieren mit einer, auf die Regler und das unterschiedliche Fahrzeugmodell zurückzuführenden zeitlichen Verzögerung auf die Geschwindigkeitsänderungen des Platooning Leaders.

Abbildung 3.14 zeigt das in der Einleitung (Kapitel 1.2.4) beschriebene Potential der Platooningfunktion. Es wird der Abstand vom Platooning-Leader zum Ego3-Fahrzeug ohne und mit aktiver Platooningfunktion gegenübergestellt. Während sich der Abstand ohne Platooningfunktion während des gesamten Szenarios ohne Platooningfunktion zwischen 32 und 58 m bewegt, wird dieser Abstand bei aktiver Platooningfunktion auf 26 bis 32 m minimiert. Dieser Abstand ist die Gesamtlänge des Platoons und soll möglichst konstant gehalten werden, um den Verkehrsfluss so hoch wie möglich zu halten.

Die Abstände zwischen den einzelnen Fahrzeugen des Platoons sind in Abbildung 4.1 dargestellt. Es ist zu erkennen, dass die Fahrzeuge weiter hinten im Platoon einen konstanteren Abstand zum Vorderfahrzeug aufweisen. Dieser ist wiederum auf das unterschiedliche Fahrzeug- und Fahrermodell des von VTD berechneten Platooning-Leaders zurückzuführen, welchem das Ego1-Fahrzeug mit seinen Reglern nicht exakt folgen kann. Die Verbesserung der Performance des Platooning-Reglers, je weiter hinten im Platoon sich ein Fahrzeug befindet bedeutet auch, dass das Modell für weitere Fahrzeuge erweitert werden kann, ohne dass sich die Performance des gesamten Platoons verschlechtert.

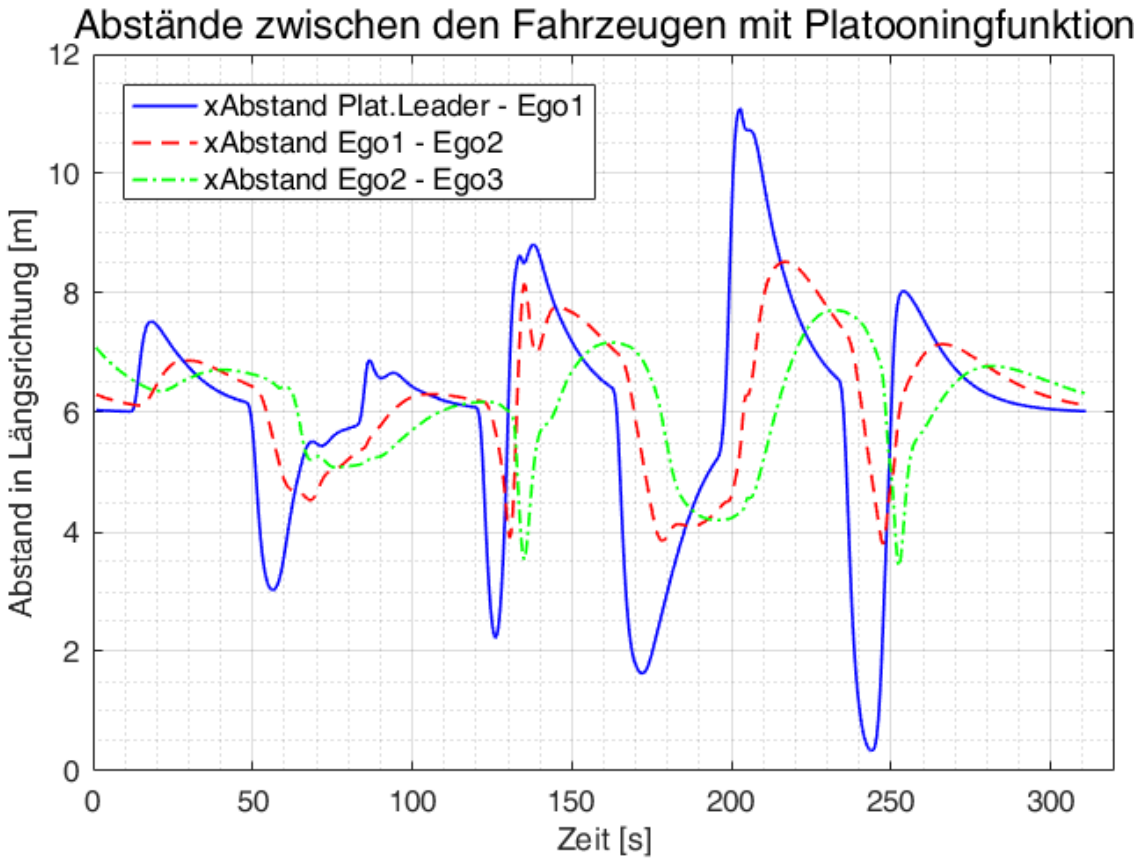

Abbildung 4.1.: Abstände zwischen den Fahrzeugen des Platoons mit dem Szenario aus Kapitel 3.4.1

Wie in Kapitel 3.4.1 beschrieben, wird die Geschwindigkeit des Platooning-Leaders zur Regelung des Abstands bei der Platooningfunktion verwendet, solange ein einstellbarer Abstandskorridor zum direkten Vorderfahrzeug eingehalten wird. Bei den letzten 3 Verzögerungen des Platooning-Leaders verlässt das Ego1-Fahrzeug den eingestellten Korridor von 4 bis 18 m und regelt den Abstand wieder mit den Geschwindigkeitsdaten des Sensors. Dieser Korridor begründet sich auf weitere Szenarien in [PHdC17], wo fehlerhafte

V2V-Kommunikation getestet werden soll. Verlässt ein Fahrzeug diesen Korridor, wird wieder auf das ACC-System umgeschaltet und einer möglichen Kollision von Fahrzeugen im Platoon wird so vorgebeugt.

4.6. Platooning Querdynamik

Die Ergebnisse der Platooningfunktion in Querrichtung sind in Abbildung 3.17 aufbereitet. Wie in Kapitel 3.4.2 beschrieben, unterscheiden sich die Varianten (c) und (d) durch die unterschiedlichen Target-Fahrzeuge. Während in Variante (c) der y-Postition des jeweiligen Vorderfahrzeug gefolgt wird, ist in Variante (d) der Platoonin-Leader das Target-Fahrezug. In Variante (d) folgen die Ego-Fahrzeuge dem Platooning Leader wesentlich exakter als in Variante (c). Dies liegt daran, dass sich der Regelfehler in Variante (c) aufintegriert. Richten sich hingegen alle Fahrzeug nach dem Platooning-Leader bleibt der Fehler für alle Fahrzeuge im Platoon gleich und gering im Vergleich zu Variante (c). In beiden Varianten reagieren die Fahrzeuge im Platoon später als gewünscht auf den Fahrstreifenwechsel des Platooning-Leaders. Dies ist auf die Charakteristik des PID-Reglers zurückzuführen. Daher folgen die Fahrzeuge im Platoon nicht exakt der vorgegebenen Spur des Platooning-Leaders. In der Variante (c) ist dieser Effekt noch stärker ausgeprägt. In weiteren Stufen der Querdynamikregelung der Platooningfunktion sollten daher andere Reglermodelle berücksichtigt werden.

4.7. ENABLE-S3 UC4 - Szenario

Dieses Szenario dient der Gesamtbewertung des TJA-Modells mit dem TJA-Controller. In den Ergebnissen des Szenarios aus Abbildung 3.18 erkennt man, dass die Aktivierung des TJA empfohlen wird sobald die Geschwindigkeit des Fahrzeugs vor den 3 Ego-Fahrzeugen (welches im späteren Verlaufs des Szenarios der Platooning-Leader sein wird) unter 60 km/h fällt. Dies ist nach 65 Sekunden der Fall. 8 Sekunden später darauf wurde der TJA der 3 Ego-Fahrzeuge manuell aktiviert. Nach 182 Sekunden fällt die Geschwindigkeit des Vorderfahrzeugs unter 20 km/h und die Platooningfunktion der 3 Ego-Fahrzeuge wird automatisch aktiv. Das Vorderfahrzeug wird zum Platooning-Leader. Steigt die Geschwindigkeit wieder über 20 km/h wird die Platooningfunktion deaktiviert und die Längsdynamikregelung wird wieder vom ACC-System übernommen. Der TJA-Controller reagiert also richtig und entsprechend den ENABLE-S3 UC4 Vorgaben.

In diesem Szenario werden alle Regler der vorhergehenden Szenarien verwendet. Die manuelle Aktivierung des TJA über das Simulink-Element TJA-Controller ist während der Simulation möglich. Damit ist die Funktionsfähigkeit des Modells und der Simulationsumgebung mit Model.CONNECT, VTD, VSM und Simulink nachgewiesen und weitere Ausbaustufen des Modells können folgen.

5. Zusammenfassung

In diesem Kapitel werden die wichtigsten Punkte aller Kapitel der Masterarbeit zusammengefasst.

Kapitel 1 Einleitung: In dieser Masterarbeit wird die Modellbildung und Simulation eines kooperativen Stauassistenten beschrieben. Dies ist Teil des ENABLE-S3 Projekts dessen Ziel es ist, physische Validierungs- und Verifizierungsaufwände durch virtuelles Testen und Verifizieren zu unterstützen und dadurch Kosten zu senken. Ein Stauassistent (engl. TJA, Traffic Jam Assistant) ist ein Fahrerassistenzsystem, welche den Fahrer beim Lenken eines Fahrzeugs unterstützen und einzelne Tätigkeiten die zum Betrieb eines Fahrzeugs nötig sind übernehmen. Dadurch soll die Sicherheit und der Komfort des Fahrers und der Passagiere erhöht werden. Ein Stauassistent unterstützt den Fahrer in Verkehrssituationen mit geringen Geschwindigkeiten bzw. bei „Stop-and-Go“ Verkehr. Diese Funktion unterstützt den Fahrer während dem als besonders unangenehm empfundenen Fahrens in Stausituationen. Stauassistenten mit Platooningfunktion und Fahrzeug-zu-Fahrzeug (engl. V2V, Vehicle-to-Vehicle) Kommunikation haben außerdem das Potential, die Verkehrsflussdichte zu erhöhen bzw. die Straßeninfrastruktur besser auszunutzen. Der Sicherheitsabstand zwischen den Fahrzeugen kann reduziert werden, da die Reaktionszeit des Fahrers oder eines Sensors reduziert wird. Die Platooningfunktion wird bei Stauassistenten üblicherweise bei geringen Geschwindigkeiten aktiviert. Durch die V2V-Kommunikation sind die Informationen über Positionen, Geschwindigkeiten und Beschleunigungen aller Fahrzeuge im Platoon für jedes Fahrzeug im Platoon jederzeit abrufbar. Dadurch können alle Fahrzeug im Platoon bei geringem Abstand einheitlicher beschleunigen. Bei höheren Geschwindigkeiten greift der Stauassistent auf die Systeme ACC und LKA zurück. In der Einleitung wurde außerdem der Unterschied zwischen paralleler und sequentieller Simulation beschrieben. Die parallele Simulation ist zwar schneller als die sequentielle Simulation, weist allerdings größere Extrapolationsfehler auf. Die Extrapolationsfehler können minimiert werden, indem der Simulationsschritt klein genug gehalten wird.

Kapitel 2 Methoden: Die Simulation dieses Stauassistenten erfolgte über die Co-Simulationssoftware Model.CONNECT. Mit dieser Co-Simulationsplattform ist es möglich, verschiedene Simulationsprogramme miteinander zu koppeln und so die Vorteile der Simulationsprogramme zu bündeln. Das in dieser Weise aufgebaut Modell des Stauassistenten besteht aus der Verkehrssimulation mit VTD, den Fahrzeugmodellen in VSM, einer Koordinatentransformation zwischen VTD und VSM und den Reglern, programmiert in MATLAB/Simulink. Die verwendeten Simulink Modelle sind ACC, LKA, TJA, TJA-Controller und dem Sensormodell. Die Visualisierung der Simulation bzw. der vor-

her erstellten Szenarien mit statischen (z.B. Straße) und dynamischen (z.B. Verkehr) Elementen erfolgt in VTD. In diesem Programm können ideale Sensoren definiert werden, welche Verkehrsdaten aus der Simulation auslesen und zu Sensordaten bündeln. Die Verkehrsdaten werden über Model.CONNECT und dem Sensormodell den Reglern des Stauassistenten übergeben. Diese berechnen die Gas- bzw. Bremspedalstellungen sowie den Lenkradwinkel. Die Ausgänge der Regler sind die Eingänge des Fahrzeugmodells welches mit VSM berechnet wird. Weitere Eingänge in VSM sind Straßeninformationen wie etwa der Reibwert, welche direkt von VTD geliefert werden. VSM berechnet die Reaktion des Fahrzeugs auf die Eingänge und übergibt die berechneten Daten wie Fahrzeugposition, -geschwindigkeit und -beschleunigung wieder an die Regler und über die Koordinatentransformation an VTD. Dieser geschlossene Co-Simulationskreislauf kann parallel oder sequentiell simuliert werden. Die parallele Simulation ist etwas schneller und wurde daher für die Co-Simulation des Stauassistenten verwendet. Es wurden zwei Modelle in Model.CONNECT aufgebaut. Ein Modell mit ACC- und LKA-Reglern, dem Sensormodell und einem Ego-Fahrzeug. Dieses wurde zu einem zweiten Modell mit 3 Ego-Fahrzeugen, dem TJA-Modell, erweitert. Beim TJA-Modell wurden die ACC- und LKA-Reglern um die Platooningfunktionen erweitert. Außerdem wurde das Simulink-Element TJA-Controller hinzugefügt, welches zwischen den verschiedenen Modi des TJA wechselt. Das erste Modell diente dem Vergleich des ACC-Reglers mit realen Messwerten. Mit dem TJA-Modell wurde ein Szenario simuliert, welche die Platooningfunktion testeten. Abschließend wurde das Verhalten des TJA-Controllers in einem Szenario, welches auf dem ENABLE-S3 Testszenario basiert, untersucht.

Kapitel 3 Szenarien und Resultate: In diesem Kapitel wurden die erstellten Modelle mit verschiedenen Szenarien simuliert. Die erste Simulation diente dem Vergleich zwischen paralleler und sequentieller Simulation. Die Ergebnisse zeigten einen sehr geringen Unterschied, daher wurde für die folgenden Szenarien die etwas schnellere parallele Simulationsmethode verwendet. In einem Szenario wurden die ACC- und LKA-Regler mit einem Modell mit einem Ego-Fahrzeug verifiziert. Dazu wurden die Ergebnisse zweier Manöver einer Messfahrt mit den Simulationsergebnissen eines den realen Manövern nachgebauten Szenarios verglichen. Der ACC-Regler mit fahrstreifenabhängigem Sensormodell reagierte ähnlich wie in den Messfahrten. Später wurde das Modell auf 3 Ego-Fahrzeuge erweitert und mit dem TJA-Regler simuliert. Der TJA kann ab einer Geschwindigkeit unter 60 km/h manuell aktiviert werden. Fällt die Geschwindigkeit weiter unter 20 km/h wird bei aktivem TJA automatisch die Platooningfunktion aktiviert. Die Längs- bzw. Querdynamik der Platooningfunktion wurde in 2 Szenarien bewertet. Mit der Platooningfunktion konnten geringere Abstände zwischen den Fahrzeugen realisiert werden als mit dem ACC-Regler, da die Fahrzeuge durch die V2V-Kommunikation schneller auf Geschwindigkeitsänderungen des Platooning-Leaders reagierten. Im Szenario für die Bewertung der Querdynamik führte der Platooning-Leader mehrere Fahrstreifenwechsel durch. Die Ego-Fahrzeuge im Platoong folgten dem Platooning-Leader in definierten Zeitabständen. Die Bedingungen des letzten Szenarios in diesem Kapitel wurden so genau wie möglich nach dem ENABLE-S3 UC4 Szenarienkatalog gerichtet. Die geschwindigkeitsabhängigen Zustände des TJA wurden durchfahren und die Ergeb-

nisse zeigten, dass der TJA seine manuelle Aktivierung zur richtigen Zeit vorschlug und wenn manuell aktiviert, automatisch in die Platooningfunktion umschaltete.

Kapitel 4 Diskussion: Die Ergebnisse aus Kapitel 3 wurden hier diskutiert. Mit dem ACC und LKA Modell wurde entschieden, das ACC-System mit fahrstreifenabhängigem Sensormodell für das TJA Modell zu verwenden, da die Ergebnisse näher an den Werten der Messfahrt lagen. Die ersten einfachen Platooning-Regler zeigten, dass es möglich ist, mit der vorhandenen Simulationsumgebung eine detailliertere Platooningfunktion zu implementieren. Durch die Simulationen mit dem TJA-Regler wurde die Funktionsfähigkeit des TJA-Modells nachgewiesen, und es kann für weitere Ausbauschritte des Stauassistenten verwendet werden.

5.1. Fazit

Die Aufgabenstellung, nämlich eine Simulationsumgebung für das virtuelle Testen eines Stauassistentenmit Platooningfunktion aufzubauen, wurde erreicht. Validierungsfahrten mit Benchmarkfahrzeugen, um die eingesetzten Regler im Modell zu testen, zeigten ähnliches Verhalten wie die Simulation mit dem ACC- und LKA-Modell. Erste Platooning-Regler zeigten den positiven Effekt der Platooningfunktion und können mit dem bestehenden Modell weiterentwickelt werden.

5.2. Ausblick

Um die genaue Szenariodefinition im ENABLE-S3 Szenarienkatalogs erfüllen zu können, sind noch weitere Regler nötig. Die weiteren Szenarien behandeln das Eindringen eines Verkehrsteilnehmers in den Platoon, eine fehlerhafte V2V-Kommunikation, das Anhängen eines weiteren Fahrzeugs an den Platoon sowie den Austritt eines Fahrzeugs aus dem Platoon. Ein weiterer möglicher Ausbauschritt des Modells bzw. der Regler ist die Verbesserung der Platooningfunktion. Momentan folgen die Fahrzeuge im Platoon dem Platooning-Leader bei einem Fahrstreifenwechsel ohne sich vorher zu versichern, ob ein sicheres Folgen des Platooning-Leaders überhaupt möglich ist. Die idealen Sensormodelle in VTD sollen in Zukunft durch realistischere Sensormodelle ersetzt werden. Ein Schwachpunkt der Simulation ist die Dauer, bis die Simulation startet. Das Laden der einzelnen Simulationselemente bei einer Simulation mit 3 Ego-Fahrzeugen nimmt 1,5 min in Anspruch. Abhilfe könnte die Umwandlung der Simulink-Elemente in FMUs schaffen, da diese nach Rücksprache mit AVL eine kürzere Ladezeit aufweisen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[22109] International Standard ISO 22178. Intelligent transport systems — low speed following (lsf) systems — performance requirements and test procedures, 2009.

[ABT+15] A. Alam, B. Besselink, V. Turri, J. Martensson, and K. H. Johansson. Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency. *IEEE Control Systems*, 35(6):34–56, 2015. Cited By :13.

[Aut15] IPG Automotive. Carmaker programmer´s guide version 5.0.2, 2015.

[AVL17a] AVL. VSM Vehicle Dynamics Simulation. Available at `https://www.avl.com/documents/10138/2095827/Product+brochure+VSM+4`, 2017. Accessed on 10.10.2017.

[AVL17b] AVL List GmbH. Model.CONNECT Integration of virtual and real components. Available at `https://www.avl.com/-/model-connect-`, 2017. Accessed on 13.10.2017.

[Axe17] J. Axelsson. Safety in vehicle platooning: A systematic literature review. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1033–1045, 2017. ID: 1.

[Ber15] Stefan Bernsteiner. *Integration of Advanced Driver Assistance Systems on Full-Vehicle Level*. PhD thesis, 2015.

[BSC+12] Carl Bergenhem, Steven Shladover, Erik Coelingh, Christoffer Englund, and Sadayuki Tsugawa. Overview of platooning systems. In *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*, 2012. ID: 174621.

[Bun17a] Bundeskanzleramt. Bauart und Ausrüstung der Kraftfahrzeuge und Anhänger. Available at `https://www.ris.bka.gv.at/Dokument.wxe?Abfrage=Bundesnormen&Dokumentnummer=NOR40089365`, 2017. Accessed on 10.10.2017.

[Bun17b] Bundeswesen für Straßenwesen. Rechtsfolgen zunehmender Fahrzeugautomatisierung. Available at `http://www.bast.de/DE/Publikationen/Foko/Downloads/2012-11.pdf?__blob=publicationFile&v=1`, 2017. Accessed on 21.08.2017.

