HAVEit

Highly automated vehicles for intelligent transport

7th Framework programme
ICT-2007.6.1
ICT for intelligent vehicles and mobility services
Grant agreement no.: 212154

The future of driving.

Deliverable D41.3
Joint System validation in vehicle (2^{nd} version)

Version number	Version 0.8
Dissemination level	CO, PU
Lead contractor	Continental Automotive GmbH
Due date	13.08.2010
Date of preparation	13.08.2010

Authors

Name	Company
Gerald Temme	DLR
Tobias Hesse	DLR
Christian Löper	DLR
Henning Mosebach	DLR
Jan Schomerus	DLR
Waldemar Schrinner	DLR
Frank Flemisch	DLR
Salim Hima	LCPC
Benoit Vanholme	LCPC
George Thomaidis	ICCS
Paulo Resende	INRIA
Armin Kaussner	WIVW

Project Manager

Alfred Hoess
Continental Automotive GmbH
Siemensstrasse 12
93055 Regensburg, Germany
Phone +49 941 790 5786
Fax +49 941 790 99 5786
E-mail Alfred.Hoess-EXT@continental-corporation.com

Holger Zeng
Continental Automotive GmbH
Siemensstrasse 12
93055 Regensburg, Germany
Phone +49 941 790 92330
Fax +49 941 790 99 92330
E-mail Holger.Zeng@continental-corporation.com

Project Co-ordinator

Reiner Hoeger
Continental Automotive GmbH
Siemensstrasse 12
93055 Regensburg, Germany
Phone +49 941 790 3673
Fax +49 941 790 99 3673
E-mail Reiner.Hoeger@continental-corporation.com

Revision and History Chart

Version	Date	Reason
0.1	08.06.2010	First stub
0.2	12.07.2010	Input from partners added
0.3	14.07.2010	Comments of the first internal review included
0.4	22.07.2010	Added figures from partners in Chapter 2
0.5	26.07.2010	Integration of DLR internal review comments
0.6	19.07.2010	D41.3 ready for HAVEit internal review
0.7	09.08.2010	All comments from the reviewer are merged into one document
0.8	18.08.2010	Final version completed

Table of Contents

Table of Figures

Executive Summary

Deliverable. D41.3 “Joint System validation in vehicle” reports the tests and validation of the HAVEit Joint System demonstrator, a trailblazer prototype integrated in the DLR test vehicle FASCar, about one year before the end of the HAVEit project.

HAVEit investigates a potential future of driving beyond manual or assisted driving: Highly automated driving. In contrast to fully automated driving, where the human is only a passenger, in highly automated driving a high percentage of the driving can be performed by an automation (called the co-system), but the human is still a driver who is in control of the highly automated vehicle.

In HAVEit, an overarching concept, architecture and several prototypes are built up in an iterative approach of concept definition, test and refinement in simulators and test vehicles. In the first phase of the project, an initial set of use cases and design interaction between driver and co-system, and of the behaviour of the co-system was developed in an integrated effort between industry and research institutions, using a theatre technique, where the co-system was emulated by a human. Initial use cases and design were documented e.g. in Del. D33.1 and D33.2.

Based on this description of behaviour and interactions, prototypes are built up and tested: As a trail blazer for the vertical demonstrators closer to serial production, a horizontal “Joint System” prototype is under development by an interdisciplinary team of several European research institutes. The term “Joint System” stands for the combination of a human and a technical co-system into an integrated whole. With this “Joint System” trail blazer the basic principles of highly automated driving are developed, refined and tested in simulators first, documented e.g. in Del. D33.3. In addition to the simulator testing, the co-system was integrated and tested in a couple of integration phases into the DLR experimental vehicle FASCar. As a benchmark one year before the end of the project, and accompanying the ongoing transfer of concepts and algorithms between horizontal and vertical subprojects, the integration and test of the Joint System demonstrator was documented in this deliverable D41.3.

Chapter 1 describes the concept of the Joint System, including the software development, the hardware migration of the Joint System demonstrator, and the configuration of the demonstrator vehicle. The flow of information and action in the co-system starts with sensor data fusion, proceeds to the planning of the manoeuvres and trajectories, the merging with the driver’s action in a mode selection and arbitration unit and the human-machine interface, and finishes with the generation of the command that is applied to the actuators of the vehicle.

Chapter 2 describes the tests and results organized regarding the relevant HAVEit use cases. The highest priority was assigned to the automation level Highly Automated, because of the highest functional complexity of this automation level. Chapter 3 describes the validation and status report of the individual system components.

Chapter 4 gives a conclusion and outlook: The validation showed that, not surprising, one year before the end of the project; there are still a couple of things to improve. The fundamental system behaviour however was already successfully tested in a real vehicle and with real sensors. The overall concept, architecture and process hold up to its expectations.

Overall, the subproject “Joint System” and the HAVEit project are on track. We will continue the iterative approach of development, test and refinement, condensing the initial ideas and concepts into test results and hands-on experience in increasingly realistic environments. Highly automated driving is no longer a theoretical vision, but can be experienced now beyond simulators also in real vehicles.

1 Concept of Joint System

In 2010, predominantly two lines of research and development exist in the domain of ground vehicle automation: Either the automation is driving the vehicle fully automated without a human driver, as demonstrated e.g. in the DARPA Grand Challenge and Urban Challenge; or assistance systems warn or support the driver, while the driver is performing the driving task. These developments can be seen related as different levels on a simplified assistance and automation scale, reaching from purely manual to fully automated control, [11]. Some assistance systems like Adaptive Cruise control (ACC) which automate one of two control dimensions could be located in the middle of this spectrum and can be called semi-automated.

HAVEit addresses especially the region of highly automated driving, where the vehicle has the technical capabilities that it could drive fully automated, but this capability is used in a way that the driver is always meaningfully involved in the driving task, [9]. HAVEit also addresses the transitions between different levels of assistance and automation. Ideally, this is done in a way that the driver can be relieved in overload and underload conditions as shown in

Figure 1.

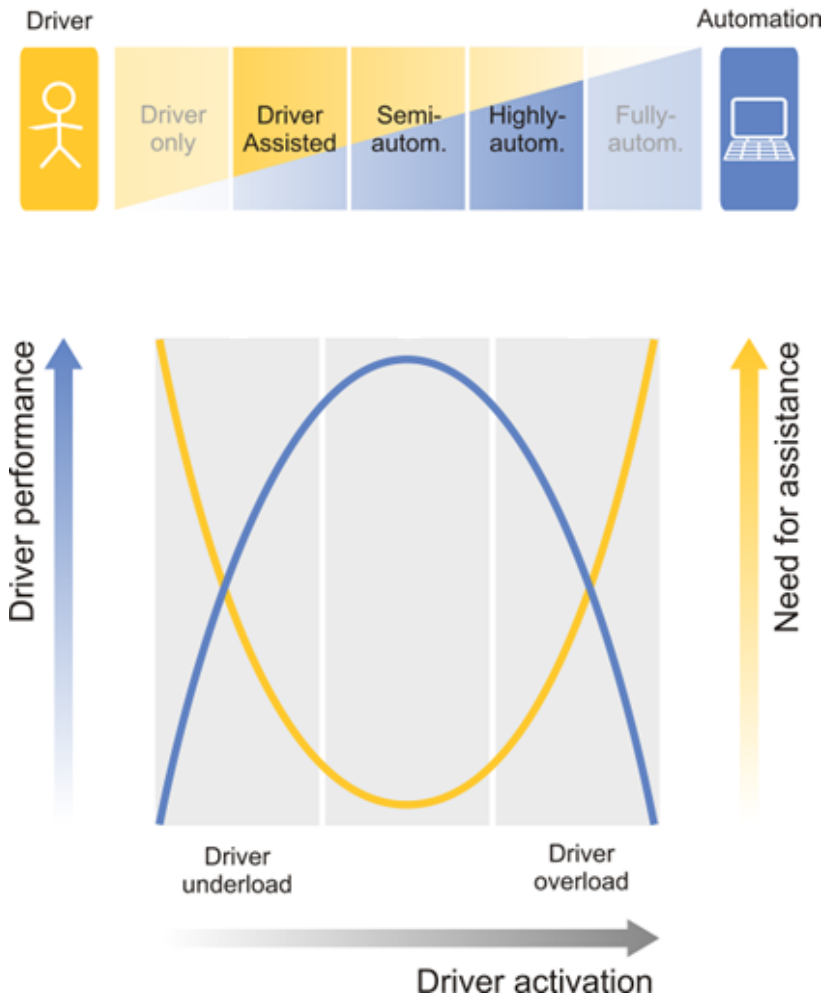


Figure 1: HAVEit assistance & automation scale and dynamic task repartition in the Joint System

This task repartition between driver and automation or co-system can be dynamically influenced by both the driver and the co-system, and is especially investigated in the HAVEit subproject "Joint System" [10]. In a joint effort of several European research institutes, the fundamental HMI principles for highly automated driving are worked out together with the partners from the vertical subprojects. After a sufficient initial understanding of the behaviour and HMI of such highly automated vehicles was generated in human centred investigations in the theatre system, [17], these principles are implemented in a Joint System Demonstrator, tested in simulators, [6] and the DLR test vehicle FASCar (described further down), and then applied to the other HAVEit demonstrator vehicles.

For the design and development of the HAVEit Joint System, several different perspectives had to be brought into a dynamic balance [10]: A more critical perspective of questioning and testing was brought together with a constructive perspective in an iterative approach. The technical perspective, where new technical systems (e.g. driver state monitoring camera, new vehicle sensors or manoeuvre-based automation) enable a more sophisticated way of dynamic task repartition, was brought together with the human centred perspective, where the design has to enable the driver to understand and to handle the highly automated system in an intuitive way.

The technical subsystems were structured as shown in the Joint System component overview in Figure 2.