[Dai17] Daimler. Vernetzte LKW. Available at `https://www.daimler.com/innovation/digitalisierung/vernetzung/vernetzte-lkw.html`, 2017. Accessed on 08.08.2017.

[Des16] Destatis, editor. *Unfallentwicklung auf deutschen Strassen 2015*. Statistisches Bundesamt, 2016.

[Doc17] ADVANCED SIMULATION TECHNOLOGIES Software Documentation. Model.connecttm user manual, 2017.

[Dok17] AVL VSM Dokumentation, editor. *AVL VSM 4 User's Guide*. AVL List GmbH, Graz, version 4.1 service release 1 edition, 2017.

[Dup15] Marius Dupuis, editor. *VIRES Virtual Test Drive - User Manual*. VIRES, issue: n edition, 2015.

[EC08] United Nations Economic and Social Council. European agrement on main international traffic arteries (agr), 2008.

[Eck16] Jacqueline Eckhardt, editor. *Lessons Learnt - European Truck Platooning Challenge 2016*. Dutch Ministry of Infrastructure and the Environemnt, Amsterdam, 2016.

[Ena17] Enable S3. About the Project. Available at `http://www.enable-s3.eu`, 2017. Accessed on 07.08.2017.

[FMI17] FMI-Standard. Functional Mock Up Interface. Available at `file:///C:/Users/rkastner/Downloads/The_Functional_Mockup_Interface.pdf`, 2017. Accessed on 10.10.2017.

[Gmb17] AVL List GmbH, editor. *Schulungsunterlagen: Coupling of Elements*. AVL List GmbH, Graz, 2017.

[HES11] Bernd Heißing, Metin Ersoy, and SpringerLink. *Chassis handbook*. Vieweg + Teubner, Wiesbaden, 2011. [electronic resource] : fundamentals, driving dynamics, components, mechatronics, perspectives / Bernd Heißing, Metin Ersoy (ed.).; ill; Electronic reproduction. New York : Springer, 2010. Mode of access: World Wide Web. System requirements: Web browser. Title from title screen (viewed on Dec. 9, 2010). Access may be restricted to users at subscribing institutions.; Includes bibliographical references.

[Inv17] Invent. Projektbericht INVENT ërfahren mobil mit 8 Sinnen. Available at `http://www.tuvpt.de/fileadmin/pdf/INVENT-Ergebnisse.pdf`, 2017. Accessed on 10.10.2017.

[IPG17] IPG Automotive. CarMaker: Virtual testing of automobiles and light-duty vehicles. Available at `https://ipg-automotive.com/products-services/simulation-software/carmaker/`, 2017. Accessed on 10.10.2017.

[Lau12] Claude Laurgeau. *Intelligent Vehicle Potential and Benefits*, pages 1537–1551. Handbook of Intelligent Vehicles. Springer London, London, 2012.

[Mat17] Mathworks. Simulink. Available at `https://de.mathworks.com/products/simulink.html`, 2017. Accessed on 10.10.2017.

[MWK17] Santa Maiti, Stephan Winter, and Lars Kulik. A conceptualization of vehicle platoons and platoon operations, July 2017 2017. ID: 271729.

[Nes13] Georg Nestlinger, editor. *Modellbildung und Simulaiton eines Spurhalte-Assistenzsystems*. TU Graz, Graz, masterarbeit edition, 2013.

[Nov16] Dominik Novak, editor. *Auswertungskatalog Stauassistenzsystem - Messfahrten 8.7. bis 20.7.2015*. Bachelorarbeit, Institut für Fahrzeugtechnik, Technische Universität Graz, bachelorarbeit edition, 2016.

[PHdC17] David Pereira, Rolf Hettel, and Alexandre Guyon de Chemilly, editors. *D1.1.1 Specification of Automotive Use Cases*. ENABLE-S3 UC4, 2017.

[Rei10] Konrad Reif. *Geschichte der Fahrerassistenzsysteme*, page 209. Fahrstabilisierungssysteme und Fahrerassistenzsysteme. Springer, 2010.

[SAKE16] Sajjad Samiee, Shahram Azadi, Reza Kazemi, and Arno Eichberger. Towards a decision-making algorithm for automatic lane change manouvre considering traffic dynamics. *PROMET Traffic and Transportation*, 28:91–103, 2016.

[Sta17] Stanford Law School. SAE Levels of Driving Automation. Available at `http://cyberlaw.stanford.edu/blog/2013/12/sae-levels-driving-automation`, 2017. Accessed on 10.10.2017.

[Ver17] Verband der Automobilindustrie. Lane Keeping Assist Systems. Available at `https://www.vda.de/de/themen/sicherheit-und-standards/spurhalteassistenzsysteme/lane-keeping-assist-systems-lkas.html`, 2017. Accessed on 08.08.2017.

[VIR17a] VIRES. VTD Virtual Test Drive. Available at `https://vires.com/vtd-vires-virtual-test-drive/`, 2017. Accessed on 10.10.2017.

[VIR17b] VIRES Simulationstechnologie GmbH. Connected to the open World. Available at `https://vires.com/open-solutions/#openscenario`, 2017. Accessed on 11.08.2017.

[VIR17c] VIRES Simulationstechnologie GmbH. OpenScenario. Available at `http://www.openscenario.org/project.html`, 2017. Accessed on 11.08.2017.

[Vir17d] Virtual Vehicle. Co-Simulation. Available at `http://www.v2c2.at/foerderprogramme/k2-forschungsprogramm/ee-software/co-simulation/`, 2017. Accessed on 09.08.2017.

[WHLS15] H. Winner, S. Hakuli, F. Lotz, and C. Singer, editors. *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort (3., überarbeitete und ergänzte Auflage Aufl.)*. Wiesbaden: Springer Vieweg, 2015.

A. Anhang

A.1. Verbindungskonfiguration des TJA-Modells

Hier werden alle Verbindungskonfigurationen des TJA-Modells, bis auf die der Koordinatentransformation, aus Kapitel 2.3.2 aufgelistet. Diese Liste kann aus Model.CONNECT exportiert und importiert werden. Der Text vor „˜“ ist die Definition des Ausgangsports eines Elements. Die Definition eines Ports ist \Elementname, Portbundle, Portname bzw. wenn kein Bundle verwendet wird \Elementname, , Portname. Nach der „˜“ folgt die Definition des Eingangsports.

\Ego1_VSM, , Vehicle.Acceleration Longitudinal, , ˜, \Ego1_TJA, , Sim_ConProVar_Ego1_acc_x,

\Ego1_TJA, , Sim_Ego1_Brake, , ˜, \Ego1_VSM, , Driver.Brake Pedal,

\Ego1_TJA, , Sim_Ego1_Gas, , ˜, \Ego1_VSM, , Driver.Accelerator Pedal,

\Ego1_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego1_LKA, , Ego1_Velocity,

\Ego1_LKA, , SteeringAngle, , ˜, \Ego1_VSM, , Driver.Steer Angle,

\VTD, , Ego1.roadPos.laneOffset, , ˜, \Ego1_LKA, , LaneOffset,

\Ego3_VSM, Vehicle_out, , , ˜, \VTD, Ego3.Vehicle, ,

\Ego3_VSM, Vehicle_out, Vehicle.Steer Angle Wheel.FL, , ˜, \VTD, Ego3.Vehicle, Ego3.fl.wheel.base.steeringAngle,

\Ego3_VSM, Vehicle_out, Vehicle.Steer Angle Wheel.FR, , ˜, \VTD, Ego3.Vehicle, Ego3.fr.wheel.base.steeringAngle,

\Ego3_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.FL, , ˜, \VTD, Ego3.Vehicle, Ego3.fl.wheel.base.springCompression,

\Ego3_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.FR, , ˜, \VTD, Ego3.Vehicle, Ego3.fr.wheel.base.springCompression,

\Ego3_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.RL, , ˜, \VTD, Ego3.Vehicle, Ego3.rl.wheel.base.springCompression,

\Ego3_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.RR, , ˜, \VTD, Ego3.Vehicle, Ego3.rr.wheel.base.springCompression,

\Sensordatenauswahl, , Ego1_Delta x, , ˜, \Ego1_TJA, , Sim_Delta_Pos_x,

\Ego3_VSM, , Vehicle.Acceleration Longitudinal, , ˜, \Ego3_TJA, , Sim_ConProVar_Ego1_acc_x,

\Sensordatenauswahl, , Ego3_Delta x, , ˜, \Ego3_TJA, , Sim_Delta_Pos_x,

\Ego3_TJA, , Sim_Ego1_Brake, , ˜, \Ego3_VSM, , Driver.Brake Pedal,

\Ego3_TJA, , Sim_Ego1_Gas, , ˜, \Ego3_VSM, , Driver.Accelerator Pedal,

\VTD, , Ego3.roadPos.laneOffset, , ˜, \Ego3_LKA, , LaneOffset,

\Ego3_LKA, , SteeringAngle, , ˜, \Ego3_VSM, , Driver.Steer Angle,

\Ego1_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego1_TJA, , Sim_UserInp_Ego1_vel_x,

\Ego1_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego1_TJA, , Sim_DesiredV_Ego1_vel_x,

\Ego1_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego1_TJA, , Sim_DesiredD_Ego1_vel_x,

\Ego3_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego3_LKA, , Ego1_Velocity,

\Ego3_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego3_TJA, , Sim_UserInp_Ego1_vel_x,

\Ego3_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego3_TJA, , Sim_DesiredV_Ego1_vel_x,

\Ego3_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego3_TJA, , Sim_DesiredD_Ego1_vel_x,

\Ego2_TJA, , Sim_Ego1_Gas, , ˜, \Ego2_VSM, , Driver.Accelerator Pedal,

\Ego2_TJA, , Sim_Ego1_Brake, , ˜, \Ego2_VSM, , Driver.Brake Pedal,

\Ego2_LKA, , SteeringAngle, , ˜, \Ego2_VSM, , Driver.Steer Angle,

\Ego2_VSM, , Vehicle.Acceleration Longitudinal, , ˜, \Ego2_TJA, , Sim_ConProVar_Ego1_acc_x,

\Ego2_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego2_LKA, , Ego1_Velocity,

\Ego2_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego2_TJA, , Sim_UserInp_Ego1_vel_x,

\Ego2_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego2_TJA, , Sim_DesiredV_Ego1_vel_x,

\Ego2_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego2_TJA, , Sim_DesiredD_Ego1_vel_x,

\VTD, , Ego1.MyRadarSensor.objectState.ext.speed.x, , ˜, \Sensordatenauswahl, , Ego1_Velocity x Vector,

\VTD, , Ego1.MyRadarSensor.objectState.base.pos.x, , ˜, \Sensordatenauswahl, , Ego1_Delta Position x Vector,

\VTD, , Ego3.MyRadarSensor_2.objectState.ext.speed.x, , ˜, \Sensordatenauswahl, , Ego3_Velocity x Vector,

\VTD, , Ego3.MyRadarSensor_2.objectState.base.pos.x, , ˜, \Sensordatenauswahl, , Ego3_Delta Position x Vector,

\VTD, , Ego2.MyRadarSensor_1.objectState.base.pos.x, , ˜, \Sensordatenauswahl, , Ego2_Delta Position x Vector,

\VTD, , Ego2.MyRadarSensor_1.objectState.ext.speed.x, , ˜, \Sensordatenauswahl, , Ego2_Velocity x Vector,

\Ego2_VSM, Vehicle_out, , , ˜, \VTD, Ego2.Vehicle, ,

\Ego2_VSM, Vehicle_out, Vehicle.Steer Angle Wheel.FL, , ˜, \VTD, Ego2.Vehicle, Ego2.fl.wheel.base.steeringAngle,

\Ego2_VSM, Vehicle_out, Vehicle.Steer Angle Wheel.FR, , ˜, \VTD, Ego2.Vehicle, Ego2.fr.wheel.base.steeringAngle,

\Ego2_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.FL, , ˜, \VTD, Ego2.Vehicle, Ego2.fl.wheel.base.springCompression,

\Ego2_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.FR, , ˜, \VTD, Ego2.Vehicle, Ego2.fr.wheel.base.springCompression,

\Ego2_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.RL, , ˜, \VTD, Ego2.Vehicle, Ego2.rl.wheel.base.springCompression,

\Ego2_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.RR, , ˜, \VTD, Ego2.Vehicle, Ego2.rr.wheel.base.springCompression,

\Ego2_VSM, Road_in, , , ˜, \VTD, Ego2.Road, ,

\VTD, Ego2.Road, Ego2.fl.contactPoint.roadDataIn.z, , ˜, \Ego2_VSM, Road_in, Road.Track Height.FL,

\VTD, Ego2.Road, Ego2.fr.contactPoint.roadDataIn.z, , ˜, \Ego2_VSM, Road_in, Road.Track Height.FR,

\VTD, Ego2.Road, Ego2.rl.contactPoint.roadDataIn.z, , ˜, \Ego2_VSM, Road_in, Road.Track Height.RL,

\VTD, Ego2.Road, Ego2.rr.contactPoint.roadDataIn.z, , ˜, \Ego2_VSM, Road_in, Road.Track Height.RR,

\Sensordatenauswahl, , Ego2_Delta x, , ˜, \Ego2_TJA, , Sim_Delta_Pos_x,

\VTD, , Ego2.roadPos.laneOffset, , ˜, \Ego2_LKA, , LaneOffset,

\Ego3_VSM, Road_in, , , ˜, \VTD, Ego3.Road, ,

\VTD, Ego3.Road, Ego3.fl.contactPoint.roadDataIn.z, , ˜, \Ego3_VSM, Road_in, Road.Track Height.FL,

\VTD, Ego3.Road, Ego3.fr.contactPoint.roadDataIn.z, , ˜, \Ego3_VSM, Road_in, Road.Track Height.FR,

\VTD, Ego3.Road, Ego3.rl.contactPoint.roadDataIn.z, , ˜, \Ego3_VSM, Road_in, Road.Track Height.RL,

\VTD, Ego3.Road, Ego3.rr.contactPoint.roadDataIn.z, , ˜, \Ego3_VSM, Road_in, Road.Track Height.RR,

\Sensordatenauswahl, , Ego1_Velocity x, , ˜, \Ego1_TJA, , Sim_Delta_Vel_x,

\Sensordatenauswahl, , Ego2_Velocity x, , ˜, \Ego2_TJA, , Sim_Delta_Vel_x,

\Sensordatenauswahl, , Ego3_Velocity x, , ˜, \Ego3_TJA, , Sim_Delta_Vel_x,

\Ego1_Desired_Speed, , Ego1_Desired_Speed_km_h, , ˜, \Ego1_TJA, , Desired_Speed_Ego1_m_s,

\Ego2_Desired_Speed, , Ego1_Desired_Speed_km_h, , ˜, \Ego2_TJA, , Desired_Speed_Ego1_m_s,

\Ego3_Desired_Speed, , Ego1_Desired_Speed_km_h, , ˜, \Ego3_TJA, , Desired_Speed_Ego1_m_s,

\VTD, , Ego1.MyRadarSensor.roadPos.laneId, , ˜, \Sensordatenauswahl, , Ego1_RadarLaneId,

\VTD, , Ego1.roadPos.laneId, , ˜, \Sensordatenauswahl, , Ego1_LaneId,

\VTD, , Ego3.roadPos.laneId, , ˜, \Sensordatenauswahl, , Ego3_LaneId,

\VTD, , Ego3.MyRadarSensor_2.roadPos.laneId, , ˜, \Sensordatenauswahl, , Ego3_RadarLaneId,

\VTD, , Ego2.MyRadarSensor_1.roadPos.laneId, , ˜, \Sensordatenauswahl, , Ego2_RadarLaneId,

\VTD, , Ego2.roadPos.laneId, , ˜, \Sensordatenauswahl, , Ego2_LaneId,

\VTD, Ego2.Road, Ego2.fl.contactPoint.friction, , ˜, \Ego2_VSM, Road_in, Road.Combined Grip.FL,

\VTD, Ego2.Road, Ego2.fr.contactPoint.friction, , ˜, \Ego2_VSM, Road_in, Road.Combined Grip.FR,

\VTD, Ego2.Road, Ego2.rl.contactPoint.friction, , ˜, \Ego2_VSM, Road_in, Road.Combined Grip.RL,

\VTD, Ego2.Road, Ego2.rr.contactPoint.friction, , ˜, \Ego2_VSM, Road_in, Road.Combined Grip.RR,

\VTD, Ego3.Road, Ego3.fl.contactPoint.friction, , ˜, \Ego3_VSM, Road_in, Road.Combined Grip.FL,

\VTD, Ego3.Road, Ego3.fr.contactPoint.friction, , ˜, \Ego3_VSM, Road_in, Road.Combined Grip.FR,

\VTD, Ego3.Road, Ego3.rl.contactPoint.friction, , ˜, \Ego3_VSM, Road_in, Road.Combined Grip.RL,

\VTD, Ego3.Road, Ego3.rr.contactPoint.friction, , ˜, \Ego3_VSM, Road_in, Road.Combined Grip.RR,

\VTD, Ego1.Road, Ego1.fl.contactPoint.friction, , ˜, \Ego1_VSM, Road_in, Road.Combined Grip.FL,

\VTD, Ego1.Road, , , ˜, \Ego1_VSM, Road_in, ,

\VTD, Ego1.Road, Ego1.fr.contactPoint.friction, , ˜, \Ego1_VSM, Road_in, Road.Combined Grip.FR,

\VTD, Ego1.Road, Ego1.rl.contactPoint.friction, , ˜, \Ego1_VSM, Road_in, Road.Combined Grip.RL,

\VTD, Ego1.Road, Ego1.rr.contactPoint.friction, , ˜, \Ego1_VSM, Road_in, Road.Combined Grip.RR,

\VTD, Ego1.Road, Ego1.fl.contactPoint.roadDataIn.z, , ˜, \Ego1_VSM, Road_in, Road.Track Height.FL,

\VTD, Ego1.Road, Ego1.fr.contactPoint.roadDataIn.z, , ˜, \Ego1_VSM, Road_in, Road.Track Height.FR,

\VTD, Ego1.Road, Ego1.rl.contactPoint.roadDataIn.z, , ˜, \Ego1_VSM, Road_in, Road.Track Height.RL,

\VTD, Ego1.Road, Ego1.rr.contactPoint.roadDataIn.z, , ˜, \Ego1_VSM, Road_in, Road.Track Height.RR,

\Ego1_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.FL, , ˜, \VTD, Ego1.Vehicle, Ego1.fl.wheel.base.springCompression,

\VTD, Ego1.Vehicle, , , ˜, \Ego1_VSM, Vehicle_out, ,

\Ego1_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.FR, , ˜, \VTD, Ego1.Vehicle, Ego1.fr.wheel.base.springCompression,

\Ego1_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.RL, , ˜, \VTD, Ego1.Vehicle, Ego1.rl.wheel.base.springCompression,

\Ego1_VSM, Vehicle_out, Vehicle.Vertical Wheel Travel.RR, , ˜, \VTD, Ego1.Vehicle, Ego1.rr.wheel.base.springCompression,

\Ego1_VSM, Vehicle_out, Vehicle.Steer Angle Wheel.FL, , ˜, \VTD, Ego1.Vehicle, Ego1.fl.wheel.base.steeringAngle,

\Ego1_VSM, Vehicle_out, Vehicle.Steer Angle Wheel.FR, , ˜, \VTD, Ego1.Vehicle, Ego1.fr.wheel.base.steeringAngle,

\VTD, New Player.Vehicle, New Player.objectState.base.pos.y, , ˜, \Ego1_LKA, , yTarget,

\VTD, New Player.Vehicle, New Player.objectState.base.pos.y, , ˜, \Ego3_LKA, , yTarget,

\VTD, New Player.Vehicle, New Player.objectState.base.pos.y, , ˜, \Ego2_LKA, , yTarget,

\TJA_Control_2, , Ego1_Activation, , ˜, \Ego1_LKA, , Ego1_Activation,

\TJA_Control_2, , Ego1_Activation, , ˜, \Sensordatenauswahl, , Ego1_Activation,

\TJA_Control_2, , Ego2_Activation, , ˜, \Sensordatenauswahl, , Ego2_Activation,

\TJA_Control_2, , Ego3_Activation, , ˜, \Sensordatenauswahl, , Ego3_Activation,

\Sensordatenauswahl, , Ego1_Delta x, , ˜, \Ego1_TJA, , TJA_Delta_Pos_x,

\TJA_Control_2, , Ego1_Activation, , ˜, \Ego1_TJA, , TJA_Activation,

\TJA_Control_2, , Ego1_Activation, , ˜, \Ego1_TJA, , TJA_Activation_1,

\TJA_Control_2, , Ego3_Activation, , ˜, \Ego3_TJA, , TJA_Activation,

\TJA_Control_2, , Ego3_Activation, , ˜, \Ego3_TJA, , TJA_Activation_1,

\Sensordatenauswahl, , Ego3_Delta x, , ˜, \Ego3_TJA, , TJA_Delta_Pos_x,

\TJA_Control_2, , Ego2_Activation, , ˜, \Ego2_TJA, , TJA_Activation,

\TJA_Control_2, , Ego2_Activation, , ˜, \Ego2_TJA, , TJA_Activation_1,

\Sensordatenauswahl, , Ego2_Delta x, , ˜, \Ego2_TJA, , TJA_Delta_Pos_x,

\Ego1_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego1_TJA, , v_Ego1,

\VTD, New Player.Vehicle, New Player.objectState.ext.speed.x, , ˜, \Ego1_TJA, , v_Target,

\Ego3_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego3_TJA, , v_Ego1,

\VTD, New Player.Vehicle, New Player.objectState.ext.speed.x, , ˜, \Ego3_TJA, , v_Target,

\Ego2_VSM, , Vehicle.Speed Longitudinal, , ˜, \Ego2_TJA, , v_Ego1,

\VTD, New Player.Vehicle, New Player.objectState.ext.speed.x, , ˜, \Ego2_TJA, , v_Target,

\TJA_Control_2, , Ego3_Activation, , ˜, \Ego3_LKA, , Ego1_Activation,

\TJA_Control_2, , Ego2_Activation, , ˜, \Ego2_LKA, , Ego1_Activation,

\Constant 1, , TimeDelay, , ˜, \Ego3_LKA, , TimeDelay,

\Constant 2, , TimeDelay, , ˜, \Ego1_LKA, , TimeDelay,

\Constant 3, , TimeDelay, , ˜, \Ego2_LKA, , TimeDelay,

\VTD, New Player.Vehicle, New Player.objectState.ext.speed.x, , ˜, \TJA_Control_2, , Traffic_Speed,

A.2. Dokumentation der Co-Simulation

Um den Einstieg in das aufgebaute Co-Simulationframework zukünftigen Benutzern zu erleichtern, wurde eine genaue Beschreibung des Vorgangs zur Erstellung der Simulationsmodelle in englischer Sprache erstellt. Das Dokument dient kann als Anleitung zum Aufbau der Modelle, sowie deren Erweiterung.

Robert Kastner

Documentation of the Co-Simulation of a Traffic Jam Assistant in Model.CONNECT

Documentation

Graz University of Technology

Faculty of Mechanical Engineering and Economic Sciences

Institute of Automotive Engineering

Member of Frank Stronach Institute

Director: Univ.-Doz. Dipl.-Ing. Dr. techn. Arno Eichberger

Graz, 12.10.2017

Abstract

This documentation deals with the installation and setup of the Co-Simulationframework. It should help to set up, understand and recreate the model of the master thesis “Modelling and Simulation of a Traffic Jam Assistant with a Co-Simulationframework”. First the installation of the needed Simulation programs is explained, followed by a short explanation of the programs concerning their usage in the Co-simulation. Then the setup of a new model in Model.CONNECT is described. Another chapter explains the current TJA model and its features. Finally the start of the simulation, the possible errors, possible error fixes and how to find data after the simulation has been run.

Contents

Abbreviations

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
FMU	Functional Mock-up Unit
GUI	Graphical User Interface
LDW	Lane Departure Warning
PID controller	Proportional–Integral–Derivative controller
LKA	Lane Keeping Assistant
TJA	Traffic Jam Assistant
TCP	Transmission Control Protocol
VSM	Vehicle Simulation
VTD	Virtual Test Drive

1. Introduction

This documentation should help recreating, understanding and setting up the model created during the master thesis “Modelling and Simulation of a Traffic Jam Assistant with a Co-Simulationframework. The goal is that the reader of this document is able to continue the work of the master thesis and refine the current model.

All the information in this documentation is based on the user manuals of the used simulation programs. They can be found at the server of the Institute of Automotive Engineering on the following location:

M:\FTG-P\3_RnD\DVS\Project_Funded\Current\2016_f001f_PA_Enable_S3\Work_01\14_UC4\05_Software_manuals

In addition to the extracted information of these user manuals personal experiences during working with the programs are documented.

For further questions to the TJA-model you can contact:
Robert Kastner
Pesendorstraße 39
4451 Garsten
E-mail: robertkastner2@gmx.at
Tel.Nr.: 0680/1282227

1.1 Co-Simulation Setup

The Co-Simulation is set up on two PCs. One Windows - PC where Model.CONNECT, VSM and MATLAB are installed and on LINUX – PC where VTD is installed. These computers are connected via a network. Their connection is explained in Chapter 4.2. Model.CONNECT is the program where the Co-simulation is conducted. All other simulation programs are implemented in Model.CONNECT.

2 Installation of the Programs

In this chapter the installation of the used simulation programs is described. In case the license files don´t work anymore Prof. Arno Eichberger can give you a contact to AVL who own the programs Model.CONNECT and VSM.

2.1 Installation of Model.CONNECT

The installation files for Model.CONNECT are located at

M:\FTG-P\3_RnD\DVS\Project_Funded\Current\2016_f001f_PA_Enable_S3\Work_01\14_UC4\02_Software_Installation\02_AVL_ModelCONNECT

You will find the installation guide there. After the installation of the program start the astlauncher.exe which can be found in the installation directory:

…\AVL\R2017b\bin

Then go to

AVL Advanced Simulation Technologies\Software Administration\Product Licensing\File

and copy the content of the file licensefile.txt to this field.

2.2 Installation of VSM

The installation file for VSM is located at

M:\FTG-P\3_RnD\DVS\Project_Funded\Current\2016_f001f_PA_Enable_S3\Work_01\14_UC4\02_Software_Installation\01_AVL_VSM

During installing the license file is requested. You can find it at the same location.

2.3 Installation of VTD

VTD was already installed on the LINUX-PC from VIRES. If you need further information for installing VTD you can contact

info@vires.com

2.4 Installation of MATLAB/Simulink

The installation file for MATLAB R2016a is located at

M:\FTG-P\3_RnD\DVS\Project_Funded\Current\2016_f001f_PA_Enable_S3\Work_01\14_UC4\02_Software_Installation\03_Matlab1016a

During installing the license key is requested. You can find it at the same location.

3 Simulation Software

The basics of all the simulation software can be learned by the user manuals of every program. Here only the necessary details to create the model for the traffic jam assistant are described

3.1 Model.CONNECT

Model.CONNECT is the Co-Simulation platform. In the tab “Project” projects can be loaded, saved and created. In the tab “Home” the model is created and the settings can be found. The tab “Simulation” is used to start the simulation and it offers the possibility to view data of the running simulation. In the tab “Results” the stored data of the simulation can be used to create charts.

3.1.1 Project

Here a new Project can be created by clicking New\Empty Project. In every project more models can be created by clicking New\Model.CONNECT™. The models of a project can be seen in the bar on the bottom of the window. It is recommended not to have more than 5 models in one project. Otherwise the project will take longer to load and save. Models inside a project can be copied and inserted via a right-click on the models in the bar on the bottom of the window.

3.1.2 Home

In this tab different simulation programs can be inserted. The used simulation programs in the Traffic Jam Assistant (TJA) – model are VTD, VSM, MATLAB and FMU. First an element of the specific program is dragged from the pane Components\Model Interfaces or Components\ICOS (Tool Interfaces) to the Topology\System pane (see Figure 1). (If the mentioned panes cannot be found go to Home\View\Panes and select them there.) The created files from the simulation program can be loaded in this element. This is done by going to the properties of the element.

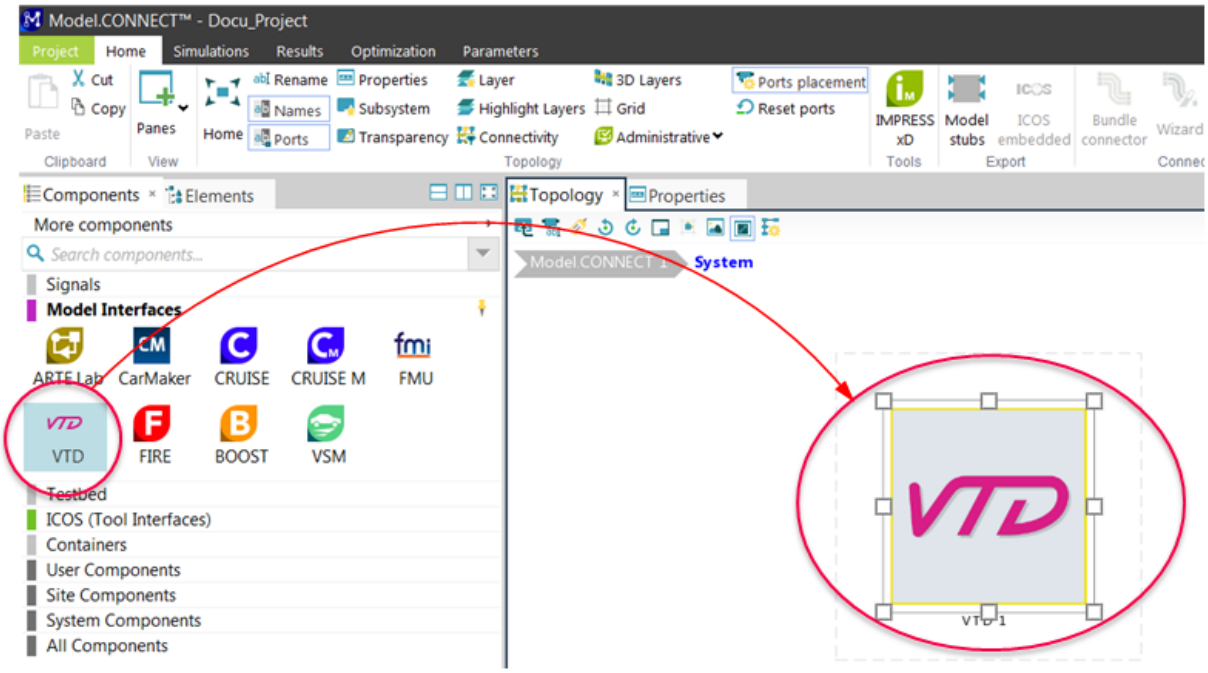

Figure 1: Insert a new Element i.e. VTD

In the tab Topology\Settings (see Figure 2) the simulation start- and end time can be changed. In Settings\Default\Simulation the step size of the Co-Simulation is set. It should always be larger than the simulation step sizes from the single simulation programs. Also the coupling mechanism can be changed. It is recommended to read the user manual detail to this chapter if you want to know more about these settings. In Settings\Default\Results the start and end time of stored data can be changed. This is useful to start saving data of a scenario at a time where the system is stable. Also the frequency in which data is stored can be changed. If a simulation has bugs it is recommended to check the boxes of "timing output" and "debug log" here. Then log-files for debugging are created at the project directory. The same settings can be changed for the "Default: ICOS" group. Those settings apply amongst others to the MATLAB/Simulink element.

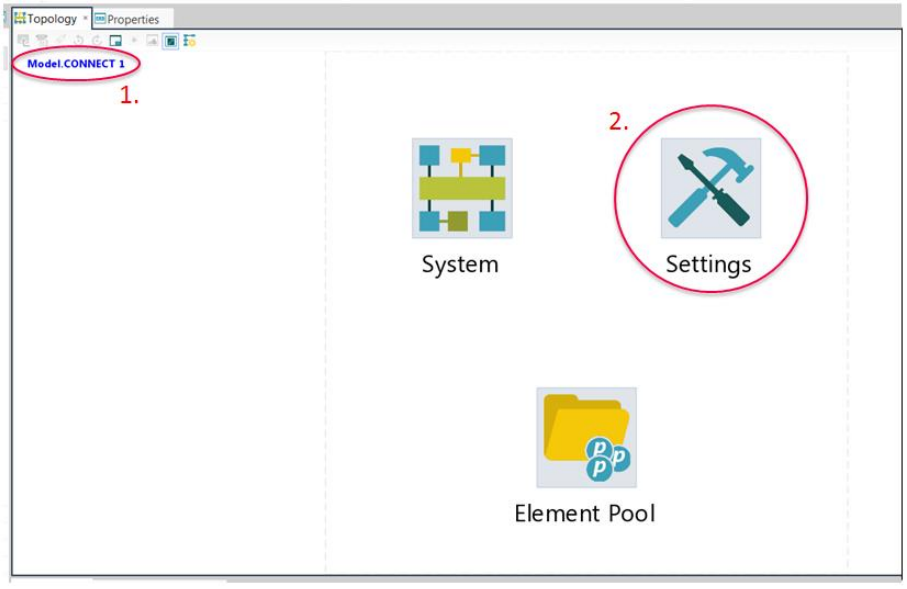

Figure 2: Model settings

3.1.3 Simulations

In this tab (see Figure 3) the simulation of the model created in the home tab can be started. Diagrams and Values of the current simulation data can be inserted and viewed during the simulation. During running of the simulation the simulation time is displayed in a bar at the bottom of the screen. In Job Control\Details the "Task information" is opened. Here the details of the simulation progress are displayed. Also error messages can be seen. If the "debug log" in home\Topology\Settings\Default\Results is checked, the log files can be opened in Task Information\Logs by clicking on "open directory". These log files are useful for debugging.

3.1.4 Results

After a completed simulation charts and tables can be created with the created .csv data files which are located at the project directory

"Project directory"\simulation\"Project Name"\"Project Name".Case_set_1.Case_1\results

This link contains the two files "mcx" and "icos". In "mcx" the VSM and VTD reults are saved and in "icos" the Simulink files are saved in the data format .csv. The .csv data files can also be loaded in other programs like MATLAB to create charts.

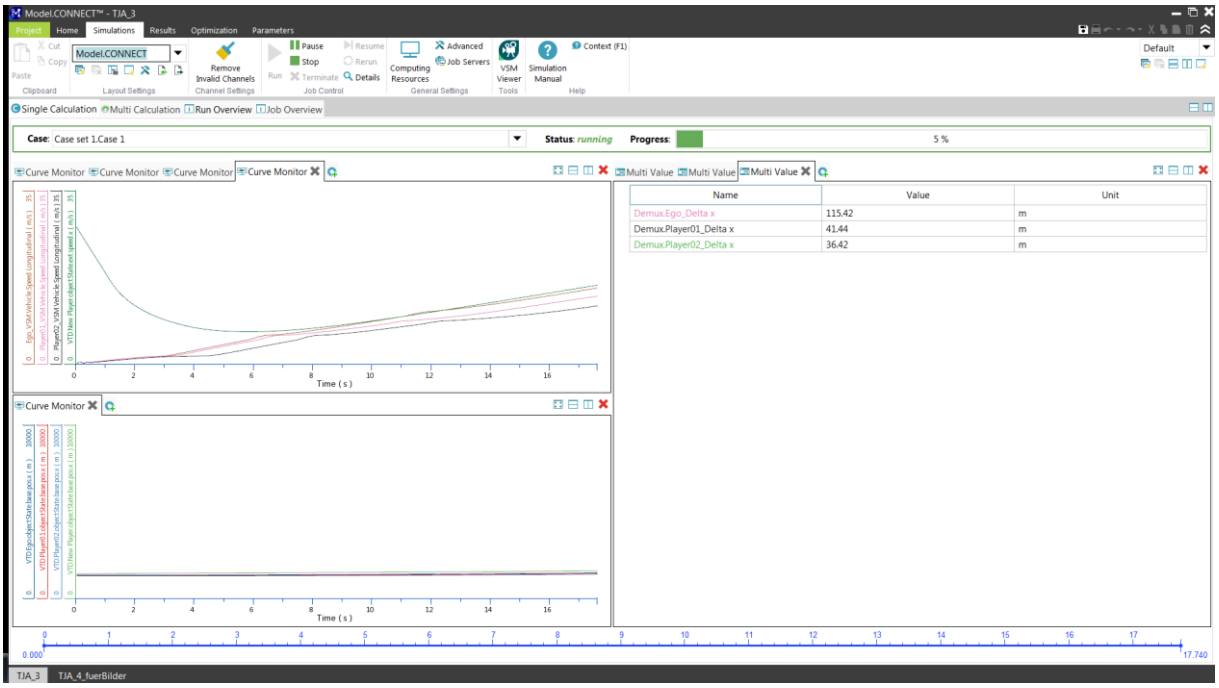

Figure 3: Simulation tab

3.2 VSM

In VSM (see Figure 4) the vehicle model is calculated. VSM is a standalone simulation program with included driver models. In the application of this co-simulation only the vehicle model is used. To load a project file in a VSM-element in Model.CONNECT first an individual VSM project (Vehicle model) has to be saved. When you open a new project a standard vehicle model is already implemented which can be modified. For the first Co-simulation the standard model was used. Therefore no further details to this program are described here.

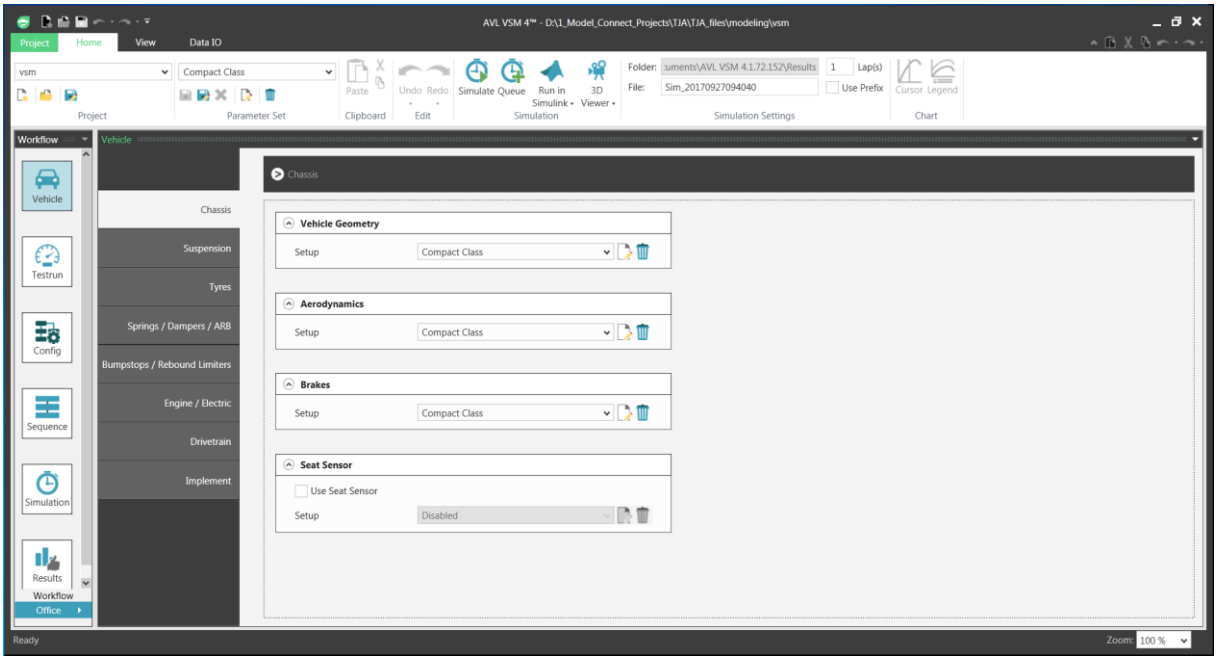

Figure 4: VSM

3.3 VTD

VTD runs on the LINUX-PC. Here the static and dynamic simulation environment of the co-simulation is created. VTD consists of three applications: the Road Designer, the Scenario Editor and the VTD-GUI. When a simulation is started, a new window with the graphical content of the simulation (simulation video) is opened.

3.3.1 VTD-GUI

This is the window (see Figure 5) that is opened when “Vtgui” is started from the desktop.