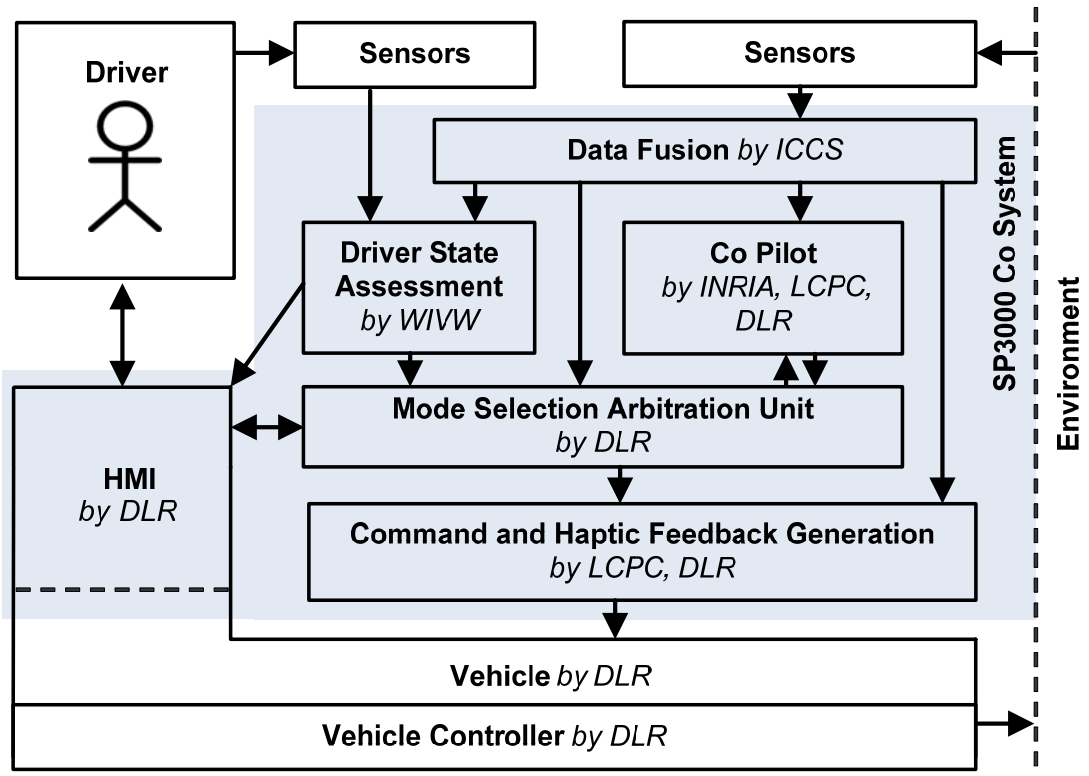


Figure 2: HAVEit SP3000 Joint System component overview

As illustrated, the Joint System is comprised of the human driver and a technical system. The latter consists of the sensors to gather information about the environment and the state of the driver, the base vehicle with some subordinate controllers, and the Co-System. The Co-System (gray marked area) represents the software part of the technical side of the Joint System. It contains different components

- to process the gathered sensor information (Data Fusion, Driver State Assessment),
- to plan manoeuvres and trajectories (Co-Pilot),
- to generate an appropriate feedback to the driver (HMI),
- to join driver and automation wishes and to switch between different levels of automation (Mode Selection and Arbitration Unit) and finally
- to calculate the necessary commands to the base vehicle (Command and Haptic Feedback Generation).

These components and the respective responsible SP3000 partners for the FASCar joint system implementation are also indicated in Figure 2.

Section 1.1 details the functionality and main in- and outputs of these components, as they are specified to work by the end of the HAVEit project in June 2011, and briefly reflects on the development process to implement these components.

The Joint System is currently being integrated into a test vehicle to create a Joint System demonstrator as a prototype to further investigate and demonstrate the basic principles of highly automated driving. Subsequently, these principles will then be gradually applied to vehicles closer to serial production.

This deliverable focuses on the test and validation of the Joint System in the FASCar rapid prototyping test vehicle as intermediate report. The final demonstration of all HAVEit demonstrators is scheduled for June 21-22, 2011. The configuration of the final demonstrator vehicle is detailed in section 1.2.

Chapter 2 presents the results of the validation of the Joint System in four different HAVEit use cases to show how much of the envisioned final functionality is already working as specified. A more detailed discussion of the current status of each Co-System component can be found in Chapter 3. The conclusion and an outlook are given in Chapter 4.

1.1 Software Development and Hardware Migration

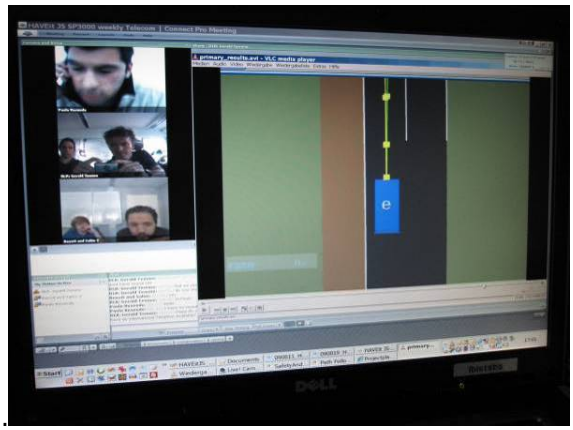
While the hardware of the demonstrator vehicle is detailed in Section 1.2, this section details the functionality and the main in- and outputs of the different software components of the Co-System, illustrated in Figure 2. The development of the Co-System is a concerted effort by the following companies and institutes from several European countries:

- DLR: German Aerospace Centre; Institute of Transportation Systems
- LCPC: Laboratoire Central des Ponts et Chaussées
- INRIA: Institut National de Recherche en Informatique et Automatique
- ICCS: Institute of Communications and Computer Systems
- WIVW: Würzburg Institute for Traffic Sciences GmbH

The different components are described in the following subsections 1.1.1-1.1.5

In such a distributed development approach, a structured and well coordinated software development process is the key element for a smooth integration and validation. One of the first necessities for a well coordinated software development process is an accepted component interface specification described in deliverable D12.1 [1]. Weekly conference calls (see Figure 3, left) were established to discuss and to document technical issues and the progress of the complete Joint System.

The software development process works as follows: Partners develop their components in a PC based framework. It allows a pre-integration of the decentralized developed Joint System components, which is managed with the help of a repository management tool [14]. The PC based framework also allows the testing of the complete Joint System in a simulation environment or by data replays recorded during FASCar test drives. Additionally, it is possible to isolate components and to test them with defined inputs independent from other Joint System components. With the framework it is also possible to test the Joint System in the demonstrator vehicle FASCar by several PCs. The final step is to migrate components from the PCs into the CSC-ECUs at the FASCar. To report and to document detected errors and action items during the development of the Joint System the bug tracking tool Mantis (Figure 3, right) is used by all SP3000 Partners.



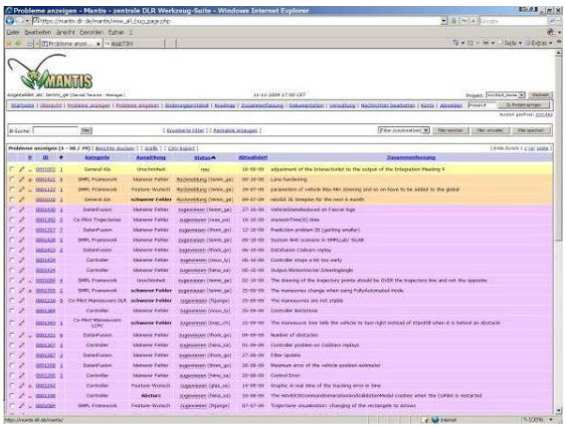


Figure 3: Left: Adobe® Acrobat® Connect™ Pro Meeting used for the weekly conference calls Right: Bug tracking tool Mantis

The development of the Joint System components is preformed by several development and integration phases until the software prototype has reached a proper level of quality according to the spiral model of software development [8]. A more detailed overview of the development and test process and philosophy behind is given in deliverable D33.3, [6].

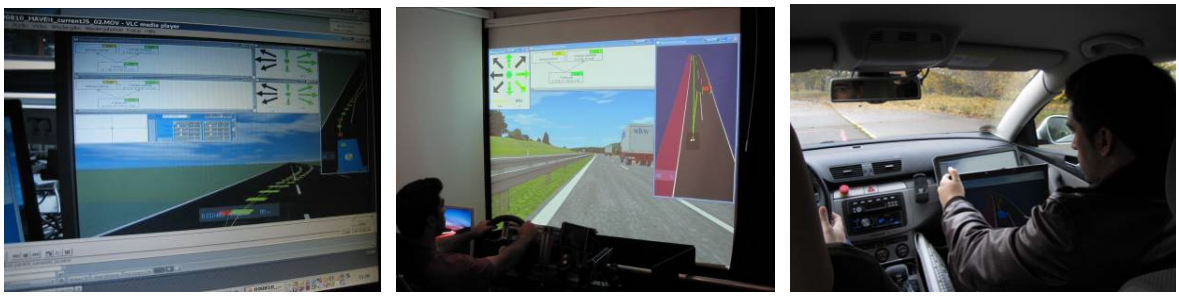


Figure 4: Left: Joint System running on a PC in SMPL++ simulation environment Middle: Joint System running on PCs in SILAB simulation environment Right: Joint System running on several automotive PCs in the FASCar

1.1.1 Data Fusion

As mentioned before and shown in Figure 2, different sensors provide an input to the Co-system. More details with regards to the sensors can be found in Section 1.2, where the hardware of the demonstrator vehicle FASCar is described.

The input from the sensors is processed in the Data Fusion component. Data delivered by this component describe the state of the ego vehicle relative to the environment. Additionally, important entities of the road environment are described. The Perception Model, which emanates from the Data Fusion component, provides an estimation for the lanes and obstacles in the region around the ego vehicle. Additionally, the relations among the ego vehicle, the lanes and the neighbouring obstacles are also described. Some examples of this kind of relational data are the ego vehicle's and obstacles' lane assignment, the heading of other vehicles relative to the ego vehicle etc.

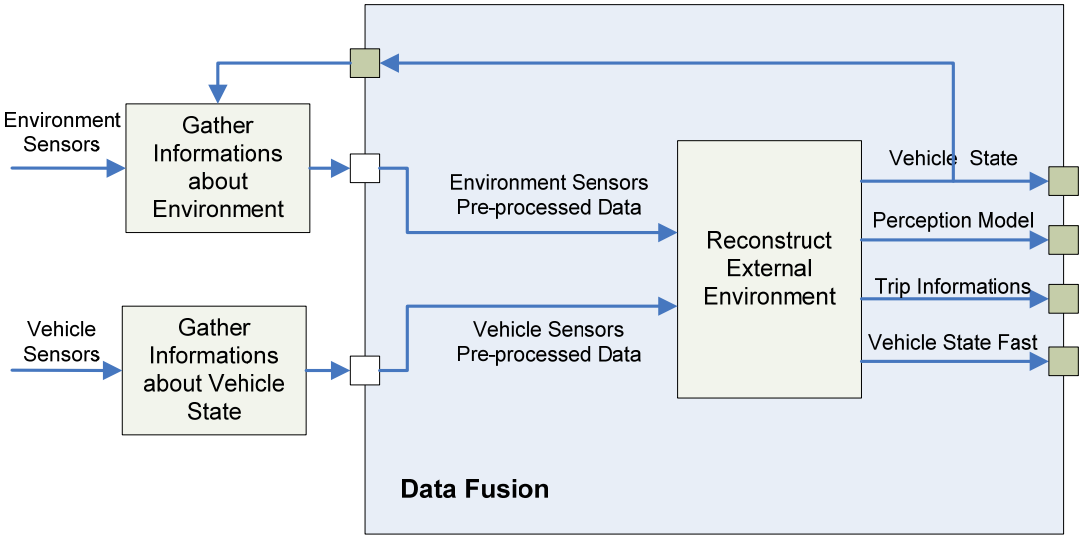


Figure 5: Data Fusion component

This data fusion function uses following inputs (see Figure 5):

- Environment Sensors Pre-processed Data: This input refers to the pre-processed data of all the HAVEit environment sensors such as vision system, laser scanners, digital maps, GPS receiver etc.
- Vehicle Sensors Pre-processed Data: This input contains the vehicle sensors' measured data such as velocity, yaw rate, steering wheel angle, longitudinal and lateral acceleration, vehicle dimensions etc.