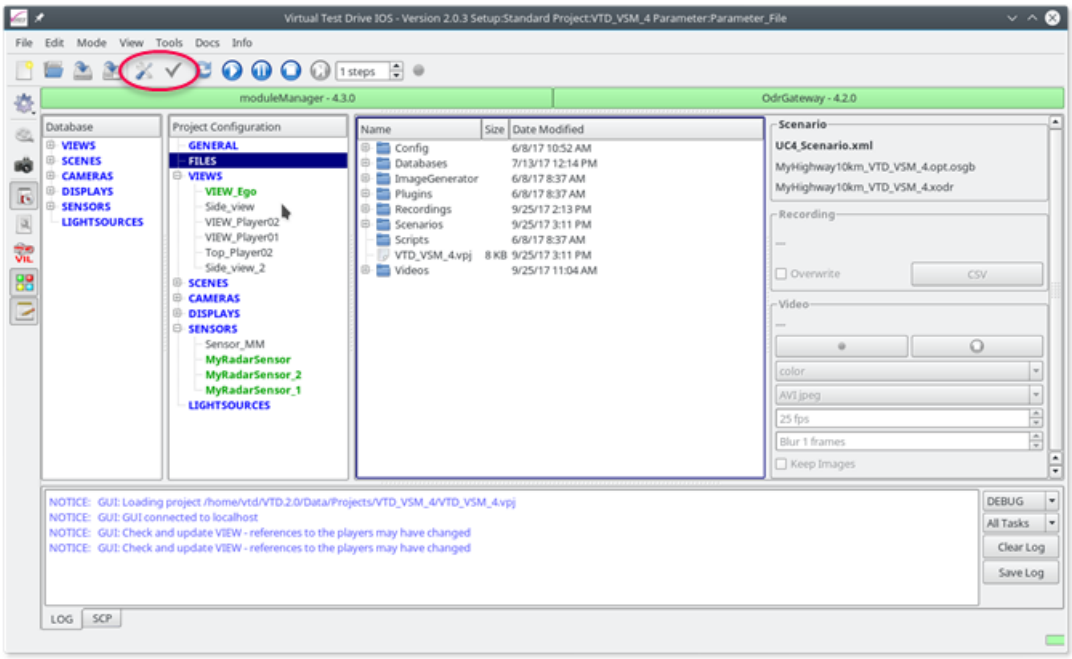

Figure 5: VTD GUI

In the GUI projects can be created and loaded. From here the Road Designer and the Scenario editor can be started via “Tools”. Basically the GUI has two states (see red marking in Figure 5). One is “Configuration” and one is “Apply Configuration”. It is recommended that changing settings of the simulation is done in the “Configuration” mode. Once everything is ready for simulation, switch to the “Apply Configuration” mode. In the area “Project Configuration” the following steps can be done (again only the necessary steps for this project are described):

- GENERAL: Project definition and author
- FILES:
    - Config: All configurations of the project. Only the module manager was changed. It is recommended not to modify anything here, unless you know what you do. The current setting work for the simulation environment of the TJA.
    - Recordings: By defining a new recording and press “Record” before a simulation starts, the video and other simulation data is saved. Then the simulation video can be replayed by switching from the “Operation” to the “Replay” mode. This is done in the tab “Mode” or by clicking on the gear-wheel symbol in the left bar.
    - Scenarios: Here the created scenarios can be loaded
- VIEWS: New views can be created. Those views can be activated (by a double click) during a running simulation. The active view is displayed bold and green.
- SCENES: Scenes are used to display more details in the simulation video and changing the environment like the weather. If sensors were defined you can activate the display of the sensor cone in the simulation video here.
- DISPLAY: The appearance of the simulation video can be modified here.
- SENSORS: Here new sensors can be added with a right-click on SENSORS. In Figure 5 4 sensors were created and 3 are activated. Sensors are activated by a double-click. If they are activated they appear bold and green. The following options for sensors are depicted in Figure 6.
    - General: the sensor model is loaded here. Sensor Plug-in can be loaded from the path */home/vtd/VTD2.0/Data/Distros/Current/Plugins/ModeuleManager*
      In this project only perfect sensors are used. The number on the right side is the number of objects the sensor is able to detect.
    - Com: This filed is important for the co-simulation since here the sensor is assigned to a vehicle and the TCP-Port number is called in Model.CONNECT to find the sensor.
    - Position: The sensor can be positioned relative to the vehicle coordinate system
    - Frustum: Here the geometry of the sensor cone is defined.

The sensors will forward the data to the Co-simulation in form of a vector. The vector has the size of the number of objects (vehicles) the sensor is able to detect. The data of the nearest vehicle is the first entry in the vector.

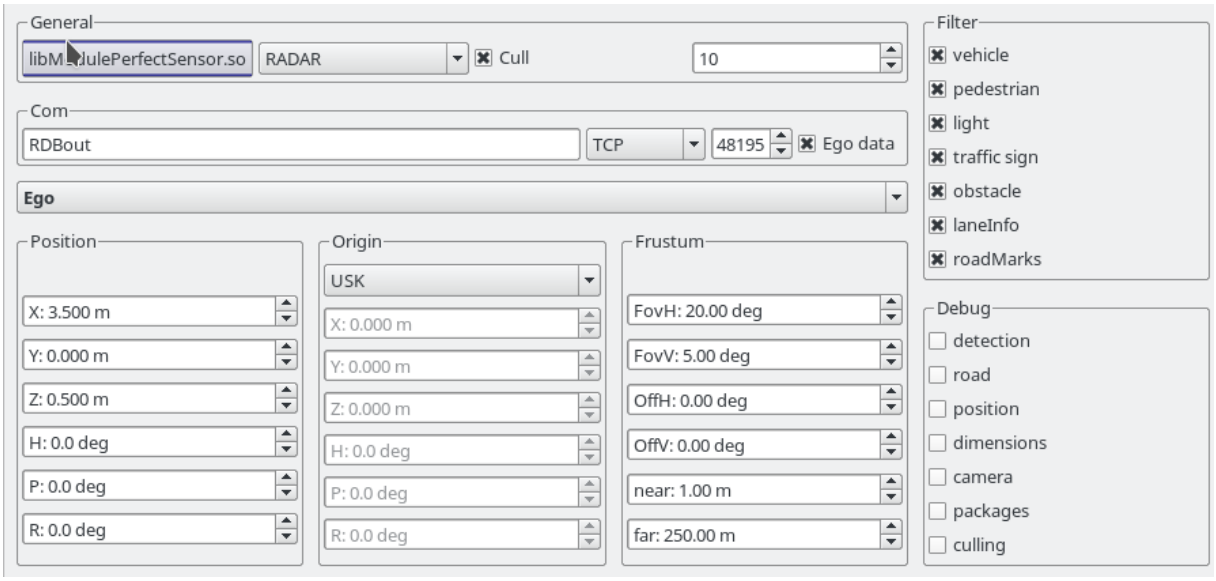

Figure 6: Options for the VTD sensors

To couple the simulation time from Model.CONNECT to VTD switch to the configuration mode and go to View\Show Parameterbrowser\Task Control\Sync and set “source” to “RDB”. Perform this change within the settings of the SETUP (source and RDB should then be green).

3.3.2 Road Designer (ROD)

In the Tutorial of the Road Designer it is described how to create roads and other static environment of the simulation. In the Road Designer (see Figure 7) new projects can be created. Within these projects new overlays (road designs) are created. It is recommended to use macros to quickly create standardized roads and environments. Once the road is created it has to be exported to use it in the Scenario Editor.

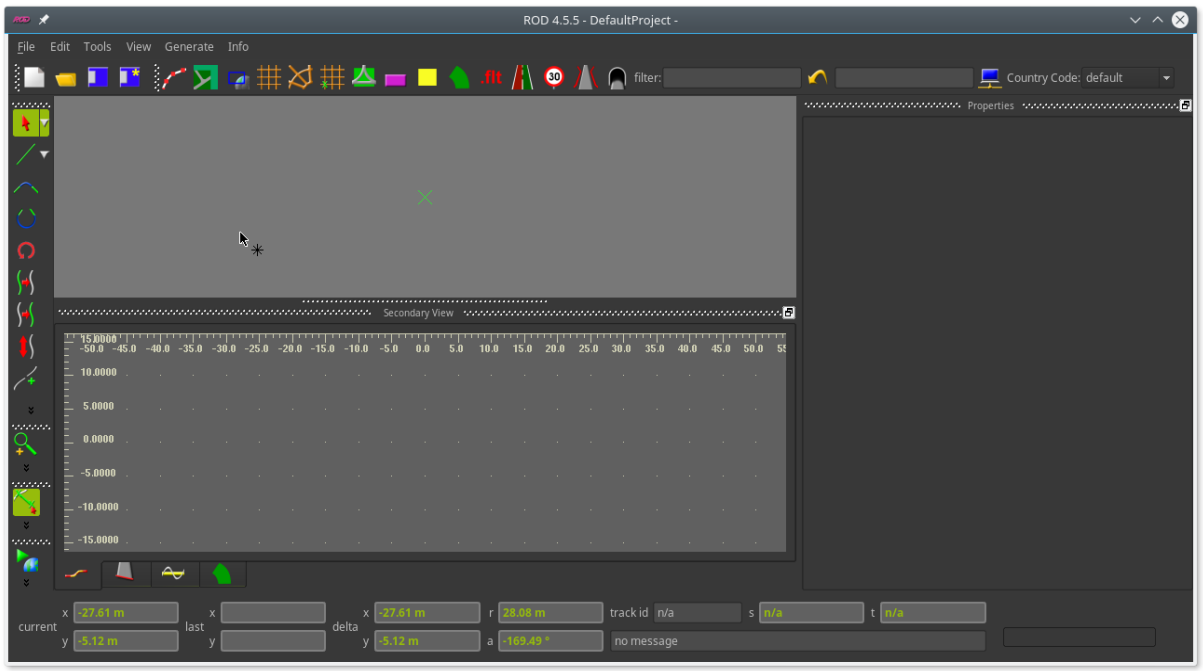

Figure 7: Road-Designer (ROD)

Therefore three files have to be exported. This is done by Generate\Database and Generate\OpenDRIVE. Those files are exported to (The following paths in this chapter refer to the LINUX-PC with VTD):

Home\vtd\VTD2.0\Runtime\Tools\ROD\"ROD-Project Name"\Odr

and

Home\vtd\VTD2.0\Runtime\Tools\ROD\"ROD-Project Name"\Database

Actually in this directory two files are created. One .opt.osgb and one .shadow.osgb file.

The OpenDRIVE file (.xodr) the logic information of the road and the database file (.osgb) contains the graphic environment.

Copy those three files to

Home\vtd\VTD2.0\Data\Projects\"Project Name"\Databases

From here the files can be loaded in the Scenario Editor.

3.3.3 Scenario Editor

In the scenario editor (see Figure 8) the dynamic content of the simulation environment is created. After creating a new scenario two files created from the Road Designer have to be loaded. Therefore go to Edit\Scenario Properties\Scene. In the layout file load the .xodr file and in the Visual Database load the .opt.osgb file.

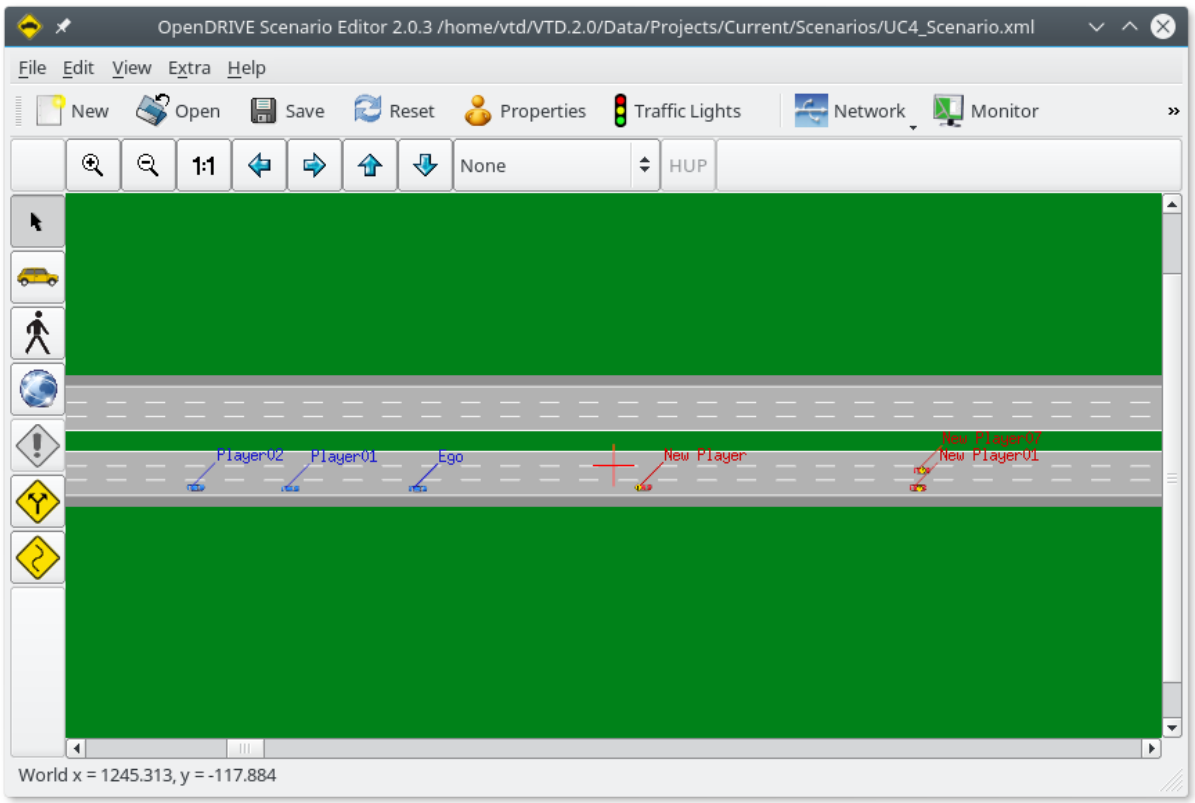

Figure 8: Scenario Editor

After loading those two files vehicles can be added to the scenario. In Figure 8 already 6 vehicles have been created by clicking on the vehicle symbol in the left bar and placing the vehicle anywhere on the road. By double-clicking at a place vehicle, the settings (see Figure 9) of a vehicle can be modified. There the vehicles can also be placed relative to other vehicles. At least on vehicle has to be set as "external" (see marking in Figure 9) to start the simulation. This external vehicle will then be calculated by VSM. All vehicles defined as "internal" are calculated by VTD. Their behaviour depends on the driver model that is assigned to them.

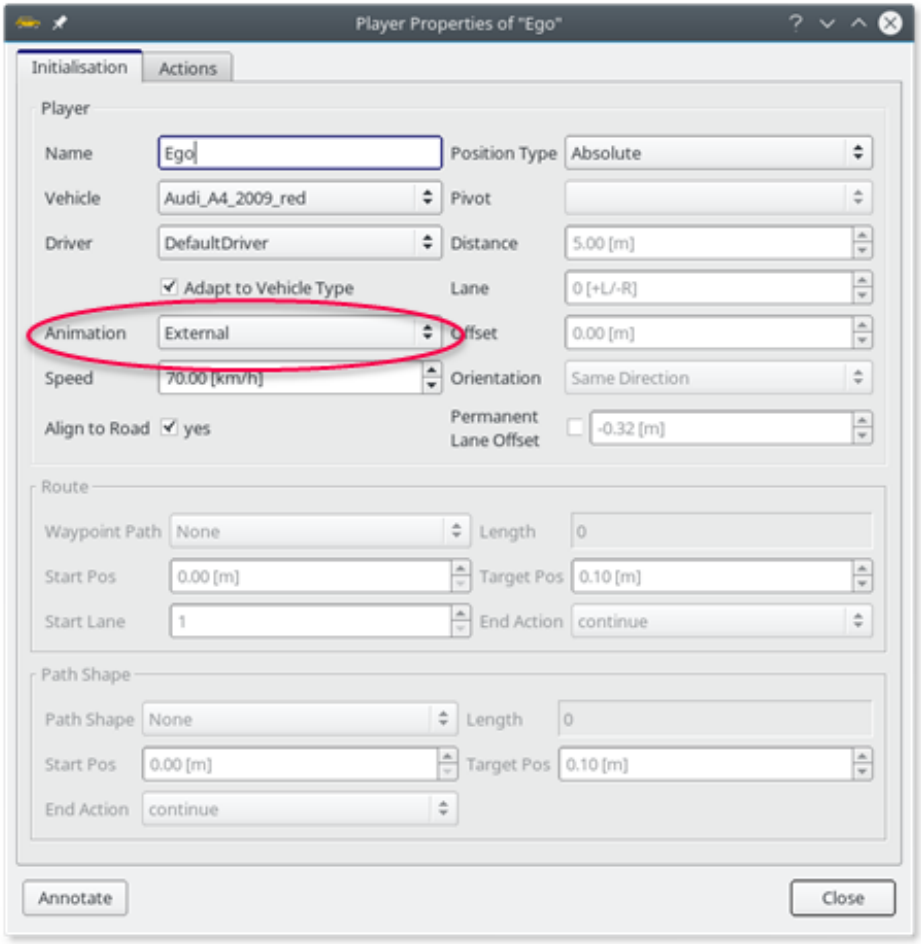

Figure 9: Vehicle settings - Initialisation

In the tab actions (see Figure 10) instructions like lane change of speed change can be given to the vehicle. The trigger of these actions can be based on time, global position or relative to other vehicles. Once an instruction is given to a vehicle the driver model is deactivated and the vehicle will perform the required tasks. The control can be given back to the driver model with the instruction "Autonomous".

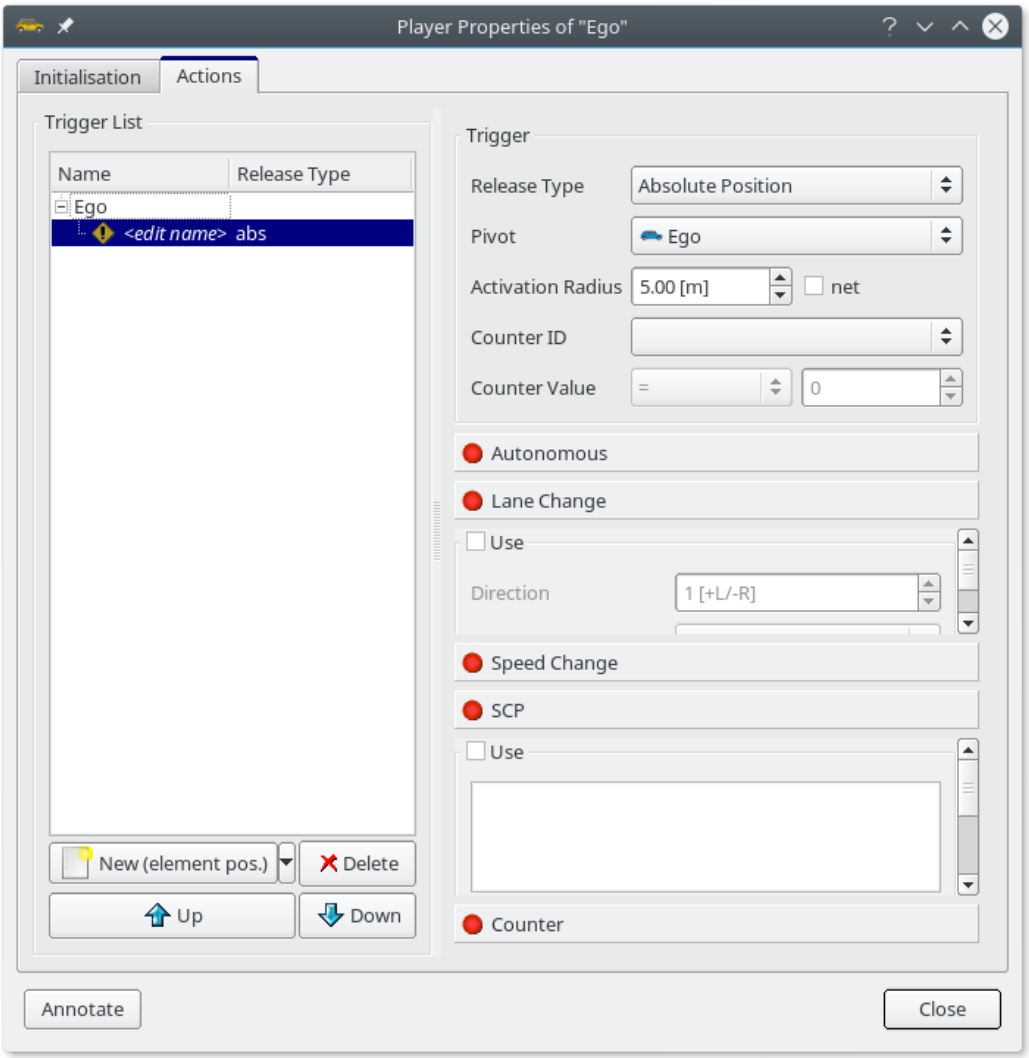

Figure 10: Vehicle settings - Actions

In Edit\Scenario Properties the following settings amongst others can be modified:

- Scene: loading the ROD-files (as described before)
- Vehicle definition
- Driver definition: change parameters of the driver model. This can for example be useful if you want a vehicle to drive autonomously in the traffic but it should not overtake other vehicles.
- Pulk Traffic: A pulk of random traffic can be assigned to an inserted vehicle. This is useful to create scenarios with congested roads.

After saving the scenario it can be loaded in the VTD-GUI as described in chapter 3.3.1.

3.4 MATLAB/Simulink

Simulink models are used in the co-simulation for the ACC, LKA and TJA controllers. The Simulink model is embedded in the Co-simulation via ICOS (Independent Co-Simulation) interface.

If the “ICOS Independent Co-Simulation”-Blocks are not located in the Simulink Library Browser, you can add them with the following steps:

1. Open MATLAB R2016a
2. Go to HOME\Environment\Set Path
3. Press “Add Folder...”
4. Search in the installation path of Model.CONNECT
   ...\AVL\R2017b\MC\icos\win64\tools\Matlab
5. Add the file ICOS_Library.slx

3.5 FMU / Coordinate – Transformation

The FMU for the coordinate-transformation between VSM and VTD is a single file which is loaded in the FMU-Element in Model.CONNECT. It is a blackbox model and cannot be modified.

4 Setting up a model

In this chapter the approach how to set up a new model is explained. First the implementation and setup of the simulation elements is described, followed by the coupling of the single simulation programs in Model.CONNECT.

As mentioned in chapter 2.1 the simulation elements in Model.CONNECT can be dragged from the components tab to the tab Topology\System. The properties of an element can be opened by a double-click or with right click\Properties. With a right-click the element can be renamed and the visual properties can be changed. The name of the element is important, since it is used to define each in- and outport.

After loading a file from any simulation program the file can be embedded into the project. This is done in the settings of an element by clicking on the button beside “choose file”. Then the file will be copied to the project directory

…\”Project Name”\”Project Name”_files\modeling

The advantage of an embedded file is that if the project is copied or “saved as” the file will stick to the project and does not have to be reloaded.

Basically all step sizes are allowed as long as the step size of the co-simulation at Settings\Default\Simulation is larger than the step sizes of the individual elements. For the TJA a co-simulation step size of 0.01 s is used. The MATLAB/Simulink step sizes are 0.005 and in VSM 0.0005 s. VTD has its own step sizes which cannot be changed in the settings.

4.1 VSM

In VSM the properties have to be opened with a right click\Properties. Then the edit properties window will appear. Here (see Figure 11) the VSM project can be loaded. By choosing a different simulation setup you can modify the VSM-file directly in the Model.CONNECT interface without opening the actual VSM-file. The simulation timestep must always be smaller than the simulation step size of the co-simulation. In Modules you can check and uncheck different modules of the VSM vehicle model. For the model TJA the settings like in Figure 11 should be used.

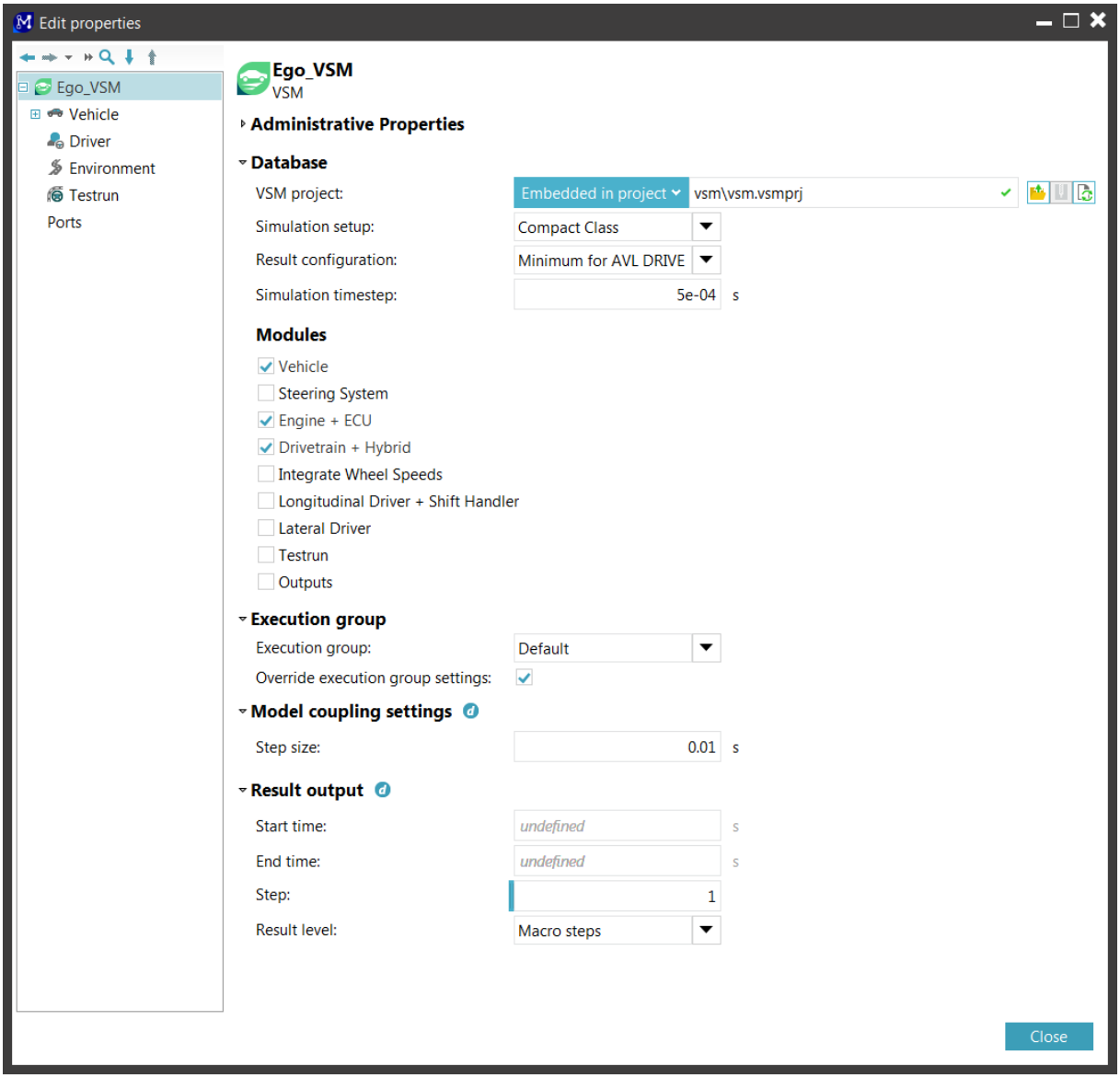

Figure 11: VSM properties

By checking the box at “Override execution group settings” one can override the settings described in chapter 3.1.2. The timespan of the result output and the frequency of data saving can also be modified for each simulation element. If the box is unchecked the global settings are kept.

4.2 VTD

In the VTD-Element properties (Figure 12) the hostname (IP-address) of the LINUX-PC where VTD is installed has to be specified. The current IP-address of the LINUX-PC is 129.27.119.52. The port is standardized and should not be changed. If these data are correct and VTD is in the “Apply Configuration” mode the scenarios, created in VTD, can be loaded. In the field sensor description the players of the chosen scenario are displayed. After “# Sensor description format:” The Sensors defined in VTD have to be entered manually in the described format. Then the sensor channels are available. Again one can override the global settings for this element by checking the box at “Override execution group settings”.

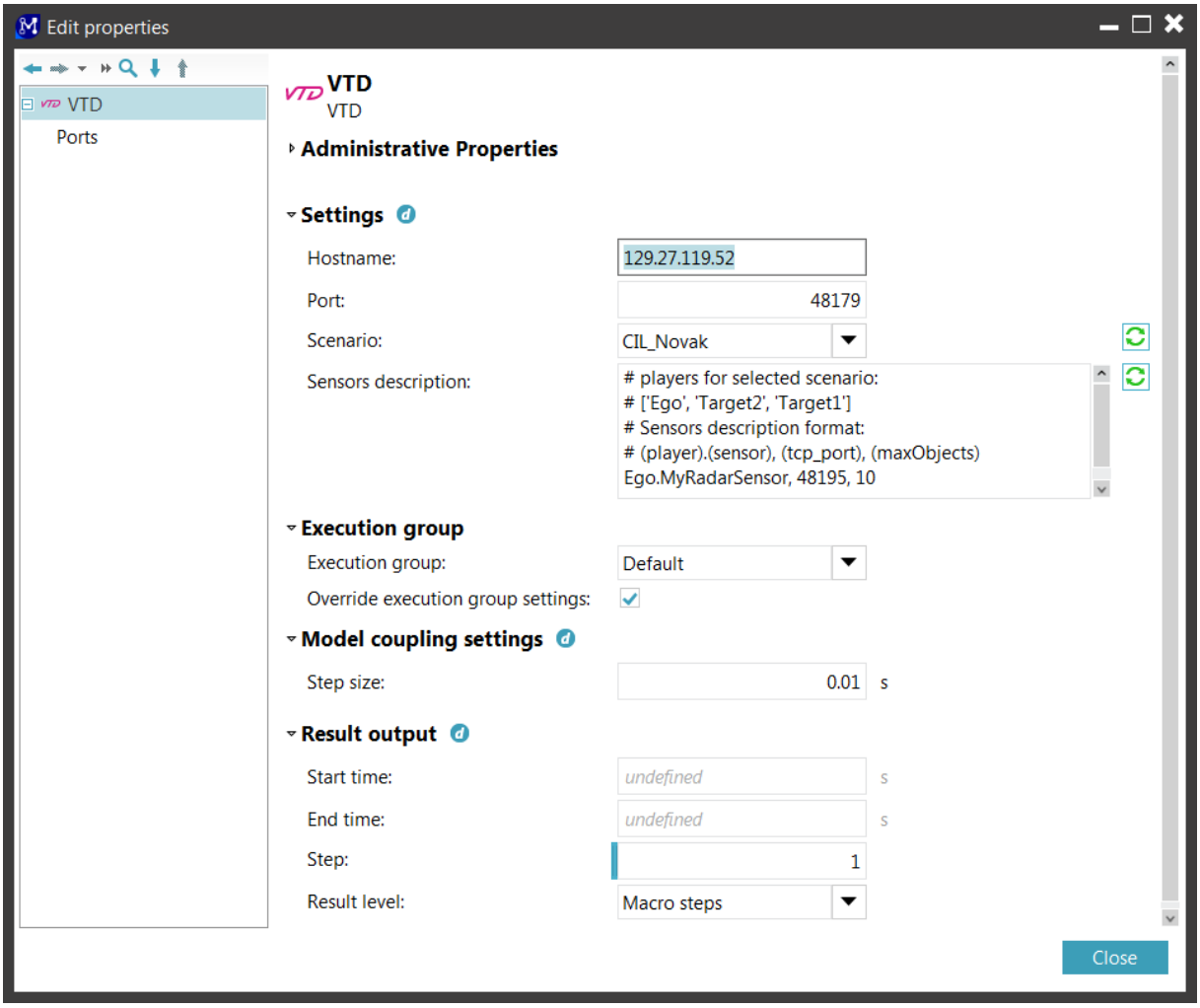

Figure 12: VTD properties

4.3 MATLAB/Simulink

In the MATLAB/Simulink properties one can load a Simulink file and if needed a MATLAB initial script. In Execution settings the platform and the location of the MATLAB executable file have to be specified. In this project MATLAB 2016a was used. The link to this executable is

D:\MATLAB\R2016a\bin\win64\MATLAB.exe

Again one can override the global settings for this element by checking the box at “Override execution group settings”.

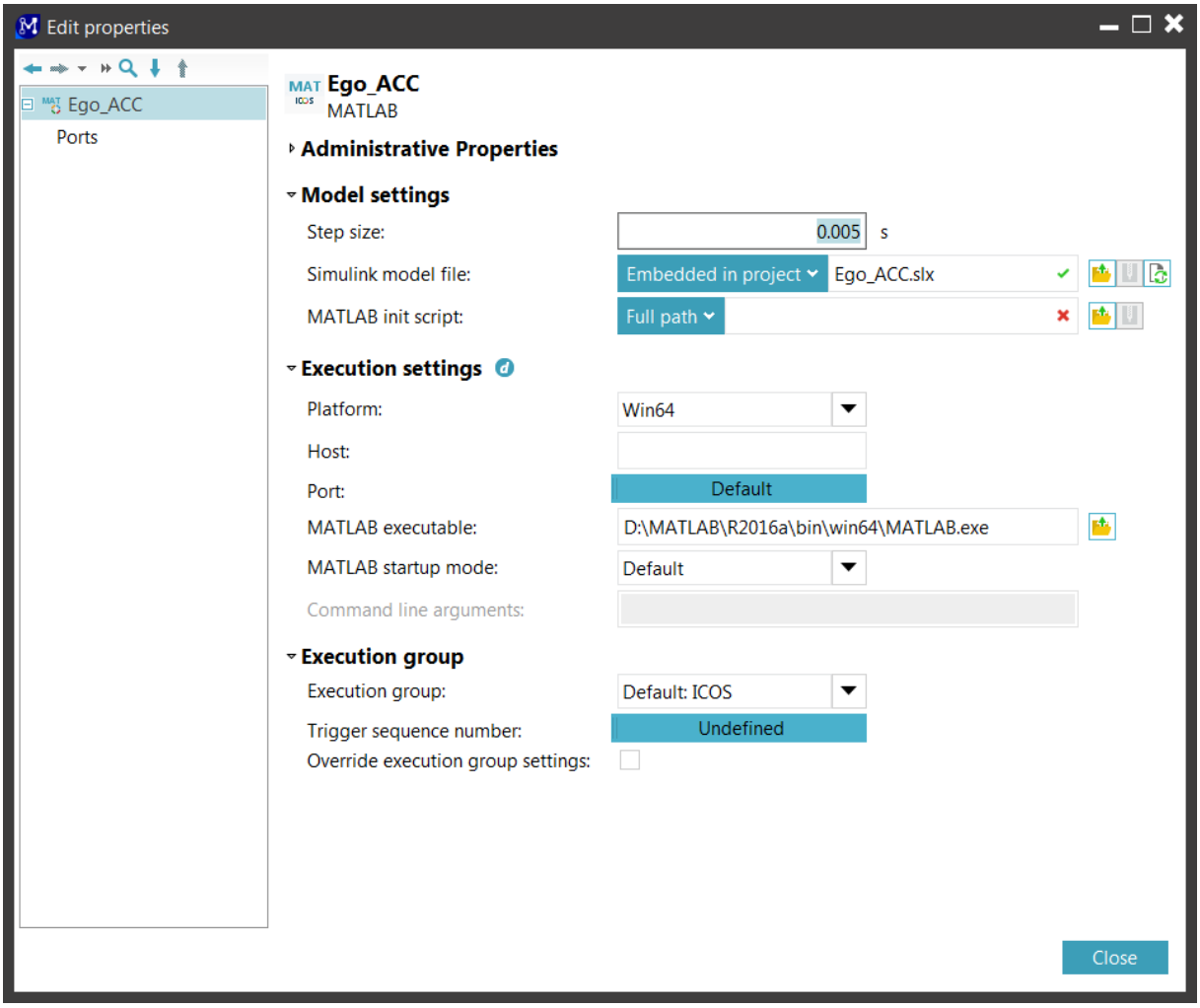

Figure 13: MATLAB/Simulink properties

4.4 FMU

In the FMU properties (Figure 14) a .fmu file can be loaded and the global settings can be overruled.

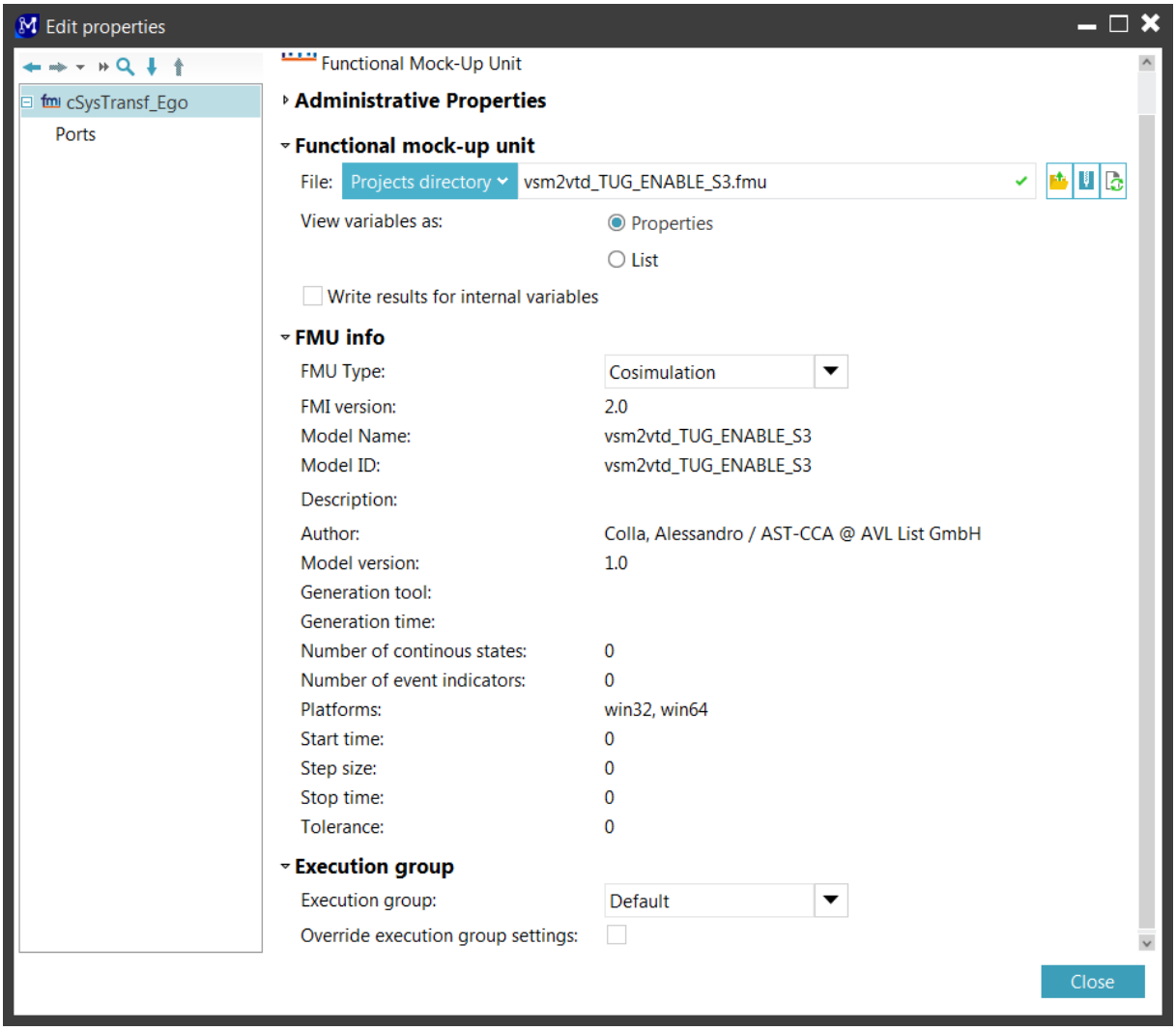

Figure 14: FMU properties

4.5 Constants

Constant values can be added to the simulation the same way as elements are dragged into the topology tab from \Components\Constant. These values can be connected to an input port of another element.

4.6 Ports and Bundles

All elements have in- and output ports which can be connected to other ports. In the Simulink the ports are defined by inserting ICOS blocks into the Simulink model. In VTD and VSM the ports are predefined and all needed ports can be selected. The port definition works in the same way for both elements. When loading a VSM project most ports are already selected. The following explanation how to add ports is done with a VTD-element. The procedure is the same as for a VSM element.

In Edit properties\Ports (Figure 15) the ports can be merged to bundles. For instance a bundle of VTD is “Ego.Vehicle”. In this bundle all the ports concerning the vehicle called “Ego” in VTD are bundled. This results in a better clarity of the whole model. Although belonging to a bundle, all ports are still individually accessible.