It provides following outputs (see Figure 5):

- Vehicle State: To be able to estimate the state of the vehicle and to predict its future movement, all the information about the current vehicle state is necessary. Vehicle State contains information about the current position and the kinematics of the vehicle. The Data Fusion component collects the data needed for the HAVEit functions from the vehicle CAN buses. These data are being filtered and translated into the generic data format and then sent to the generic HAVEit bus.
- Perception Model: The perception model consists of state estimations of objects that lie in the area of interest of the ego vehicle. Each object type (for example vehicle, lane or road) is described by a state vector. The state vector contains physical quantities or variables with continuous or discrete values. It contains information about obstacles, lanes and intersections. The obstacle list describes up to 8 objects and up to 3 lanes.
- Trip Information: This output contains information about the trip type and duration.
- Vehicle State Fast: This is a subset of Vehicle State consisting of vehicle speed, yaw rate and system timestamp. The "Command Generation and Validation" component uses this information to estimate the position of the vehicle, between two trajectory deliveries of the "define the trajectories" component.

The data fusion component receives data from the sensors that monitor the environment around the vehicle and the ego vehicle itself. Sensors are considered as heterogeneous sources of information that provide variables which describe the objects that surround the vehicle. Each of these sources of information transmits its data in its own spatial and temporal frame with a defined refresh rate. The Perception Model gathers all these data and executes several tasks in order to provide estimates that are synchronized and aligned in the spatial state. In general, the output of the Data Fusion is divided in three main categories: ego vehicle state estimation, lane estimation and object estimation. Ego vehicle state estimation provides information about the vehicle's kinematic state (velocity, acceleration, etc.) as well as information about vehicle's position relative to the environment (offset relative to the lane). Object estimation uses the output of the laser scanners and performs the object tracking providing estimates of the state (position, velocity) of the objects that are detected. Lane estimation uses multiple sources of information (lane detection camera, digital map) in order to estimate the lane geometry ahead of the ego vehicle.

1.1.2 Co-Pilot

The Co-Pilot component supports the driver by identifying the current driving situation and providing a recommendation of the manoeuvre to be executed by the driver and a trajectory to be tracked by the vehicle controllers in a highly automated mode. This trajectory is determined taking into account the driving strategy (manoeuvre), the current vehicle state, the perception model, the driver inputs and other vehicle related constraints.

In order to achieve a strong cooperation with the driver, irrespective of the automation level, the Co-Pilot process is achieved using two main functionalities:

- The definition of a driving strategy, provided by fast and simple algorithms, evaluates the possibility of performing several predefined manoeuvres.
- The definition of a trajectory, using the previously selected driving strategy to limit the trajectory planner search space, thus reducing its calculation time.

The Co-Pilot component receives the following inputs:

- Current vehicle state
- Perception model: a local map (given by environment sensors) with the objects and lanes in the environment that can be extended by the use of communication
- Arbitrated driver inputs: driver preferred settings
- Driver primary task command: the commands that the driver activated or that he is currently performing in the vehicle

The Co-Pilot component provides the following outputs:

- Set of trajectories with descriptions: contains all generated trajectories and the available manoeuvres.
- Limit of health horizon: remaining time or distance until the end of the available trajectories.

Manoeuvres

The “driving strategy” or manoeuvre planning component is the first component of the Co-Pilot. A manoeuvre describes the behaviour of the vehicle on a linguistic abstraction level. By this the behaviour is defined completely in the longitudinal and lateral direction. The “manoeuvre planning” component determines which manoeuvre should be executed next and which other manoeuvres are also feasible in the considered situation.

For reliability reasons the outcome of the manoeuvre planning component consists of a fusion of the results from three manoeuvre planning algorithms which work in parallel. Two algorithms build up a manoeuvre grid and one algorithm generates a manoeuvre tree. A forth sub-component fuses the results of the tree and the grid algorithms into one single output. This output is available in both the tree as the grid representation (see Figure 6).

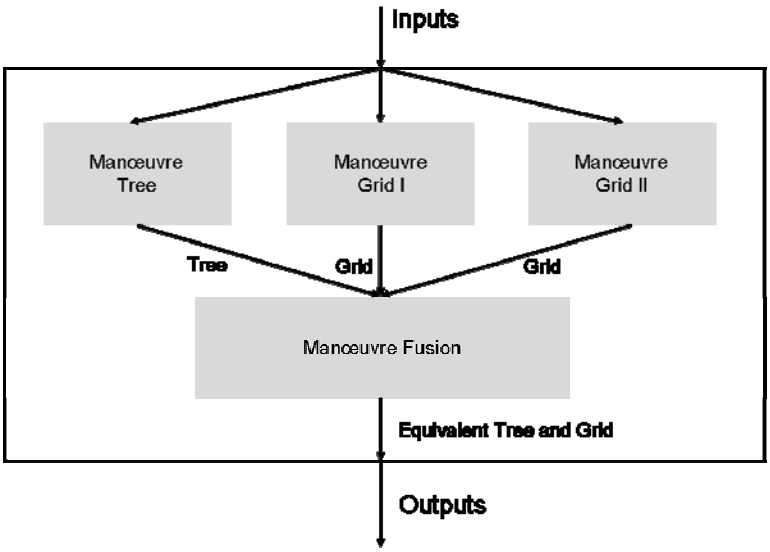


Figure 6 Manoeuvre planning component in detail

The manoeuvre tree (see Figure 7) offers an integrated representation of the current action of the vehicle and possible future actions regarding to the current situation, starting with the current manoeuvre as the root of the tree. Feasible manoeuvres which can possibly follow the current manoeuvre are located as leaves in the tree. A quality rating called valential is assigned to all feasible manoeuvres to show the preferences of the automation. The calculation of the valential is based on fuzzy logic.

Further information about the approach can be found in [13].

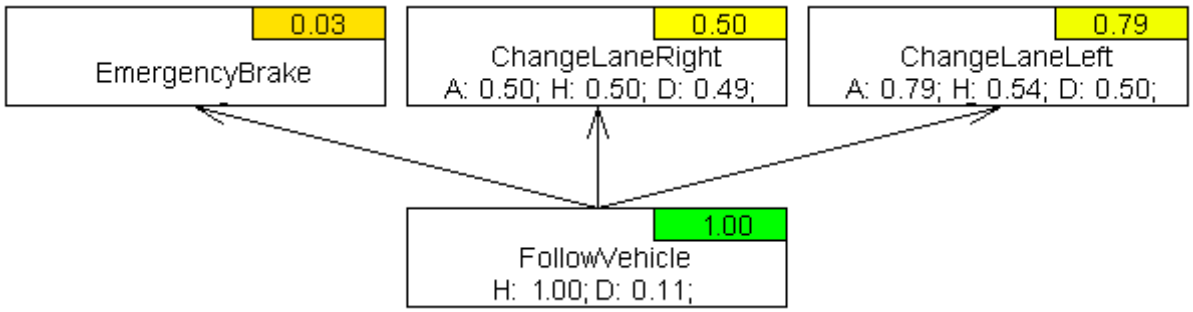


Figure 7 Example manoeuvre tree

The manoeuvre grid builds a solution space as the combination of three longitudinal actions and three lateral actions, Figure 8. On the *longitudinal* side the vehicle can be decelerated, accelerated or held at current speed. In the *lateral* direction, the vehicle can change lane to the right, to the left, or can stay in the same lane. To these nine manoeuvres, a *minimum risk state* manoeuvre (MRM) is added, which corresponds to stopping on the right most lane, and an *emergency* manoeuvre, corresponding to maximum braking till standstill, as shown in Figure 8.

A first manoeuvre grid algorithm calculates the risk associated with each of the eleven manoeuvres. It allows reducing the calculation time of the trajectory component by discarding highly risky zones of the solution space in an early stage of the calculations. A second manoeuvre grid algorithm evaluates other performance indicators on the remaining manoeuvres, such as the speed, comfort and fuel consumption performance. The total performance of each manoeuvre is the weighted sum of the different performance indicators, with the weights allowing tuning the character of the Co-Pilot [12][17][19].

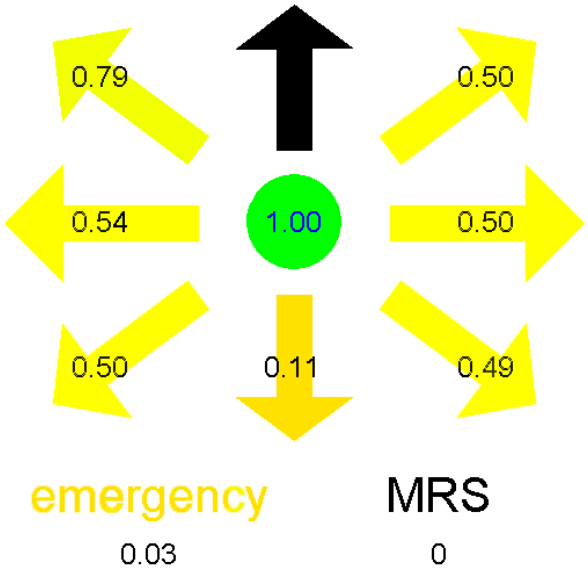


Figure 8: The manoeuvre grid

Trajectories

The trajectory sub-component uses the output of the manoeuvre sub-component to generate a detailed spatio-temporal description of the optimal trajectory. A total of four trajectories (one per lane plus a minimum risk trajectory) are generated, see Figure 9. The trajectory that will be used by the controller corresponds to the one with the highest valential given by the manoeuvre algorithm. The trajectories will not decide to change the manoeuvre performed by the controller even if it detects a collision since that task is supposed to be assigned to the manoeuvre component.

In practice, once a coarse plan has been defined, a specific motion plan is assigned to the vehicle. This motion plan defines the state of the vehicle for the future time instants. This sequence of desired states in time is the so-called trajectory.