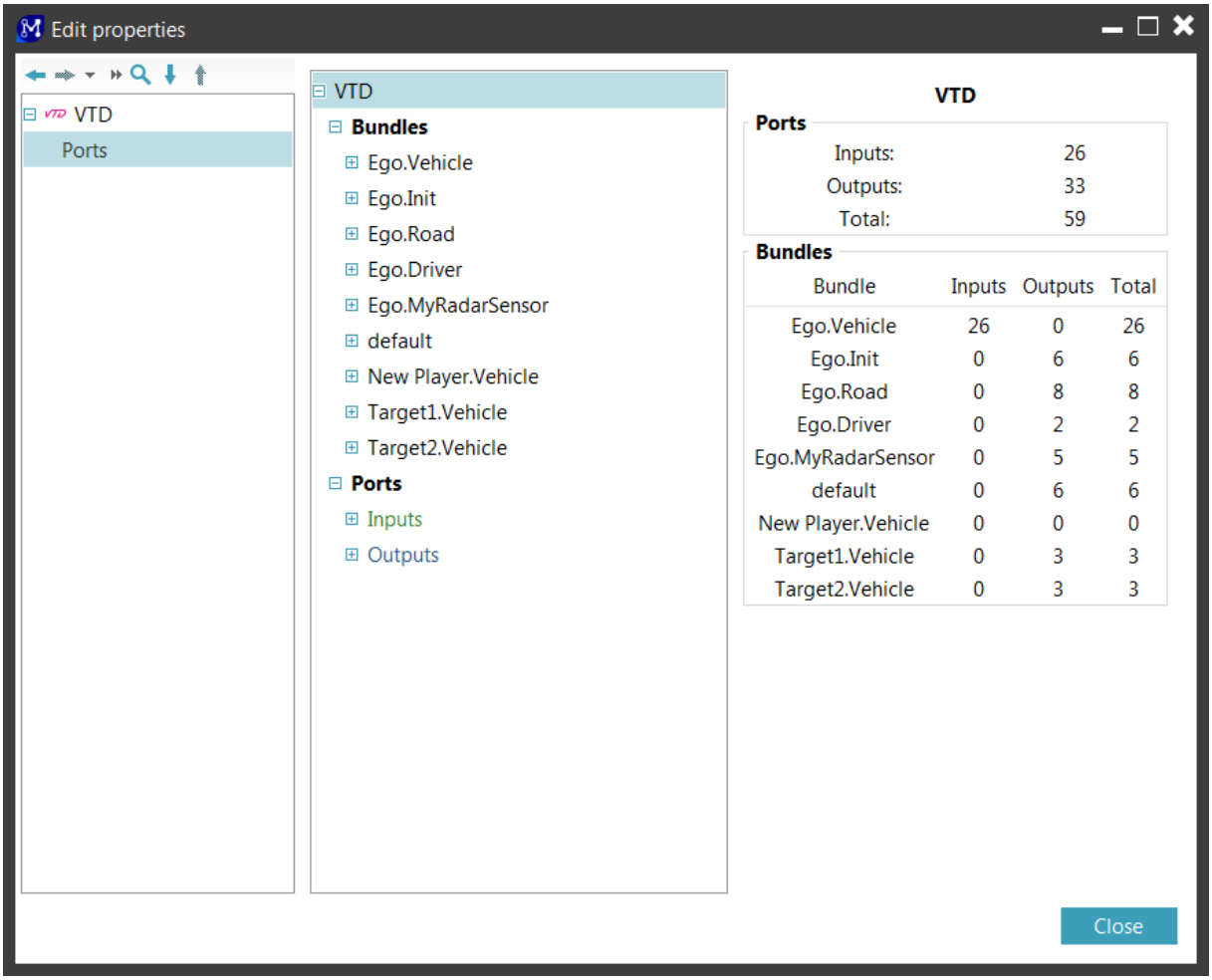

Figure 15: Ports of an element

The in- and output ports have to be loaded when a new model is set up. This is done by clicking on Ports\Inputs(or Outputs)\Add row (see Figure 16).

Figure 16: Port selection

After clicking on “Add row” the window in Figure 17 appears. There all required channels can be added. In this example the port “Ego.objectState.ext.speed.x” is already added and “Ego.objectState.ext.speed.y” is to be added.

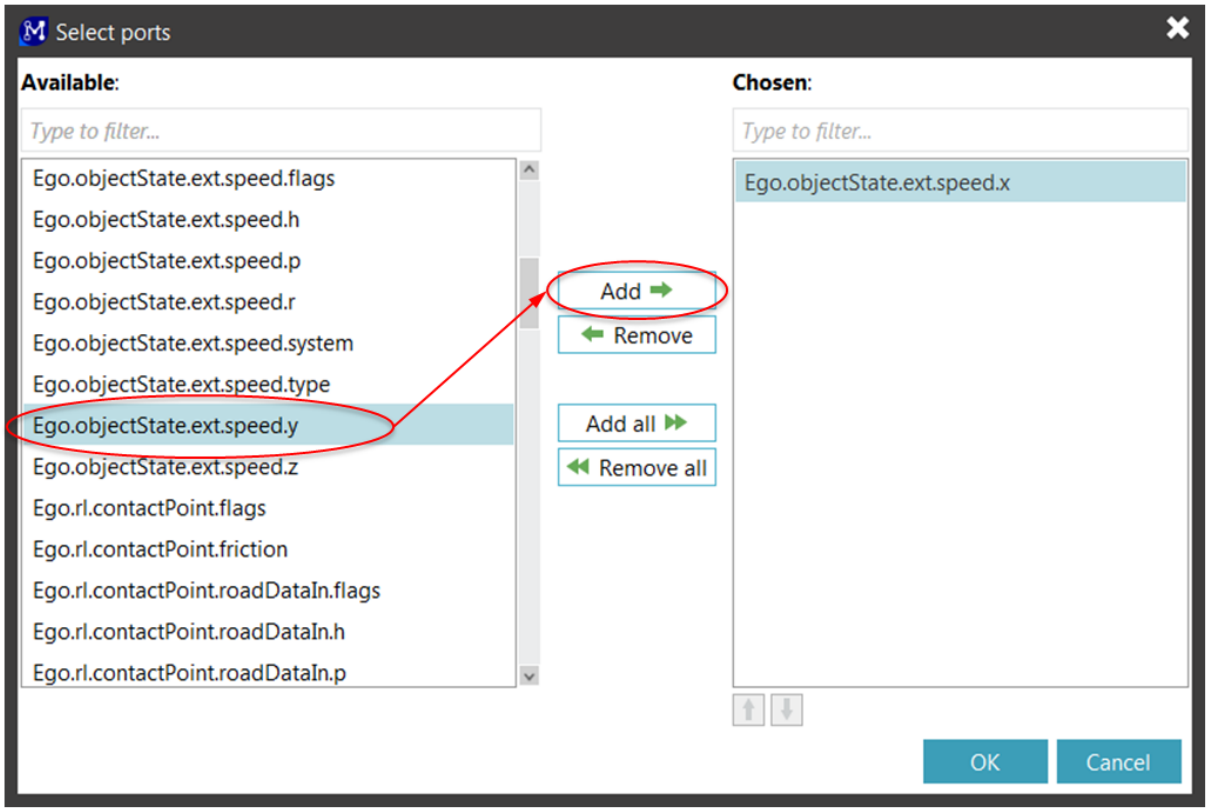

Figure 17: Port selection

After the ports have been selected they are displayed in the ports window (Figure 18) and their details can be viewed by clicking on them. They are automatically assigned to a predefined bundle. It is possible to change their name, type and unit, although it is recommended not to change this data. Model.CONNECT will automatically convert all units internally to SI-units when they are coupled with other elements. Here also limits, initial values and scaling factors for the values of the ports can be set.

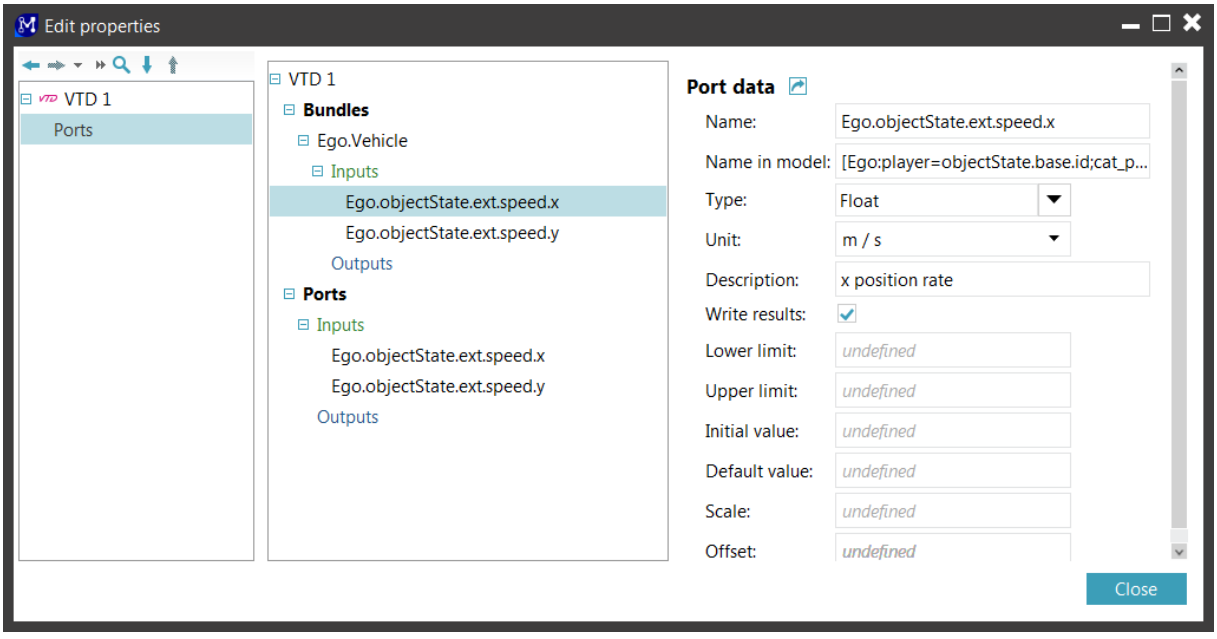

Figure 18: selected ports

Figure 19: Element with one bundle

After closing the "Edit properties" window the bundle will appear on the edge of the element (see Figure 19). Now the element is ready to be coupled with another element.

In a Simulink file the ports are added with ICOS blocks from the library directly in MATLAB/Simulink (see Figure 20)

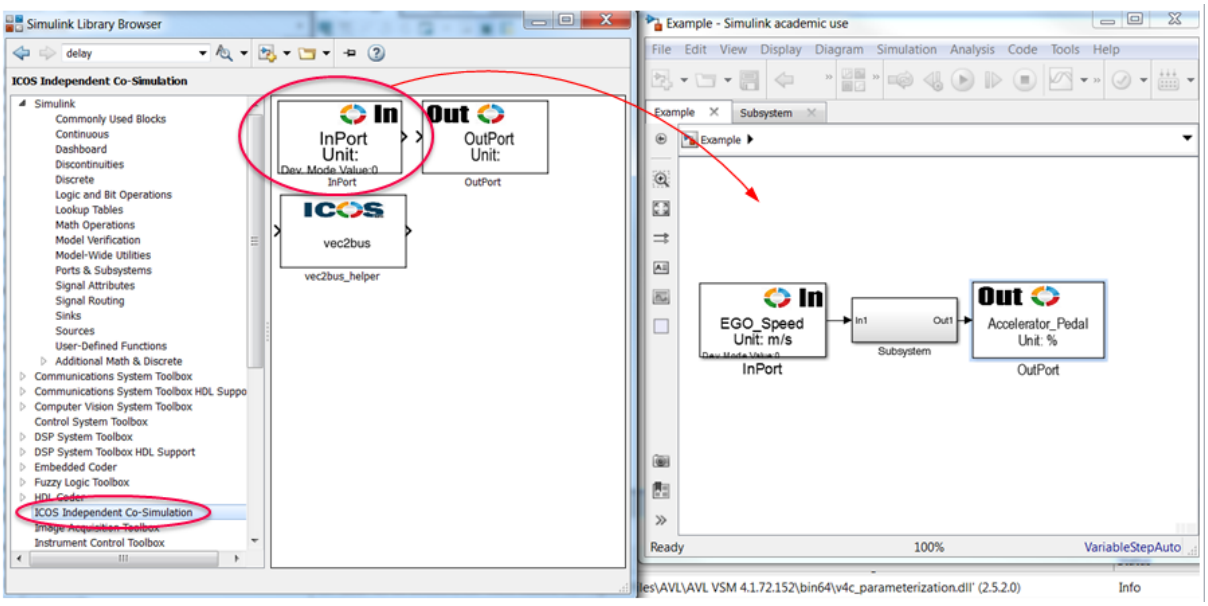

Figure 20: Inserting ports in a Simulink file

By double-clicking on an input or output block the properties of the ports like name, element and data type are selected (see Figure 21).

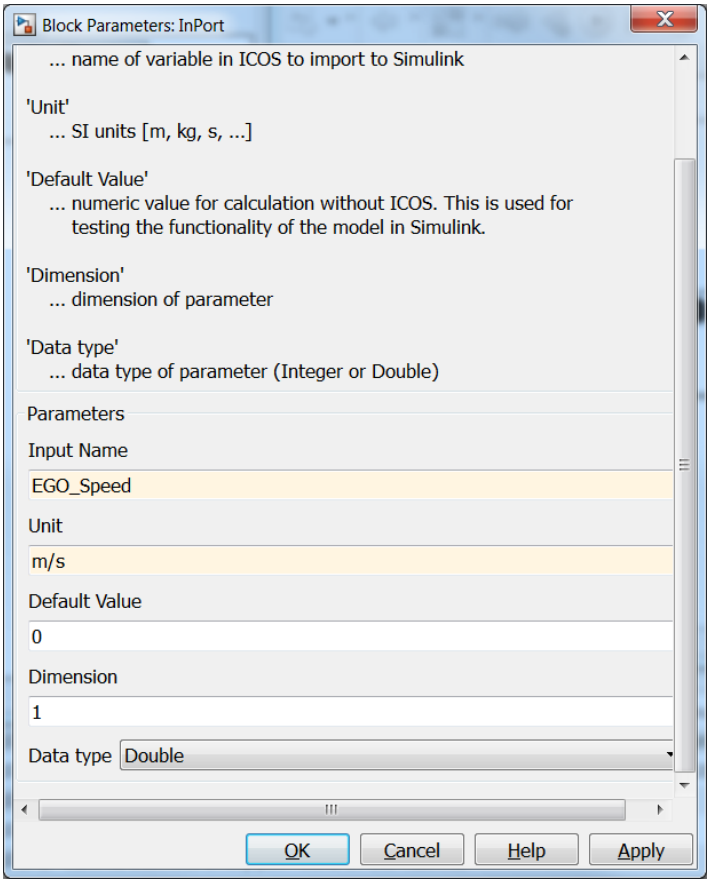

Figure 21: Port definition simulink

If input ports in VSM are not assigned their value is set 0. The input ports of all other elements have to be coupled otherwise the simulation won´t start. Output ports don't have to be assigned to input ports. Sometimes unassigned output ports are useful, because if the ouput port is loaded, data of this port can be displayed during simulation and the values are saved to the .csv.file.

4.7 Coupling of the elements

To start coupling elements the model has to consist of at least two elements. One with an input port and one with an output port. In the following example (see Figure 22) a VTD element, a VSM element and a MATLAB/Simulink element have been inserted in the model. The input ports "Ego.objectState.ext.speed.x" and "Ego.objectState.ext.speed.y" of the VTD element have already been added. A VSM element with a VSM project has also been added. The VSM ports are loaded automatically when loading a project. The amount of the bundles depends on the modules that are checked in the VSM properties, respectively how much information for the simulation of the vehicle model is required. The MATLAB/Simulink element has one outport "Steering angle" and the inport "Vehicle speed". This could represent a LKA (Lane Keeping Assistant) system that depends on the speed of the ego vehicle.

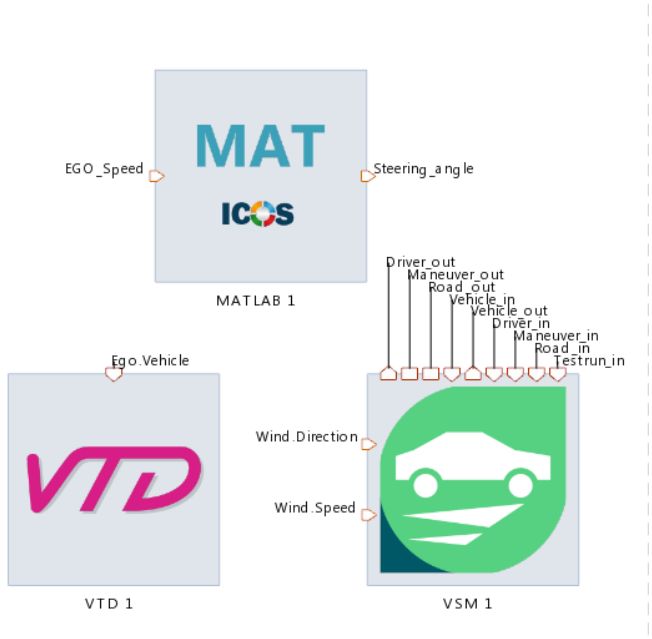

Figure 22: Simple model

To connect the ports of the elements, hold “Cntrl”-key and mark the two elements that should be coupled. Then go to Home\Connection\Bundle connector (see Figure 23)

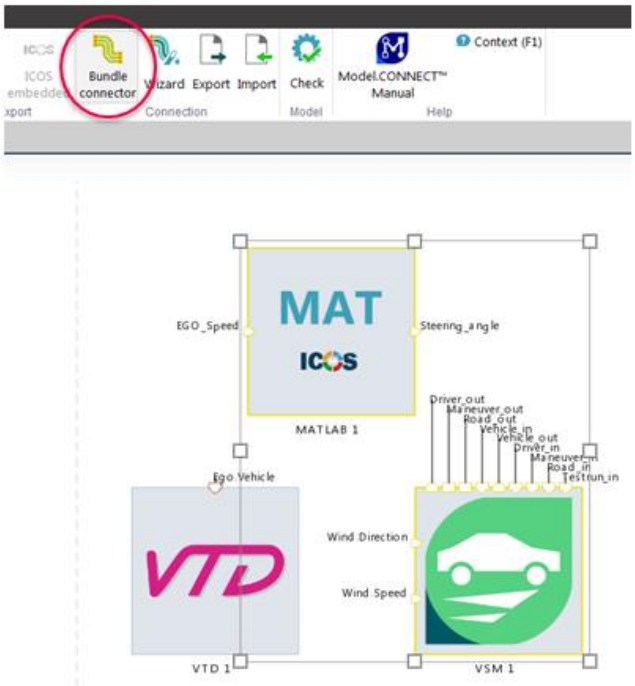

Figure 23: Bundle Connector

After clicking on the “Bundle connector” the window in Figure 24 opens. Here the in- and output ports of each element are listed. They can be sorted by bundles in the drop-down menu. The ports are coupled by clicking on the arrows beside one input- and one output port. Here the output port “Steering angle” of the MATLAB/Simulink element is connected to the input port “Driver.Steer Ange” of the VSM element and the output port “Vehicle.Speed Longitudenal” is connected to the input port “EGO_Speed”. After a port has been connected the arrow beside the port is filled.

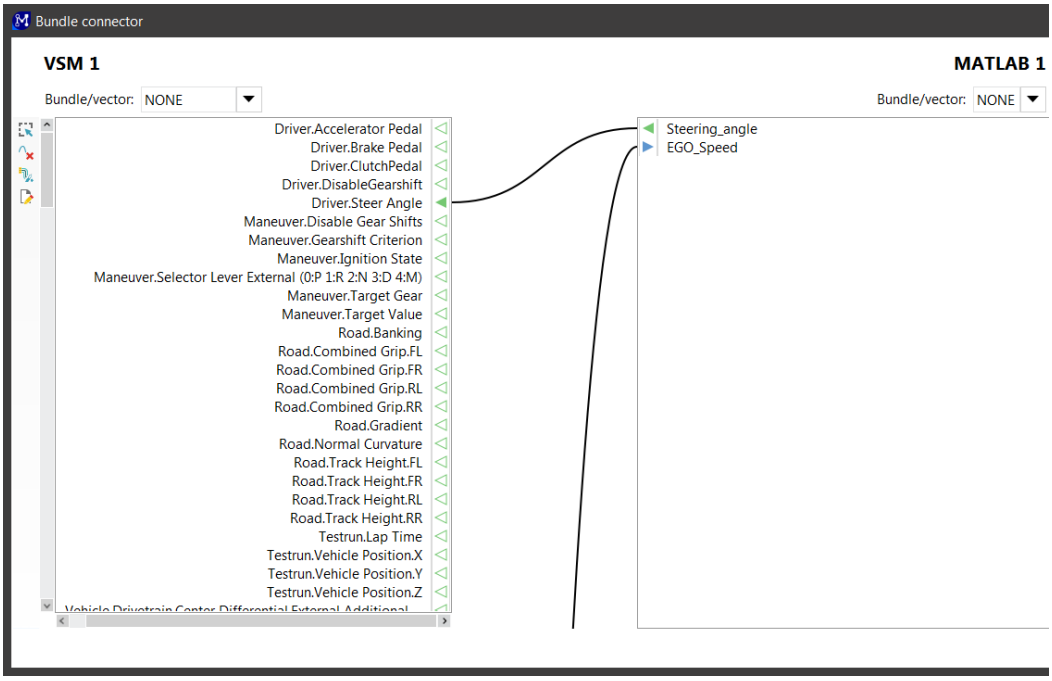

Figure 24: Connectiion of the ports

After closing the “Bundle connector” the model looks like Figure 25.