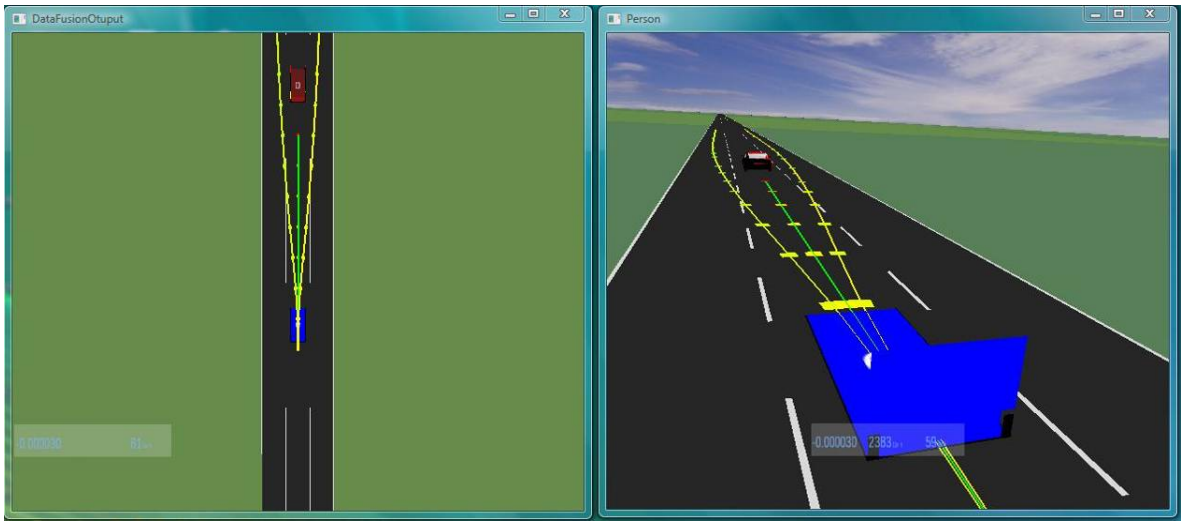


Figure 9: Trajectory planning

1.1.3 Driver State Assessment (DSA)

The DSA component combines two information sources: direct driver monitoring and indirect driver monitoring. Direct information about the driver is derived from the Driver monitoring System (DMS), which observes driver's eye movements and gaze direction by a near-infrared-camera. Indirect driver monitoring means the observation of driver's activity (direct inputs on the steering wheel, pedals, in-vehicle information systems etc.) and performance measures (e.g. lane keeping quality). A set of signals is used to derive relevant parameters that correlate with increasing drowsiness or distraction.

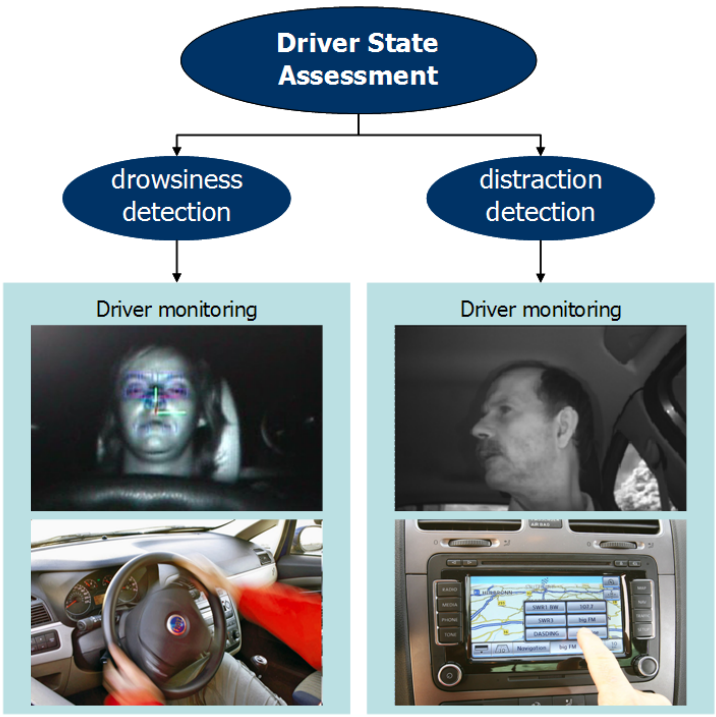


Figure 10: Driver state assessment

The main principles of the DSA algorithm can be summarized as the following:

- Long-term drowsiness vs. short-term distraction are assessed separately
- Direct (output of the DMS) and indirect measures (internally calculated parameters from driving performance and driver's activity) are fused
- Driver's state is assessed on multiple levels with a differentiated parameter set for each level
- In a calibration phase the "normal" driving style of each individual driver is assessed in order to adapt the algorithms (especially for the calculation of some indirect parameters)
- A manoeuvre detection and classification (e.g. lane changes, sharp curves) is included in order to define the appropriateness of calculated parameters (especially indirect ones from driving behaviour) and to decide on their inclusion in the output

The DSA utilizes the following inputs:

- The current automation level (hands-off vs. hands-on) is considered for the decision on the inclusion of parameters in the output
- Signal quality (e.g. detection of lane markings) to determine confidence of the final output
- Inputs for the classification of state "distracted":
  - Information about the driver looking on/off the road (output of the DMS)
  - Information about driver's use of onboard systems, buttons pressed on the steering wheel etc.

- Inputs for the classification of the multiple states of “drowsiness”:
  - Direct: output of DMS diagnostic
  - Indirect: critical lane keeping behaviour, critical steering wheel activity, critical distances to a lead vehicle, critical duration of unintended hands-off driving, inadequate reaction times to take-over requests, etc.

The DSA provides the following outputs:

- Confidence of the final output, determined by signal quality (e.g. detection of lane markings)
- 2 distraction states: distracted vs. not distracted
- 5 drowsiness states: undefined, awake, slightly drowsy, drowsy and sleepy
- State “unresponsive”

A detailed description of the algorithm can be found in deliverable D32.2 [5].

1.1.4 Mode Selection and Arbitration Unit (MSU)

The Mode Selection and Arbitration Unit (MSU) determines the influence of the automation on the vehicle actuators. The MSU enables the driver and the automation to arbitrate an appropriate assistance and automation level (see Figure 11) based on the inputs from the Driver, the Data Fusion, the Co-Pilot and the Driver State Assessment.

For this selection the MSU needs the following inputs:

- The *vehicle state* which contains all vehicle relevant information which are dynamic like the current speed
- *Relevant targets and key environment information’s* which is a subset of the full perception model of the Data Fusion and describes the main obstacles and the current lane
- *Driver’s primary task command* which contains the input of the driver on the pedals, on the steering wheel as well as information about the use of the indicator.
- *Drivers automation level request* includes status information from the automation buttons or stalk that the driver uses for switching between the HAVEit automaton levels
- *Drivers drowsiness or distraction level* indicates the status of the driver
- The *set of trajectories with description* describes the current status of the Co-Pilot by a manoeuvre representation.

Beside the selection of the optimal automation level the MSU manages the transitions (see Figure 12) between the automation levels and controls the visual, acoustic and haptic Human-Machine-Interaction (HMI) (see Figure 13) that is used to inform the driver about the current automation level and transitions.

The MSU delivers the following outputs for the other Joint System software components:

- The *current and requested automation level* which describes the states of the MSU state machine like the current automation level, the current transition or the current transition type (immediate, impossible, with or without escalation…)

- The *arbitrated driver inputs* contain variables like the chosen set speed by the driver for the automation levels Semi Automated or Highly Automated as well as the recommendation by the MSU to initialize a lane change or Minimum Risk Manoeuvre at the Co-Pilot.
- In the *HMI Interface output* all relevant information by which the MSU controls the visual, acoustic and haptic HMI are included.

The final version of the MSU will run on a fail-safe CSC (chassis and safety controller) embedded ECU plarform (see Figure 26 in the demonstrator vehicle and on a PC in the simulator). To run on a CSC under Autosar the MSU algorithms need to be checked by the MISRA C rules and meet Autosar standards. The MSU will communicate with the Joint System over CAN.

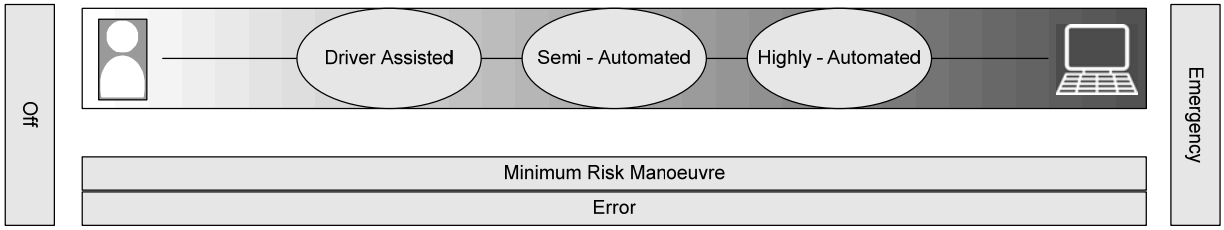


Figure 11: HAVEit assistance and automation spectrum / scale

Figure 11 shows the available automation levels within the SP3000. On the automation spectrum three different HAVEit automation levels are depicted [10]:

- Driver Assisted (DA) is a level in which the driver drives by his own. He is assisted by the Co-System automation by warnings that are given by visual, acoustic or haptic means for example in case of lane departure.
- Semi Automated (SA) is a level in which the automation controls the longitudinal dimension of the vehicle motion. This functionality is comparable to an ACC system that is enriched with speed limit recognition and automated speed adaptation. The driver can set the speed and the time gap to a vehicle in front. Regarding lateral control of the vehicle he receives the same kind of assistance as in the level Driver Assisted.
- Highly Automated (HA) is a level in which mainly the automation controls the longitudinal and lateral movements of the vehicle. Compared to fully automated control the driver is still requested to be in the loop and to take over control if required, e.g. in case of reaching the system limits. As in level Semi Automated the driver can set the speed and the time gap to a vehicle in front, but doesn't need to steer manually. Additionally he can activate a highly automated lane change.

There are additional automation levels which are only used in case the above described levels can no longer be continued:

- *Minimum Risk Manoeuvre (MRM)* and *Emergency* are levels that are needed to bring the vehicle to a safe state in case the driver cannot react fast enough or does not react at all. The Co-System takes over the vehicle control and stops the vehicle in adequate way.
- The *Error* state is a state in which the co-system breaks down completely so that none of the automation levels is further available and the driver has to take over vehicle control.

A potential reaction to the Error state could be that a watchdog unit alerts the driver by a warning and initiates an immediate take-over.