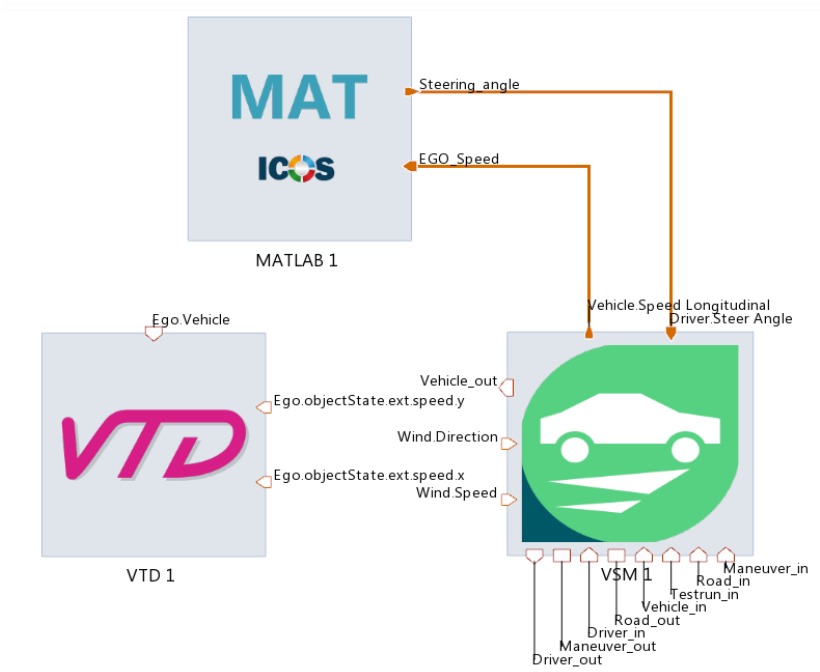

Figure 25: Model with connected ports

The ports and bundles on the edge of each element can be re-arranged by holding “Shift” and drag the ports to the desired location. An input port can receive data only from one output port. But it is possible to connect one output port to several input ports.

5 The TJA model

Until now all the necessary information to build up a model was described. In this chapter the actual TJA model (state 28.09.2017) is described. The model is located at

D:\1_Model_Connect_Projects\TJA_3

In Figure 26 the TJA model is depicted. It consists of 3 identical Ego-vehicles (red boxes). Each of them has its own ACC and LKA Controllers expanded with the platooning function, the VSM vehicle model and the coordinate transformation between VTD and VSM. The controllers are programmed in a simulink file. The MATLAB/Simulink element "TJA-Controller" is used to activate and deactivate the platooning function. The element "Lane Dependant Sensordata selection" is used to read out the sensor data from VTD and forward it to the controllers.

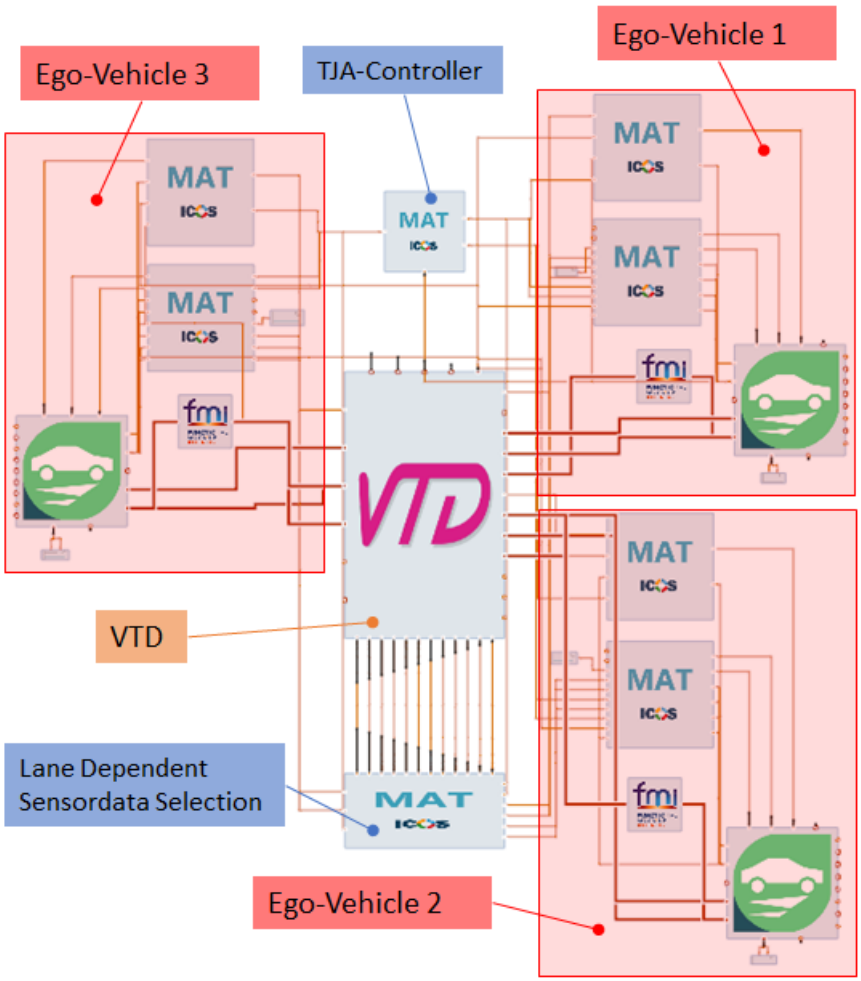

Figure 26: TJA model

5.1 Port Connections

The port connections of each Model.CONNECT model can be exported by clicking on Home\Connection\Export. The export file (TJA_3-conns.csv) of the TJA model can be found at

M:\FTG-P\3_RnD\DVS\Project_External\Current\2011_023f_IA_MSE_VDC-DAS\Work_01\00_Diplomarbeiten_Ergebnisse\00_11_DA_Kastner\03_Daten

Of course all connections can also be seen directly in the model as described in chapter 4.7 in the "Bundle connector".

5.2 Elements

This subchapter briefly describes the elements of the TJA-model. The files of the elements are all embedded in the project and can be found at

D:\1_Model_Connect_Projects\TJA_3\TJA_3_files\modeling

The elements from chapter 5.2.1 to 5.2.4 are identical for all ego vehicles. That means also the content of the simulink files of these elements are the same.

5.2.1 Ego_TJA (Ego_TJA.slx)

This is the ACC controller expanded with the platooning capability. The ACC controller is taken from the IPG CarMaker. The TJA-Controller switches between the ACC and the TJA mode. The Platooning function is implemented at the Simulink subsystem

Ego_TJA/AccelCtrl/ACC ECU/ACC Control/Distance Control Algorithm/desired ax

The desired gap between two vehicles can be changed at

Ego_TJA/AccelCtrl/ACC ECU/ACC User Input

The minimum distance between vehicles for platooning and in ACC-mode can be changed at

Ego_TJA/AccelCtrl/ACC ECU/ACC Control/Desired Distance Calculation with time gap

The desired speed of the ego vehicle is an input port which is connected to a constant. In this way the desired speed can be changed directly in Model.CONNECT and it is not necessary to open the Simulink file.

5.2.2 Ego_LKA (Ego_Steering_TJA.slx)

This is the LKA controller. It is expanded for the platooning function. In the normal ACC mode the LKA is active if the TJA switches to platooning mode. In the ACC mode the lane offset is controlled with a PID controller. The lane offset is directly delivered from VTD. No Sensor is used for this information, since the road mark information of sensors cannot be read out by Model.CONNECT yet. In the platooning mode the vehicles follow the position of the platooning leader with a certain time

delay which is set by a constant in the model. The time delay for following the platooning leader is an input port which is connected to a constant. In this way the following time can be changed directly in Model.CONNECT and it is not necessary to open the Simulink file.

5.2.3 Ego_VSM (vsm\vsm.vsmprj)

This is the VSM model. For the TJA the standard vehicle in VSM is used. The ignition state and the Selector Lever are defined by a constant.

5.2.4 cSysTransf_Ego (vsm2vtd_TUG_ENABLE_S3.fmu)

This is the coordinate transformation between VTD and VSM. It cannot be changed and has fixed in- and outports.

5.2.5 VTD

The VTD element loads the Scenario and defines the sensors that are used in the model.

5.2.6 TJA_Control_2 (TJA_Control2.slx)

With this model the states of the other controllers is set. Depending on the speed of the traffic vehicle the different modes are chosen. If the vehicle drives with a speed > 60 km/h no action is carried out. If the velocity decreases below 60 km/h the manual activation of the TJA is recommended. After the TJA has been manually activated the platooning function will automatically be activated if the traffic speed decreases below 20 km/h.

5.2.7 Lane Dependant Sensordata Selection (Demux_LDACC_TJA.slx)

As described in chapter 3.3.1 the sensor data from VTD is a vector. In this Simulink file the target vehicle for the other controllers is selected from the vector of vehicles. In case of the Lane Dependant Sensordata Selection the nearest vehicle which is on the same lane as the ego vehicle is chosen as target vehicle. This algorithm is carried out for the three sensors of each ego vehicle. Therefore there the same model three times in this file.

5.3 VTD-Settings

The VTD Project for the TJA-Model is

/home/vtd/VTD2.0/Data/Projects/VTD_VSM_4/VTD_VSM_4.vpj

and the Scenarios are located at

/home/vtd/VTD2.0/Data/Projects/VTD_VSM_4/Scenarios

5.4 Adding a new Ego-Vehicle

There are more ways how to implement a new ego vehicle. The following steps are one possibility how to do it.

1. Copy all elements inside one red box from Figure 26 inside the model.
2. Copy all files that belong to the copied elements from *D:\1_Model_Connect_Projects\TJA_3\TJA_3_files\modeling*
   These are Ego_Sterring_TJA.slx, Ego_TJA.slx and the vsm folder with the project. An exception is the coordinate transformation. The same .fmu file is used for all ego vehicles.
3. Rename the new elements
4. Expand the TJA_Control2.slx to four ego-vehicles
5. Expand the Demux_LDACC_TJA.slx to four ego vehicles.
6. Create a scenario in VTD for 4 externally controlled vehicles (see chapter 3.3.3)
7. Define a new sensor in VTD (see chapter 3.3.1) and assign it to the new externally controlled vehicle.
8. Define the new sensor in the VTD element in Model.CONNECT (see chapter 4.2)
9. Load the necessary in- and output ports in the VTD element. The necessary channels can be found by looking at the ports existing ego vehicles. (see chapter 4.6)
10. Connect the ports of all the new elements the same way as the existing ego vehicles (see chapter 4.7)

It would also be possible to export the port configuration of the old model, copy all the necessary entries for a new ego vehicle in the exported file, rename them, save the file and import it to the model. Since the efforts to find out the necessary channels under all exported channels is very high, it is recommended to connect the ports manually step by step for each element.

5.5 Used Ports and description

The following Table 1contains all VTD- and VSM-ports that have been used in the TJA-model. They are sorted by bundles.

Table 1: Used Ports for TJA-model

Element	Bundle / Port Name	Description
VTD	**Bundle: "PlayerName".Vehicle**	**Vehicle inports**
	"PlayerName".fl.roadQuery.x	x co-ordinate (inertial) of query location
	"PlayerName".fl.roadQuery.y	y co-ordinate (inertial) of query location
	"PlayerName".fr.roadQuery.x	x co-ordinate (inertial) of query location
	"PlayerName".fr.roadQuery.y	y co-ordinate (inertial) of query location
	"PlayerName".rl.roadQuery.x	x co-ordinate (inertial) of query location
	"PlayerName".rl.roadQuery.y	y co-ordinate (inertial) of query location
	"PlayerName".rr.roadQuery.x	x co-ordinate (inertial) of query location
	"PlayerName".rr.roadQuery.y	y co-ordinate (inertial) of query location
	"PlayerName".fl.wheel.base.springCompression	compression of spring front left

Element	Bundle / Port Name	Description
	“PlayerName”.fr.wheel.base.springCompression	compression of spring front right
	“PlayerName”.rl.wheel.base.springCompression	compression of spring rear left
	“PlayerName”.rr.wheel.base.springCompression	compression of spring rear right
	“PlayerName”..objectState.ext.speed.x	x velocity
	“PlayerName”..objectState.ext.speed.y	y velocity
	“PlayerName”.objectState.ext.speed.z	z velocity
	“PlayerName”.objectState.ext.speed.h	heading angular velocity
	“PlayerName”.objectState.ext.speed.p	pitch angular velocity
	“PlayerName”.objectState.ext.speed.r	roll angular velocity
	“PlayerName”.fl.wheel.base.steeringAngle	wheel steering angle front left
	“PlayerName”.fr.wheel.base.steeringAngle	wheel steering angle front right
	“PlayerName”.objectState.base.pos.x	x position
	“PlayerName”.objectState.base.pos.y	y position
	“PlayerName”.objectState.base.pos.z	z position
	“PlayerName”.objectState.base.pos.h	heading position
	“PlayerName”.objectState.base.pos.p	pitch position
	“PlayerName”.objectState.base.pos.r	roll position
VTD	**Bundle: “PlayerName”.Init**	**Initial positions outports**
	[initial]“PlayerName”.x	initial x position
	[initial]“PlayerName”.y	initial y position
	[initial]“PlayerName”.z	initial z position
	[initial]“PlayerName”.h	initial heading position
	[initial]“PlayerName”.p	initial pitch position
	[initial]“PlayerName”.r	initial roll position
VTD	**Bundle: “PlayerName”.Road**	**Road information outports**
	“PlayerName”.fl.contactPoint.roadDataIn.z	z road position of wheel front left
	“PlayerName”.fr.contactPoint.roadDataIn.z	z road position of wheel front right
	“PlayerName”.rl.contactPoint.roadDataIn.z	z road position of wheel rear left
	“PlayerName”.rr.contactPoint.roadDataIn.z	z road position of wheel rear right
	“PlayerName”.fl.contactPoint.friction	road friction at contact point
	“PlayerName”.fr.contactPoint.friction	road friction at contact point
	“PlayerName”.rl.contactPoint.friction	road friction at contact point
	“PlayerName”.rr.contactPoint.friction	road friction at contact point
VTD	**Bundle: “PlayerName”.”SensorName”**	**Sensor outports**
	“PlayerName”.”SensorName”.objectState.base.pos.x	relative x position of the detected object to the vehicle with the sensor
	“PlayerName”.”SensorName”.objectState.base.pos.y	relative y position of the detected object to the vehicle with the sensor
	“PlayerName”.”SensorName”.objectState.ext.speed.x	relative x velocity of the detected object to the vehicle with the sensor
	“PlayerName”.”SensorName”.roadPos.laneId	lane ID of the detected object
VTD	**no bundle**	
	Ego.roadPos.laneOffset	lane offset, positive left
VSM	**Bundle: Vehicle_out**	**Vehicle outports**
	Vehicle.Steer Angle Wheel.FL	wheel steering angle front left
	Vehicle.Steer Angle Wheel.FR	wheel steering angle front right
	Vehicle.Vertical Wheel Travel.FL	compression of spring front left

Element	Bundle / Port Name	Description
	Vehicle.Vertical Wheel Travel.FR	compression of spring front right
	Vehicle.Vertical Wheel Travel.RL	compression of spring rear left
	Vehicle.Vertical Wheel Travel.RR	compression of spring rear right
	Vehicle.Body Angular Velocity Pitch	pitch angular velocity
	Vehicle.Body Angular Velocity Roll	roll angular velocity
	Vehicle.Heading	heading position
	Vehicle.Pitch Angle	pitch angle
	Vehicle.Roll Angle	roll angle
	Vehicle.Speed Lateral	y velocity
	Vehicle.Speed Longitudinal	x velocity
	Vehicle.Acceleration Longitudenal	x acceleration
	Vehicle.Wheel Contact Point Position TrackX_FL	x road position of wheel front left
	Vehicle.Wheel Contact Point Position TrackX_FR	x road position of wheel front right
	Vehicle.Wheel Contact Point Position TrackX_RL	x road position of wheel rear left
	Vehicle.Wheel Contact Point Position TrackX_RR	x road position of wheel rear right
	Vehicle.Wheel Contact Point Position TrackY_FL	y road position of wheel front left
	Vehicle.Wheel Contact Point Position TrackY_FR	y road position of wheel front right
	Vehicle.Wheel Contact Point Position TrackY_RL	y road position of wheel rear left
	Vehicle.Wheel Contact Point Position TrackY_RR	y road position of wheel rear right
VSM	**Bundle: Road_in**	**Road inports**
	Road.Combined Grip.FL	road friction at contact point
	Road.Combined Grip.FR	road friction at contact point
	Road.Combined Grip.RL	road friction at contact point
	Road.Combined Grip.RR	road friction at contact point
	Road.Track Height.FL	z road position of wheel front left
	Road.Track Height.FR	z road position of wheel front right
	Road.Track Height.RL	z road position of wheel rear left
	Road.Track Height.RR	z road position of wheel rear right
	Road.Banking	banking of the road
	Road.Gradient	gradient of the road
VSM	**Bundle: Maneuver_in**	**Maneuver imports**
	Maneuver.Ignition State	ignition state
	Maneuver.Selector Lever External (0:P 1:R 2:N 3:D 4:M)	Gear selection
VSM	**Bundle: Driver_in**	**Maneuver imports**
	Driver.Accelerator Pedal	accelerator pedal
	Driver Brake Pedal	brake pedal
	Driver.Steer angle	steer angle

6 Simulation of the Model

This chapter describes how to start and stop a simulation and how the ICOS embedded mode works.

6.1 Starting and Stopping the Simulation

The co-simulation is started by clicking on Simulation\Job Control\Run. Then the model is loaded and simulated. VTD has to be in the “Apply Configuration” mode and is started automatically by Model.CONNECT. Before starting the co-simulation in Model.CONNECT, run the simulation in VTD on the LINUX-PC for about 5 seconds and stop it.

During the simulation the all ports can be displayed over the simulation-time by creating charts and load the ports there. (see chapter 3.1.3).

The simulation can be paused, stopped and terminated. If you want to quit the simulation regularly, press “Stop”. Sometimes this might take too long or the simulation has an error and the simulation hangs up in the status “stopping”. In this case click “Terminate”. If the simulation still freezes in the state “terminating” press “Terminate” again. Now the simulation quits and the status switches to “terminated”.

If the simulation fails the status “failed” will be displayed.

6.2 Starting from MATLAB/Simulink

If you want to run the model and see the Simulink simulation as if only the Simulink file is simulated, Model.CONNECT offers the “ICOS Embedded” mode at Home\Export\ICOS embedded. This can be useful during the development of the controllers. After clicking on the ICOS embedded button (see small red circle in Figure 27), Model.CONNECT asks for the Simulink file from that the simulation should be started (large red circle in Figure 27). After choosing the file the simulation starts the preparation of all elements. The actual simulation starts when you press the play button of the Simulink file.