- The *Off* state is a state in which the driver switched off the complete Co-System. The MSU is no longer active, there are no automation levels available and the driver has complete control over the vehicle.

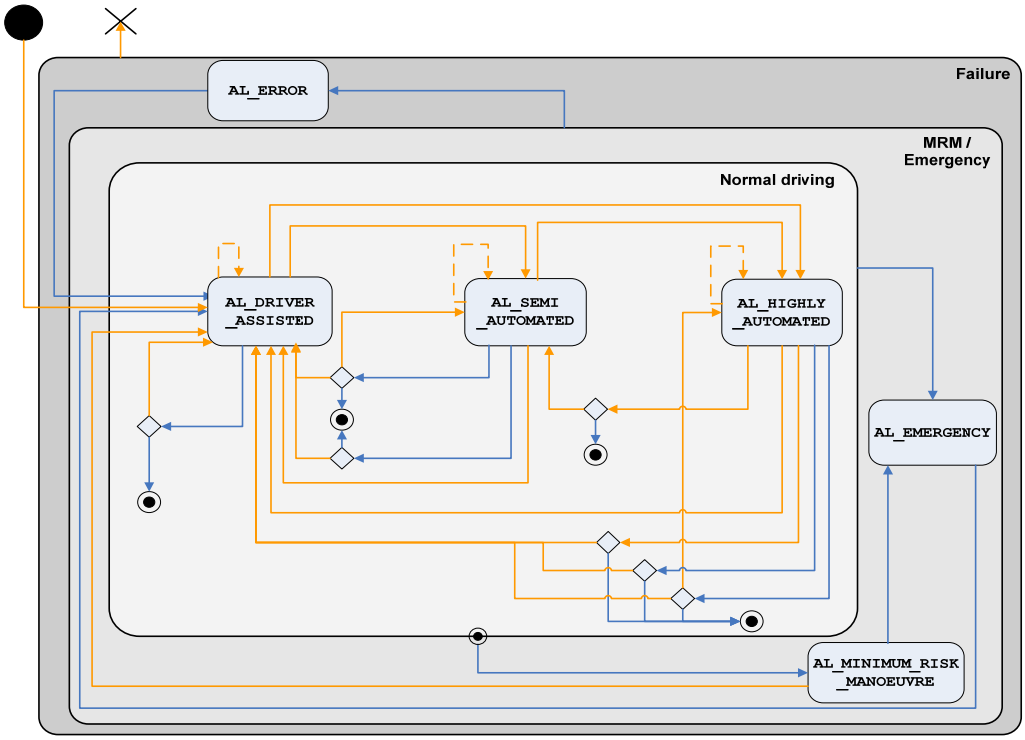


Figure 12: Automation levels and transitions of the MSU state machine

Figure 12 shows a UML diagram of the SP3000 MSU state machine. In the diagram the automation levels as well as all transitions between the levels are documented. The blue lines indicate transitions initialized by the automation. The orange lines indicate transitions initialized by the driver. The diamonds describe decision points within a transition. At this point the MSU checks if the driver is able to take over the control. If yes, the transition is completed successfully. If not, the MSU starts an escalating HMI to bring the driver back in the loop. If the driver does not take over, the MSU starts a transition to the Minimum Risk Manoeuvre (MRM), which means that the automation stops the vehicle comfortably on the hard shoulder.

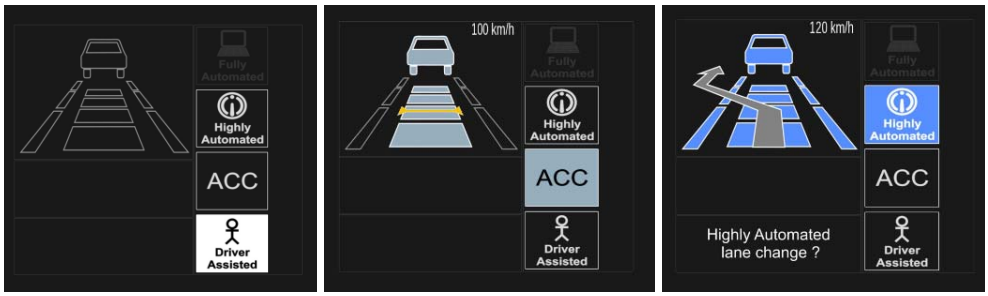


Figure 13: Displays for the three automation levels (Driver Assisted (left), Semi Automated /ACC (middle) Highly Automated with lane change request (right))

Figure 13 shows snapshots from visual HMI in the automation levels DA (left), SA (middle) and HA (right). A more detailed description of the SP3000 HMI is available by the deliverable 33.3 “Validation by Simulation“ ([6], chapter 5).

1.1.5 Command and Haptic Feedback Generation

The main objective of the Command and Haptic Feedback Generation component is on the one hand to ensure the comfortable execution of a requested manoeuvre in terms of trajectory (geometric location and velocity profile) in Highly Automated mode and on the other hand to generate a torque feeling and haptic feedback on the steering wheel and pedals.

This component receives the following inputs:

- Trajectory information from the Co-Pilot component
- Vehicle state from the Data Fusion component (at low frequency),
- Reduced vehicle state from the acquisition component (at high frequency)
- Driver actions (pedals, steering wheel)
- Current and requested level automation from the MSU component
- HMI interface, which contains additional haptic feedback commands from the MSU for transitions

The component provides the following outputs:

- Necessary steering wheel angle value in term of curvature
- Necessary acceleration to follow the speed profile of the trajectory
- The haptic HMI like torque and forces on the steering wheel and acceleration and braking pedals to be felt by the driver.

More details on the functionality of this component will be given in the sequel of this section.

In the SP3000 the *command generation and haptic feedback* component is implemented in dual core PC with Windows XP as operating system. The final version will be implemented on microcontroller CSC-ECU, in particular using static memory allocation rather than dynamic memory allocation. The need for the dynamic memory allocation is motivated in the development phase in order to speed up tests on FASCar.

Command Generation

From the Co-Pilot component, the Command Generation receives the planned trajectory corresponding to the automation level, as a set of way points and velocity profile and brings the vehicle to follow it as good as possible by preserving an optimal level of comfort and security. This constitutes a challenging issue with respect to the complexity of the vehicle dynamics, the variation of the vehicle parameters and jumps caused by trajectories changing.

A straightforward simplification consists in dividing the control task in two decoupled sub-controllers. The first sub-controller influences the acceleration to control the longitudinal motion (longitudinal position and velocity), and the second sub-controller controls the lateral motion (lateral position and heading angle), see Figure 14. With this decoupling longitudinal and lateral controllers are designed separately and their performances can be evaluated separately.

For the longitudinal part, it seems sufficient, for the actual stage of development, to use a proportional and integral controller to control the vehicle's velocity requested by the Co-Pilot. In addition, we have added an anti-windup action to the controller in order to prevent the windup phenomena of the integral action when the control saturates.

Besides, the lateral control is more complex to deal with and needs more powerful control technique than the longitudinal control. Motivated by its development maturity and also its wide range of applications, our choice was made on the H-infinity technique. This technique can deal with multivariable systems including uncertainties in their parameters which represent an adequate framework to address the problem of the vehicle's parameter variation such as mass and adherence for example. Furthermore, this technique offers more flexibility to act on the quality of the control action in term of smoothness and also to reject external perturbations like trajectory curvature and wind gust.

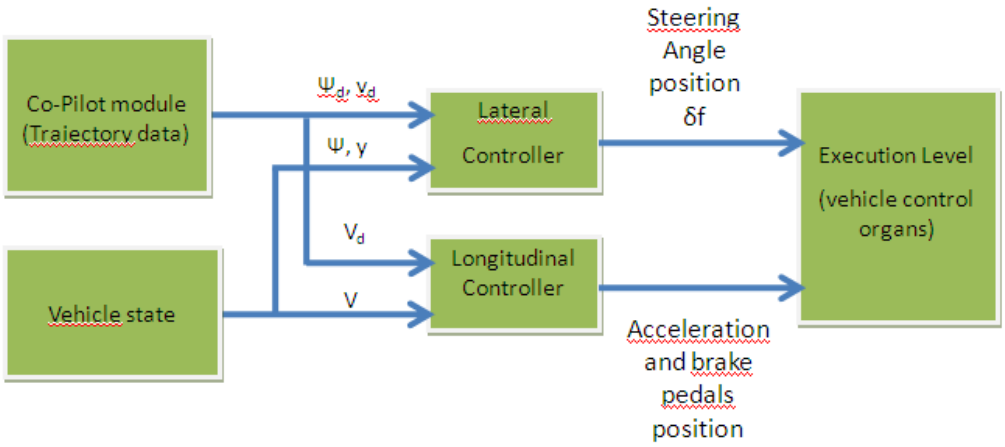


Figure 14 Controller architecture

Haptic Feedback Generation

The Haptic Feedback Generation (see Figure 15) calculates the general haptic feedback, for example the haptic HMI for the feedback in case of lane departure, which is part of the highly

automated system. This calculation is performed by the functions “Criticality based feedback generation” together with the state machine for the “h-feedback pattern”. Therein, the component uses the output of the Command Generation (curvature and acceleration of current trajectory), the environment information from the Data Fusion, and the manoeuvre description from the Co-Pilot.

By the “additional feedback analyser” (see Figure 15) the Haptic Feedback Generation also manages the additional, transition specific haptic feedback signals requested by the MSU.

The functionality of all haptic feedback patterns is stored in the “feedback pattern” library.

Finally, the “feedback manager” prioritises and combines the generated general haptic feedback and additional haptic feedback to one output of the Haptic Feedback Generation which controls the active pedals and steering wheel of the demonstrator.

Haptic feedback examples:

- The Haptic Feedback Generation warns the driver via a vibration and force at the steering wheel if he starts to leave a lane unintentionally (LDW).
- In the automation level Highly Automated (HA) the automation controls the vehicle by modifying the steering momentum, thereby adapting the feedback torque for a certain steering angle. This still allows the driver to slightly influence the lateral control of the automation in HA. Additionally, the driver is able to feel the automation at the steering wheel.
- During a transition from DA to HA the Haptic Feedback Generation acknowledges the handing over of the lateral control from the driver to the automation by a short specific activation vibration at the steering wheel

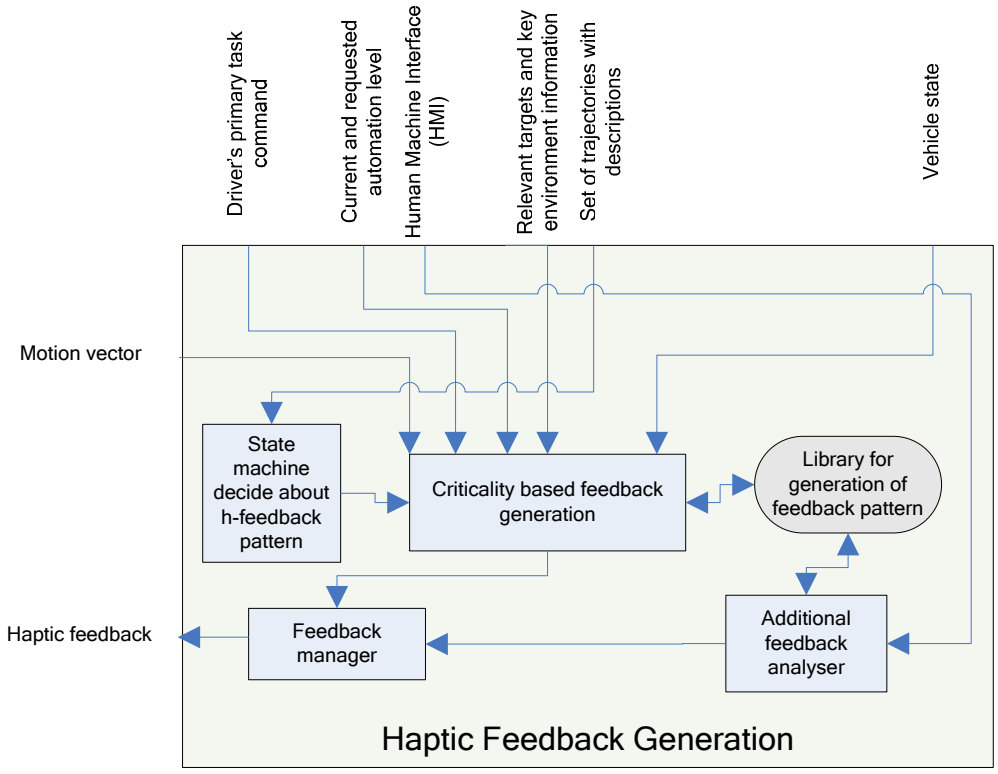


Figure 15: Haptic Feedback generation architecture

1.2 Configuration of Demonstrator Vehicle

The demonstrator vehicle FASCar (Figure 16) serves as test and development tool for the Joint System vehicle integration and demonstration. The vehicle properties allow demonstrating the full span of Joint System technology after finalising the commissioning phase. Its electronic architecture enables the control of longitudinal and lateral dynamics as well as enhanced HMI functionality like driver feedback methods (haptic, visual and acoustic feedback). The vehicle is also equipped with sensors for environment perception. The interface to the vehicle electronics provides electronic access to actuators for steering, brake and throttle which enables on-board Co-Pilot units to intervene into the current vehicle manoeuvre. Due to the demand of safety aspects the additional vehicle electronic devices are partly realized with redundant hardware systems and its software has been developed taking safety developing methods into account that were designed for avionic software systems.

Figure 16: Demonstrator vehicle FASCar

The main sensors used in the FASCar to provide environmental information to the Joint System are a laser scanner system integrated in the demonstrators' front bumper and an in-vehicle camera connected to an image processing unit for lane marking detection. For development purposes this sensor system is complemented by high-precision DGPS data. An overview of the vehicle's sensor equipment is given in Table 1.

Base vehicle	Volkswagen Passat B6 Extended interface to system electronics Drive-by-wire modifications Additional sensors
Environment perception	Lane detection camera Laser scanner object tracker

Additional sensors	Differential GPS unit I2V and V2V communication
Additional actuators	Active throttle pedal Driving wheel feedback actuator Active brake booster
System architecture	Redundant data management system Redundant ECUs based on avionic systems Fail silent ECUs FlexRay bus components

Table 1: Configuration of demonstrator vehicle FASCar

The FASCar has an open system architecture which provides an optimal development environment for rapid-prototyping of driver assistance systems and for tests and evaluations of close-to-production systems provided by the HAVEit partners. These close-to-production systems are integrated either as pure software code running on our hardware or even feature their own ECU, communicating via CAN-Bus or FlexRay. The validation of the assistance systems is conducted by means of specially adapted methods.

1.2.1 Laser Scanners

The laser scanner system which is developed, adapted and provided by SICK in the HAVEit project consists of three single LUX sensors integrated in the demonstrators' front bumper. One sensor is installed in the centre of the vehicle looking straight ahead. Two further sensors are integrated in the left and right front corner of the demonstrator facing outwards at 30° to 40°.

The main purpose of the laser scanner system is the detection and tracking of other vehicles, pedestrians and stationary objects (guard rails) as basis for the data fusion which will calculate an overall environment perception taking the information of further sensors into account. The latest version of the laser scanner software also provides the opportunity of detecting lane markings. Together with the lane marking information of the front camera this additional feature might help to provide redundant and robust lane information.

Power supply, synchronization and data provision is performed by a central switch box. The LUX ECU processes raw data including time alignment, low level fusion with map data, object tracking and classification as well as relative positioning by providing relative distances to landmarks such as lane markings and crash barriers. Data acquired from the scanners are processed in a control unit supplied by SICK which provides a consistent object list aggregated from measurements of all installed units to other HAVEit applications. This resulting laser scanner output is provided as input to the HAVEit fusion system. Table 2 shows the key parameters of the laser scanner system. A view on one single LUX laser scanner and its mounting position in the front bumper of the vehicle is shown in Figure 17.

Detection specification	
Field of view Single laser scanner (tail scanner)	Horizontal coverage: -54° … + 64° (118°) Vertical coverage: 3.2°
Field of view Front system (3 scanners, partly overlapping)	Horizontal coverage: 200°
Effective range	200 m
Accuracy	0.1 m
Update rate	12.5 Hz, 25 Hz or 50 Hz
Sensor input	
Subject vehicle state	Vehicle velocity Vehicle yaw rate Steering angle GPS Position
Sensor output	
Object data (single object list for all scanners installed)	Classification (truck, passenger vehicle, …) Position (lat. and long.) Geometric object outline Velocity and direction of object Lane properties (width, type)

Table 2: Key parameters of laser scanner system

Figure 17: Laser scanner component and its final position mounted below the beams

1.2.2 Lane Detection Camera

The lane detection sensor in the FASCar is based on a detection system developed by Continental, a compact unit featuring the video camera and processing circuit in a single case. As a model improvement of the Passat B6, Volkswagen has recently introduced a lane keeping system based on this sensor type which is included in the Joint System demonstrator vehicle. Due to a modification of the firmware, this sensor is now providing raw data via the CAN bus and therefore no additional sensor has to be installed in the vehicle. The calibration of the sensor has been carried out by Volkswagen itself during the serial manufacturing process of the vehicle. Information provided to the Joint System is summarized in Table 3. As input the front camera system needs information about the ego vehicle state, e.g. subject vehicle speed and subject vehicle yaw rate. As an output the camera sensor provides information about the lane (e.g. lane width, lane curvature, lane marker width and type) and about the lateral position of the subject vehicle within the lane. The cycle time of this lane information is approximately 40ms.

Detection specification	
Image Sensor	CMOS
Resolution	640 x 480 pixels
Range	Maximum: 40m … 60m Minimum: 4m
Dynamic of camera	110 dB
Field of view	horizontal coverage: 42° vertical coverage: 30°
Cycle Time	40ms
Sensor input	
Subject vehicle state	Direct connection to power-train CAN of the vehicle
Sensor output	
Lane information	lateral position of the vehicle in the lane lane width lane curvature marker width marker type heading angle

Table 3: Key parameters of the front camera system

This front camera system is mounted behind the windshield. Together with the rain and light sensor it is integrated in the bracket of the middle rear view mirror (Figure 18). The hardware configuration is used in Passat cars for series production.

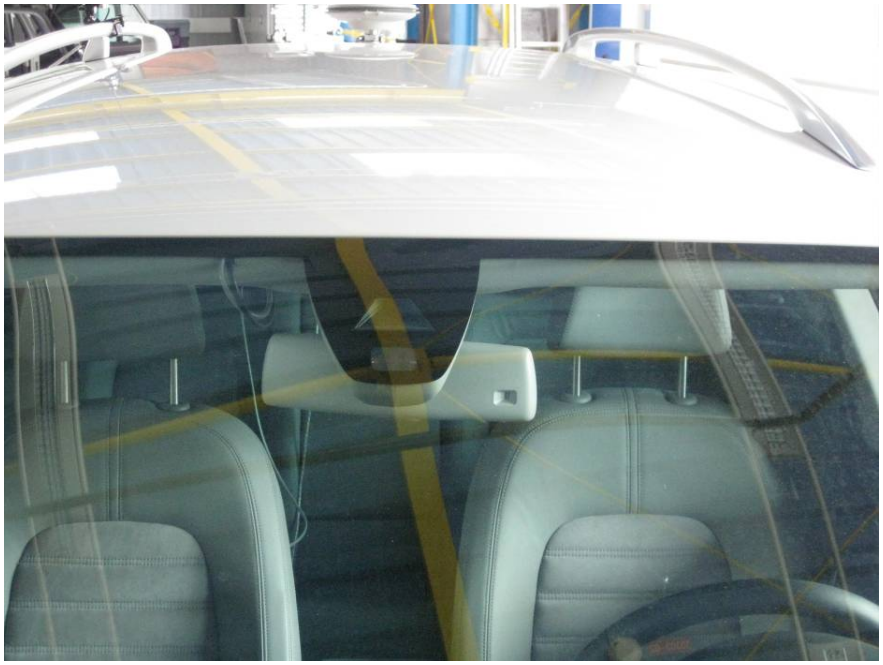


Figure 18: Location of the lane detection camera mounted above the rear mirror

1.2.3 GPS

The vehicle is equipped with a DGPS positioning unit in order to receive lane information from a digital map in order to extend the data fusion by lane matching and to support the fusion during commissioning activities. A Novatel SPAN CPT GPS system has been installed and calibrated providing all positioning information on CAN bus. The satellite receiver of this system is capable of RTK (real time kinematic) positioning operation, which means that very precise positions can be provided with low latency even when the vehicle is moving. This operation mode is only available if differential correction data is provided and satellite visibility is not obstructed. An empirical value often stated for the horizontal positioning accuracy of such systems is 2 cm with a latency of less than 50 ms, however due to the nature of the GPS system there is no guarantee of accuracy. Precision estimates given by the manufacturer are stated in Table 4.

The component has therefore been installed on a carrier plate in the trunk of the vehicle where it is firmly fixed to the vehicle body and carefully aligned with the vehicle's axis. While the SPAN CPT can automatically correct a deviation of its mounting position from the vehicle's centre of gravity, acceleration measurements can be improved by explicitly providing an offset vector taking the position of the centre of gravity of the vehicle into account. The external GPS antenna is mounted on the roof of the vehicle, represented in Figure 19.