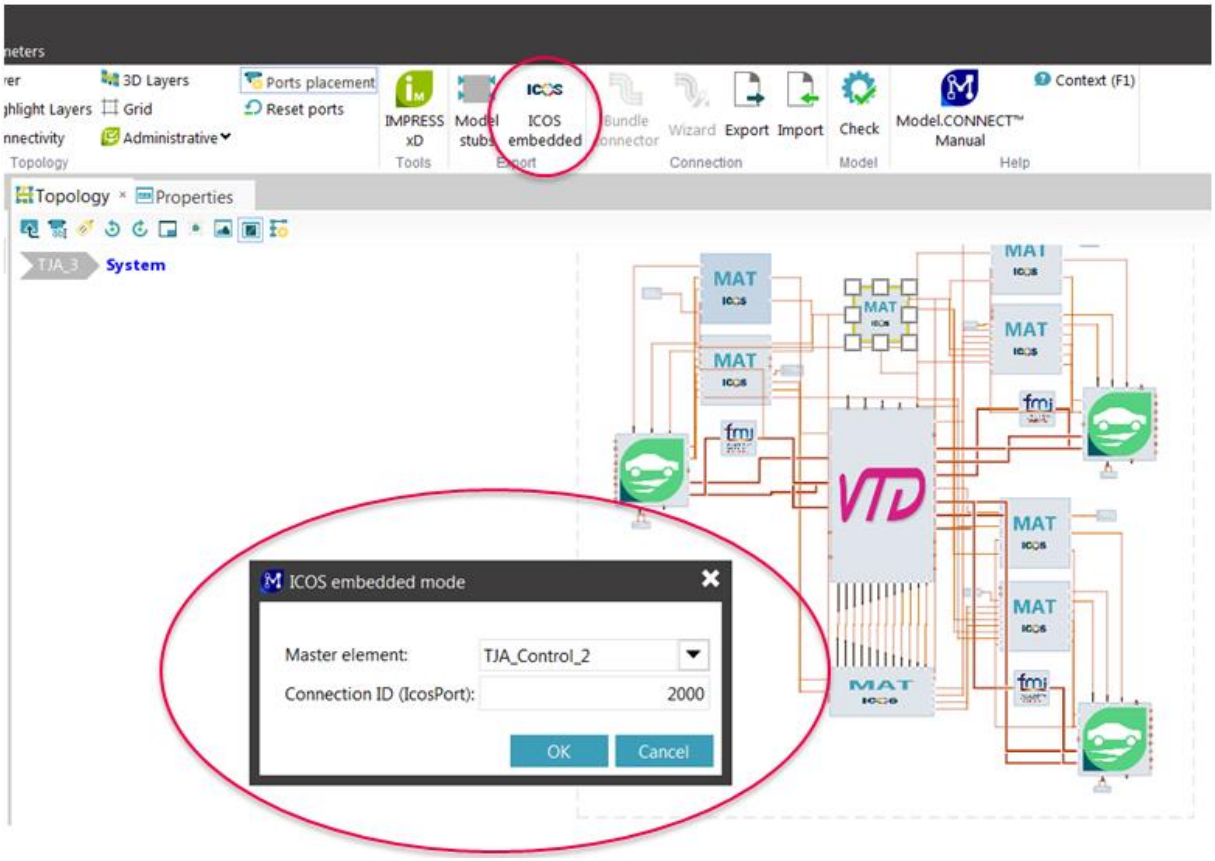

Figure 27: ICOS embedded mode

Before starting the embedded mode the element settings of the MATLAB/Simulink element, that contains the Simulink file from that the simulation should be started, have to be changed. In the MATLAB executable filed write “ICOSPort:2000” (see Figure 28).

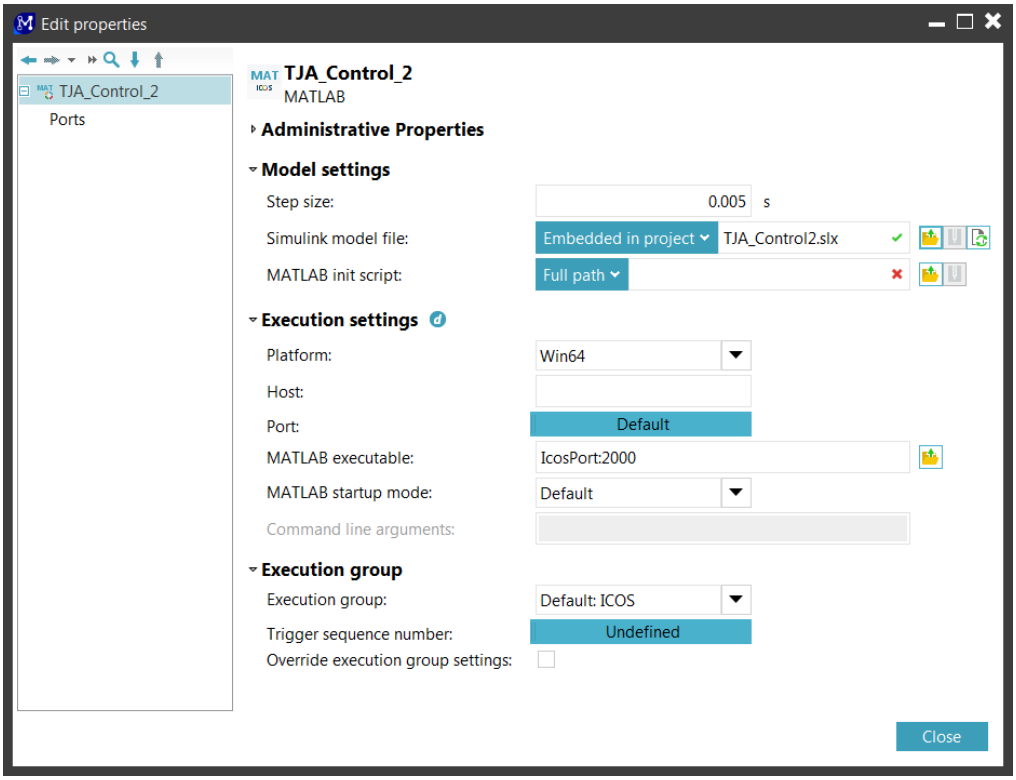

Figure 28: Element properties in ICOS embedded mode

When the ICOS embedded mode is started Model.CONNECT exports a MATLAB file (.m-file) to the folder of the Simulink file. In case the Simulink file is embedded in the Model.CONNECT project this path is

\"Project Name"\"Project Name"_files\modeling

This MATLAB (.m-file) file has to be loaded in the Simulink file at (see Figure 29)

File\Model Properties\Model Properties\Callbacks\InitFcn

The MATLAB file has the name of the element that is exported. It is recommended that this step is done before the embedded mode is started. Before starting the Simulink file and with that the whole simulation be sure that the current MATLAB path is also

\"Project Name"\"Project Name"_files\modeling

If a model has been run in the "ICOS embedded" mode and the next time it should run without the embedded mode, the "InitFcn" in the Simulink file has to be removed again.

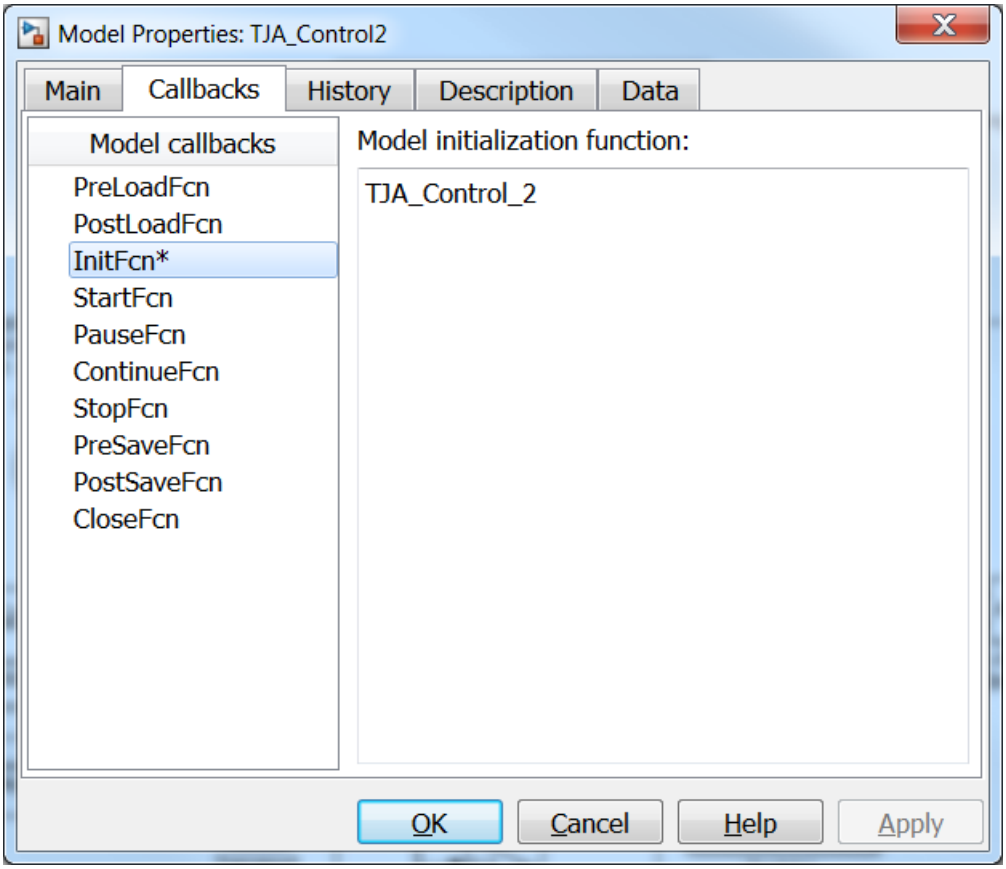

Figure 29: ICOS embedde mode initial function in MATLAB

6.3 Starting in VTD only

It is possible to run a created scenario in VTD only. This is possible with the VTD standalone version. Execute the file

/home/VTD2.0_Standalone/vtdStart.sh

Copy the project from

/home/VTD/Data/Projects

to

/home/VTD2.0_Standalone/Data/Projects

and start the scenarios in the standalone version of VTD. The ego-vehicle will drive with the assigned driver.

6.4 Possible errors and solutions

If and the simulation won't start again after the simulation has been terminated it is recommended to restart the ICOS server. This is done by clicking on Simulations\General Settings\Job Servers. There the current job database of the local job server can be reset and restarted.

Check at Simulations\Job Overview\Status if there are no running simulations before a new job is started. To be sure that there are no running jobs reset the job server as described before.

If a controller produces unexpected output, check the ports of the VSM element for unwanted scaling factors or initial values. Figure 30 shows the property window of the VSM element with the location of the limits, initial value and scaling factor of the selected port.

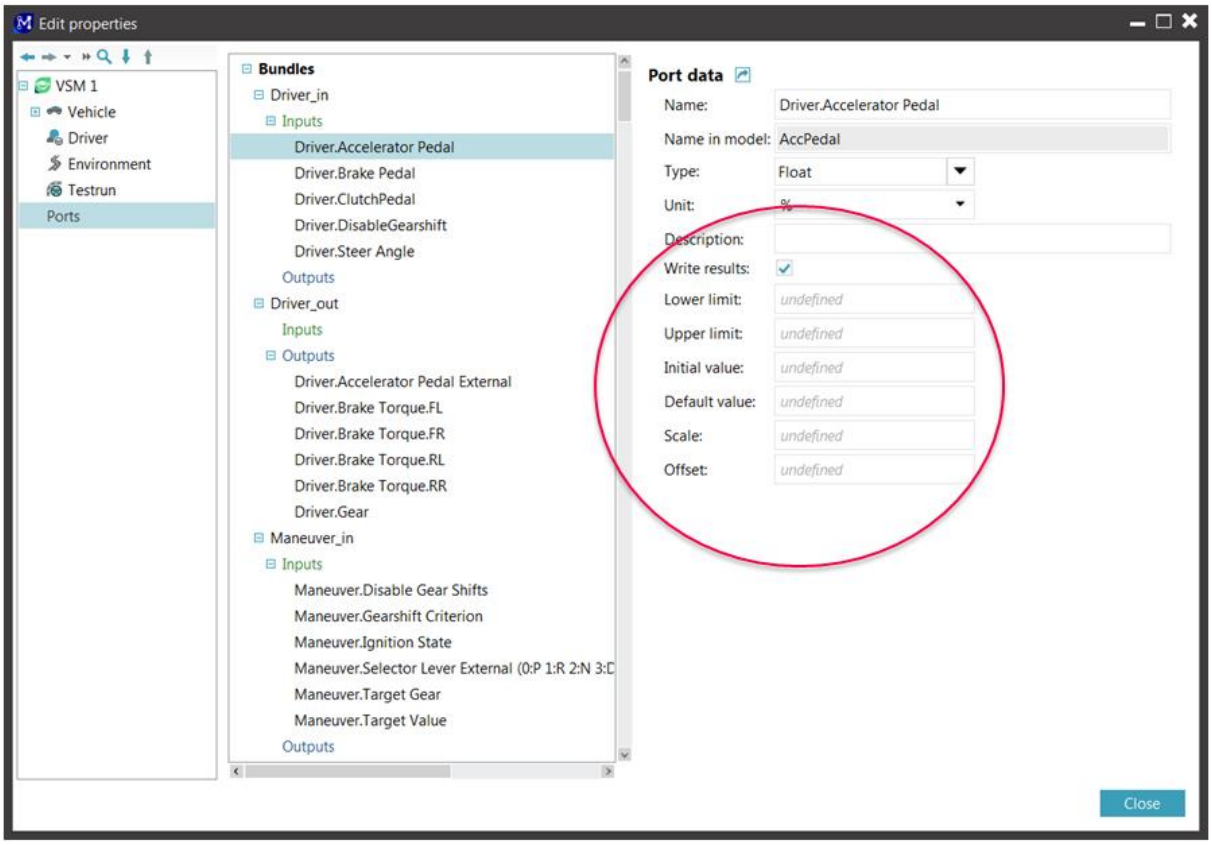

Figure 30: scaling factors and initial values in VSM

After changing a scenario in VTD, check the activated sensors (see chapter 3.3.1). They are not changed automatically with the scenario. So they have to be activated and deactivated manually depending on the scenario and their ego-vehicles. Also check the player that the sensors are assigned to after switching scenarios.

If a model has been copied or "saved as" to another location and the files of the elements haven't been embedded in the project, they have to be reloaded in the element.

6.5 Other messages

If e.g. simulink files are changed and safed after the Model.CONNECT project was saved or closed and the Model.CONNECT project is reloaded, Model.CONNECT will ask you want to overwrite the files with the existing project files or keep them as they have been saved. (Figure 31)

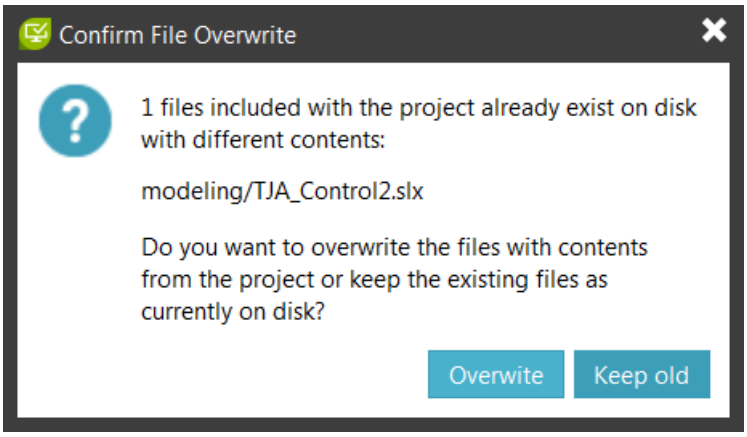

Figure 31: Confirm file overwrite

That means if changes made to the file should be kept, one should choose “Keep old”.

6.6 Result output

The result output of the TJA-model can be found at

D:\1_Model_Connect_Projects\TJA_3\simulation\TJA_3\TJA_3.Case_set_1.Case_1\results

“TJA_3” can be replaced by any other project.

7 Model.CONNECT projects

During the master thesis some Model.CONNECT projects were created. They are all located at

D:\1_Model_Connect_Projects

The following Table 2describes their content and the fitting VTD scenario. The model described in chapter 5 is the Model.CONNECT Folder TJA_3.

Table 2: Model.CONNECT projects

Model.CONNECTFolder	Model.CONNECT Project	Models	VTD Project	Scenario	Comments
Versuch_1	Versuch_1.proj	Sensoren	ModelConnect	Highway10km_ModelConnect	Sensor tests
	Versuch_2.proj	Sensoren	ModelConnect	Highway10km_ModelConnect	All working sensor channels
	VSM_CRUISE_MATLAB_Versuch_3.proj	VTD, Simulink	ModelConnect	Highway10km_ModelConnect	Changed example project with VSM, Cruise und Simulink
Versuch_2	Versuch_2.proj	VTD, Simulink	ModelConnect	Highway10km_ModelConnect	Example with Arno
VTD_VSM	VTD_VSM.proj	Lenken	ModelConnect	Highway10km_ModelConnect	VSM-VTD work together
VTD_VSM_2	VTD_VSM_2.proj	VTD input	ModelConnect	Highway10km_ModelConnect	
VTD_VSM_3	VTD_VSM_3.proj	ACC	ModelConnect	Highway10km_ModelConnect	VSM und Simulink work together, but without VTD
	VTD_VSM_3c.proj	ACC	ModelConnect	Highway10km_ModelConnect	Model without integrator
VTD_VSM_4	VTD_VSM_4.proj	Complete ACC Model	VTD_VSM_4		Complete ACC Model with VTD

Model.CONNECTFolder	Model.CONNECT Project	Models	VTD Project	Scenario	Comments
					(not working)
VTD_VSM_5	VTD_VSM_5.proj	LKA_Model	VTD_VSM_5	VTD_VSM_5	
				VTD_VSM_5-2	Roadmarks 2cm above ground in VTD_VSM_5-2
				VTD_VSM_5-4	Scenario created without macro
	VTD_VSM_5a	Sensoren für LKA	VTD_VSM_5	VTD_VSM_5-4	Support project Poljicak
VTD_VSM_6	VTD_VSM_6.proj	Platooning Model			Platooning model without controllers, control more ego vehicles
VTD_VSM_7	VTD_VSM_7.proj	Manoeuvre_1_2	VTD_VSM_4	Maneuvre_1	Target changes Lane
				Maneuvre_2	Cut into Lane test
		Manoeuvre_1_2_2ACC		Maneuvre_1_2ACC	Target changes Lane with 2 ACC vehicles
				Maneuvre_2_2ACC	Cut into Lane test with 2 ACC vehicles
		Manoeuvre_1_2_3ACC		Maneuvre_1_3ACC	Target changes Lane with 3 ACC vehicles
				Maneuvre_2_3ACC	Cut into Lane test with 3 ACC vehicles
				Bzw. alle Scenarios mit 3ACC	
		Manoeuvre_1_2_LDACC		Possvible with all 3 ACC scenarios	LD stands for Lane dependable ACC
		Manouvre_3_4		CurveCapabilityTest	Curve Capability Test (longitudinal speed change)
				CurveCapabilityTest_lateral	Curve Capability Test (lateral speed change)

Model.CONNECTFolder	Model.CONNECT Project	Models	VTD Project	Scenario	Comments
		Z_Manual		Manual_Driving	Ego-vehicle is controlled with sliders in Matlab/Simulink
Support_Project	Support_Project.proj		ModelConnect	Highway10km_ModelConnect	Model for the support mails
	Support_Project2.proj		ModelConnect	Highway10km_ModelConnect	Model with Signal creator input
VSM_Vires	VSM_Vires.proj	VSM-Vires Coordinate transformation incl. controller	ModelConnect	Highway10km_ModelConnect_Ego	With first ACC controller
AVL_VTD_VSM	AVL_VTD_VSM.proj	VSM-Vires Coordnate transformation (only)	ModelConnect	Highway10km_ModelConnect_Ego	
TJA	TJA.proj	TJA	VTD_VSM_4	TJA	3ACC, first Steering Model
				UC4_Scenario2	
				UC4_Sceanrio3	
		TJA_1Ego		TJA_1Ego	
		TJA_2_1Ego		TJA_1Ego	Steering model 2
		TJA_variableGap		UC4_Scenario2	ACC-gap depends von velocity (Platooning)
				UC4_Sceanrio3	
		TJA_3_1_Ego		TJA_1Ego	
		ACC_LKA_1Ego		CIL_Novak / TCL Novak	Testing scenario for chapter 3.2

Model.CONNECTFolder	Model.CONNECT Project	Models	VTD Project	Scenario	Comments
		ACC_LKA_1Ego_LDACC		CIL_Novak / TCL Novak	Testing scenario for chapter 3.2
		TJA_3_1Ego_Parallel		ParSeq	Testing scenario for chapter 3.1
		TJA_3_1Ego_Sequentiell		ParSeq	Testing scenario for chapter l 3.1
TJA_2	TJA2.proj	TJA_3	VTD_VSM_4	Alle mit 3 Egos	Trial model
		TJA_2_PL		PlatooningQuerdynamik	Testing scenario for chapter 3.3.2
		TJA_2_vF		PlatooningQuerdynamik	Testing scenario for chapter 3.3.2
		TJA_2_PL_mitPlatooning		PlatooningLaengsdynamik	Testing scenario for chapter 3.3.1
		TJA_2_PL_ohnePlatooning		PlatooningLaengsdynamik	Testing scenario for chapter 3.3.1
TJA_3	TJA3.proj	TJA_3	VTD_VSM_4	UC4_Scenario	Testing scenario for chapter 3.4
TJA_4	TJA4.proj	TJA_4	VTD_VSM_4	UC4_Scenario_4Ego	Testing scenario for chapter 3.4

List of Figures

List of Tables

A. Appendix

a. Location of files

Installation files:

M:\FTG-P\3_RnD\DVS\Project_Funded\Current\2016_f001f_PA_Enable_S3\Work_01\14_UC4\02_Software_Installation

User manuals:

M:\FTG-P\3_RnD\DVS\Project_Funded\Current\2016_f001f_PA_Enable_S3\Work_01\14_UC4\05_Software_manuals

Model.CONNECT projects:

D:\1_Model_Connect_Projects