General sensor data	
Relevant output	Position (WGS84) in Latitude / Longitude or UTM Heading vs. true north GPS system time
Update rate	100 Hz (with IMU, GPS only: 20 Hz)
Sensor input	GPS correction data, e.g. according to the RTCA specification
GPS Accuracy	
RTK	1 cm + 1 ppm (of baseline to correction station) RMS
Single solution (no correction data available)	5 - 10 m
Dead reckoning accuracy	
10s GPS outage	20 cm, 0.02 degrees
30s GPS outage	1.5 m, 0.03 degrees
60s GPS outage	6.0m, 0.05 degrees

Table 4: GPS/IMU system parameters

Complementary to the satellite receiver, the SPAN CPT system is equipped with an inertial measurement unit (IMU) consisting of three accelerometers and three gyrometers. The SPAN CPT system has a tight coupling between GPS and IMU data, thus allowing the GPS positioning algorithm to perform better under adverse satellite reception conditions. This concept goes one step beyond conventional GPS/IMU applications, in which typically position data from a standalone GPS receiver are combined with IMU measurements in a Kalman filter. It is generally very promising in the automotive domain, where bad satellite reception is much more common than in aviation or maritime applications.

However, in order to keep systems closer to series-production status, the HAVEit specification requires only standard consumer-grade GPS data to be available in the vehicle thus the final HAVEit systems will not depend on RTK data. Nevertheless, the availability of such information is particularly useful for testing and validation purposes as well as temporary bypassing sensor systems for e.g. controller development.

Figure 19: GPS receiver antenna mounted on the roof of the vehicle

1.2.4 Communication Hardware I2V, V2V

Figure 20: 4G Cube – compact wireless router

The communications hardware [3] added to the FASCar demonstrator vehicle consists of a 4G Cube (Figure 20) – a compact wireless router running SCOPE and OPENWRT router operating system – connected by Ethernet to the already existing vehicle onboard computer where the HAVEit Joint System Framework is running.

This communications system is a Vehicle Web Service Communication Framework (VWSCF) also called SCOPE that has automatic service discovery.

This communication system is already integrated into the HAVEit Joint System Framework that is running in the FASCar demonstrator vehicle and is suited for infrastructure-to-vehicle (I2V) communications as well as for inter-vehicular communications (V2V).

The communications component is connected to the data fusion. There is no direct link between the on board computer and infrastructure. Communication is managed by 4G Cubes (provided by partner INRIA) and vehicle/infrastructure data is propagated to the dynamic mesh network.

Hardware	
CPU	x86 500 MHz AMD Geode LX800
Software features	
	OLSR Mesh
	Service discovery using OLSR
Interfaces	
Web interfaces and protocols	HTTP Web services Scope Server IPV4 / IPV6
Communication ports	1 mini-PCI a/b/g Serial port USB
Graphics interface	VGA output

Table 5: 4G cube features

The communication system is installed in order to extend and to improve the world perception in the vehicle by providing additional environmental information that the onboard sensors cannot provide due to their limited range.

The use of the wireless communication allows interactions between infrastructure and vehicle (I2V) as well as between vehicles (V2V).

For each situation, I2V and V2V, a scenario will be demonstrated to prove the concept and to validate the proposed wireless communication system:

- I2V: a dynamic speed limit is sent by the infrastructure to the FASCar demonstrator vehicle that will adjust its current speed accordingly.
- V2V: a priority vehicle coming from behind (were there is a limited sensor coverage) will inform the ego vehicle of its presence that will remain in its current lane not being allowed to perform lane changes.

The communication architecture is modular and based on embedded Linux boxes (4G Cubes) that are used to provide automatic connectivity for X2V applications. These boxes can be used as simple routers, or like advanced communication units. The 4G cube features are detailed in Table 5.

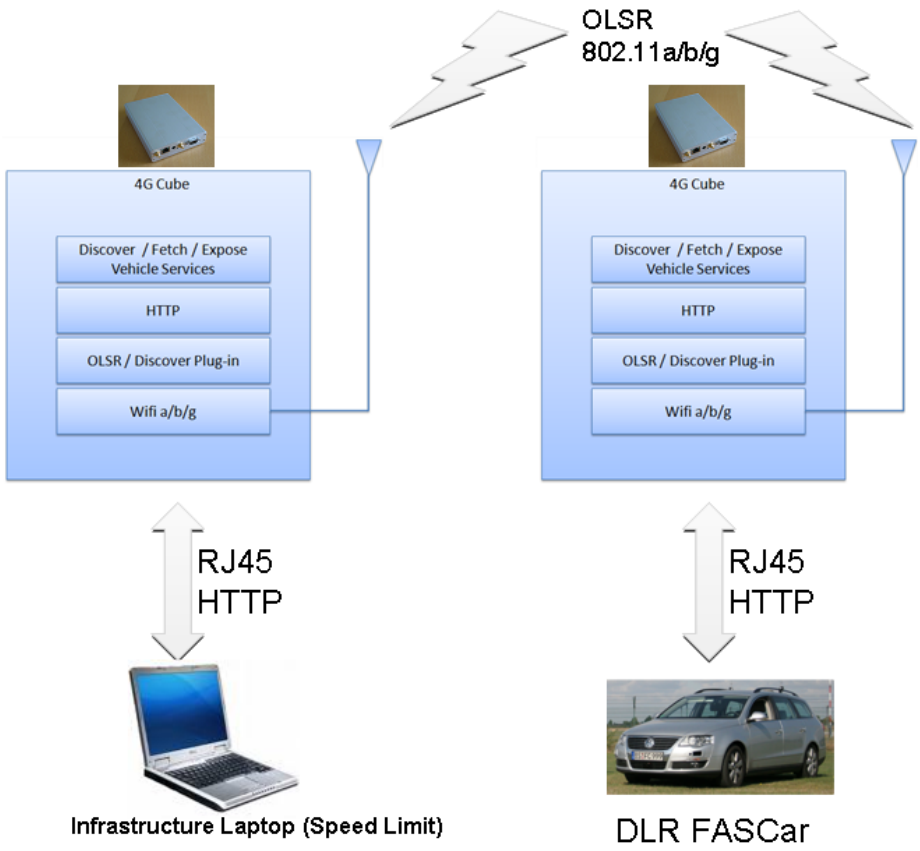


Figure 21: I2V communications architecture

1.2.5 Driver Detection Camera

The Driver Monitoring System (DMS) provided by CAF is a vision based system observing head and face of the driver in order to provide information about his/her state of degradation (Figure 22 and Figure 23). The system thereby provides support not only in the event of drowsiness, but also during actions which temporarily divert attention from the driving task, such as inserting a CD or operating the navigation equipment. Thanks to a compact camera module including a CMOS sensor and pulsed NIR light sources (invisible near infrared light) DMS monitors driver's drowsiness (DDM) and inattentiveness (DIM, meaning “distraction” in HAVEit). A distant processing unit analyzes the image flow provided by the camera to extract information about the driver's eyelid and blinking patterns and face characteristics. The system electronic functions use these parameters to reliably determine driver's drowsiness and inattentiveness.

The system works by night and day taking into account that drowsiness is not just a night-time phenomenon and is also likely to occur during the morning or the afternoon. The system is fully automatic; it doesn't require any input or specific behaviour from the driver.

The image processing algorithms of the DMS application are designed for the camera component to be mounted on the dashboard or in the instrument panel. The camera component position and orientation are optimized for monitoring the driver's face in the axe of the driver's eye ellipse midpoint.

The outputs of the DMS serve as input for the DSA software component (see Section 1.1.3).

Figure 22: Camera for direct driver state assessment

Figure 23: Detection of the driver drowsiness

1.2.6 Vehicle Data Interface

The vehicle system architecture offers one private CAN port that enables bidirectional access to relevant vehicle data (see Table 6). With the ego-vehicle state provided by this CAN access and the additional information of the environment perception the joint system is able to generate a command vector that performs longitudinal and lateral control of the vehicle. This motion vector contains the demanded values for engine torque, brake pressure and steering angle and is extended by further values like gear-shift command, wipers or other HMI-commands.

Received data	
Environment perception data	Laser scanner data Lane detection camera data D-GPS data
Dynamic ego vehicle data	Accelerations Steering angle Yaw rate …
Inputs of the driver on the HMI	Buttons Pedal forces and position Steering wheel angle
Transmitted data	
Vehicle motion commands	Engine torque Brake pressure Steering angle
Other commands	Gear shift Wipers Horn

Table 6: Set of available, relevant data

While the data exchange of the sensor values and the set points for the longitudinal control uses the CAN communication the more safety and time critical steering system is connected to the steer-by-.wire system of the vehicle via FlexRay. An overview about the relevant CAN architecture is given in Figure 24. The vehicle behaves as slave to the Joint System. The demanded motion control will be carried out automatically.

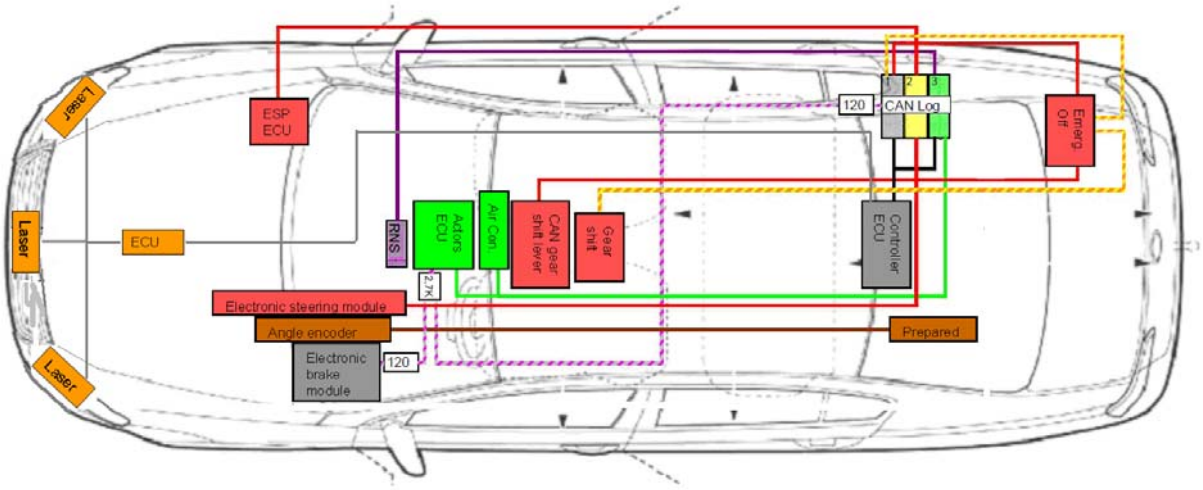


Figure 24: Basic system architecture of the FASCar

1.2.7 Drive-by-Wire Interface

The drive-by-wire interface of the vehicle allows a separate control unit (here the Joint System) to take control over the engine, the braking system and the steering system. The engine torque can be commanded by a torque vector that sets the current torque recommendation for the engine. For this purpose an electronic box has been implemented that emulates the voltage signal from the position sensors of the vehicle. In addition to that the gas pedal enables the Joint System to command separate force feedback signals, e.g. variable force points and vibration patterns are applied to the force feedback pedal.

The deceleration of the vehicle is commanded by a brake pressure signal which can apply up to 150 bar braking pressure on the serial hydraulic brake system. An active brake booster maintains both the mechanical connection of the braking pedal with the master brake cylinder as well as the pneumatic brake assistance function via its two vacuum chambers, thus ensuring that manual brake operation by the driver is possible at all times. Additionally, an electromagnetic valve is installed, enabling the device to actuate the main brake cylinder based on the electronic command. Typically, active brake boosters have an assigned ECU which accepts brake demands via a CAN interface and controls the hydraulic pressure in the braking system.

While the actuators for longitudinal control are series products, the steer-by-wire system is a prototype system that requires additional sensors. A specific steer-by-wire system has been developed that ensures steering intervention with a very high reliability. Its system architecture is presented in Figure 25. The commanded steering values generated by the Joint System are transformed onto the FlexRay and XCC architecture (X-by-wire Control Computer, see [2]), which are working in a complete redundant mode, including hardware components and software processes. This safety architecture and its components are explained in detail in [1].

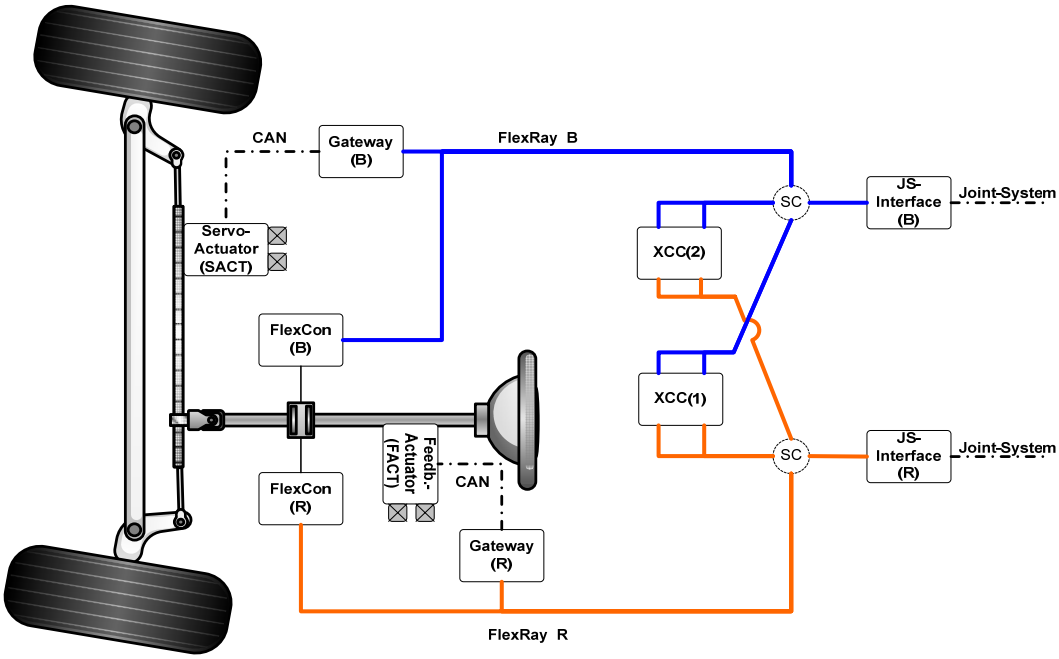


Figure 25: Steer-by-wire architecture

When the system operates in steer-by-wire mode, the safety clutch is open, thus releasing the mechanical connection between the front axle and the steering wheel. Haptic feedback to the driver is generated via the driver feedback actuator while the steering task is performed by the servo steering actuator. In case of a failure in one of these actuators, the clutch can be engaged in order to restore the mechanical connection between the steering wheel and the front axle and therefore allow the driver to control the vehicle via a conventional steering system. Each of the two redundant XCC control units can command the engagement of the clutch. As an inherent safety feature, the clutch closes also automatically by means of a permanent magnet whenever the power supply is cut. This ensures that the system will enter a safe state (unassisted mechanical steering) even in case of total loss of electric power.

1.2.8 CSC-ECU

The CSC electronic control unit shown in Figure 26 provides scalable microcontroller performance in order to implement different vehicle functions concerning chassis control, driver assistance systems as well as safety functions. Communication is provided optionally using CAN or/and optionally FlexRay.

The main features are listed in Table 7. A detailed CSC ECU description can be found in HAVEit Deliverable 21.1 “CSC & ´XCC HW-development completed” [2]

Figure 26: CSC ECU

Flash	3MB
RAM	160KB
Max. clock frequency	150MHz @Ta = 105°C
Safety classification	redundant microcontroller MCU
Interfaces	3 x CAN controller 1 x FlexRay controller
Dimensions of box	width 129mm length 172,4mm height 36mm

Table 7: Main features of the CSC ECU

The Joint System demonstrator is working in its final configuration with three CSC ECUs included in a CAN network. On these CSC-ECUs three software components of the Co-System as part of the Joint System will be executed:

- Driver State Assessment (see Section 1.1.3)
- Mode Selection and Arbitration Unit (see Section 1.1.4)
- Command and Haptic Feedback Generation (see Section 1.1.5).

2 System Validation and Status Report by Use Cases

At the beginning of the HAVEit project a use case catalogue was developed that covers all use cases that should be in the focus of this project. The use cases do not cover all possible situations of driving a highly automated vehicle but were selected to deal with the most important situations still manageable for the realization and testing. Scenarios are defined in HAVEit as a sequence of use cases, instantiated in a simulator or test ground. An overview of the HAVEit use case classes can be found for example in the deliverable D33.3 [6]. In the current validation, approximately one year before the end of the project, some of the most important use case classes were selected to test and validate the joint system.

In each of the following sections, one use case class is addressed and its validation in a test vehicle is discussed. Each section is structured analogously: Firstly, it contains a brief description of the addressed use case class and its concrete application in the test run. Secondly, an overview over the used system components is presented. It details which components were completed and used already as specified and which ones were modified for the testing or only partly used. The component Driver State Assessment will be integrated as planned later in the project, and was therefore not included in this validation. To illustrate each use case validation a short video sequence is shown and discussed. Further, up to two diagrams are included to detail the most important aspects of each use case, such as the planned and driven path during a lane change or the measured distances to an obstacle. Each section concludes with a short summary.

The tests were performed on a test site in the deserted military base in the vicinity of Braunschweig, shown in Figure 27: The test track has some shortcomings, one of the most prominent being the middle lane marking which is a solid and not a dashed line. To overcome this drawback, the middle lane markings had to be artificially set to dashed in the software. In addition, there are only very small safety areas to the sides of the used potion of the test track. Thus, only very low velocities could be tested safely.

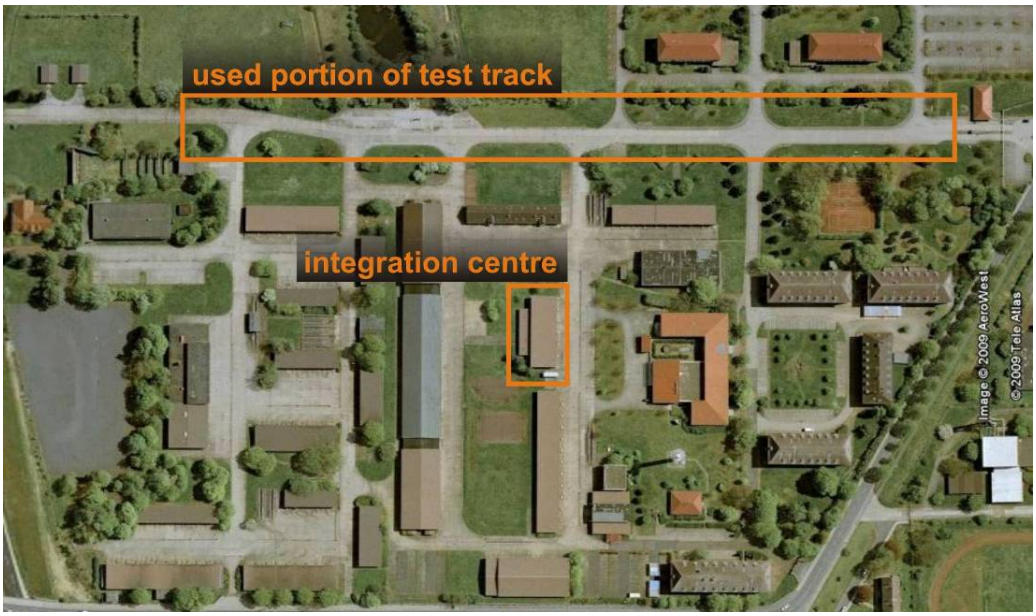
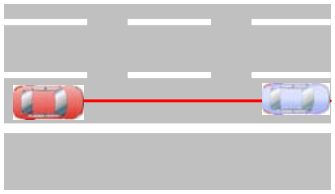


Figure 27: Test site in deserted military base near Braunschweig

As the final HAVEit Joint System demonstrator vehicle is still in its commissioning phase, validation was performed using a preliminary test vehicle, which is equipped almost identically to the final FASCar demonstrator vehicle. However it does not offer steer-by-wire capability and poses a few more constraints with regards to accessibility of vehicle signals and usable displays. Therefore, it was not yet possible to test the generation of haptic feedback to the driver in this preliminary vehicle. Further, due to limited accessibility of vehicle signals, additional buttons instead of the indicator lever had to be used to initiate lane changes.

Finally, some software and hardware development steps have not yet been completed. The tests were performed using the Joint System software framework running on windows computers. The integration of CSCs, on which certain system components will run in future, is planned for fall 2010. Also, still a preliminary version of the longitudinal controller was used.

2.1 Use case “Driving and Detected Obstacle in Highly Automated”



The use case class Driving and Detected Obstacle (see deliverable D33.3) deals with a deceleration and braking situation triggered by a detected obstacle in front of the ego vehicle.

As the most challenging sub-case, driving in automation level Highly Automated was tested. In the test the ego vehicle is driving on the right lane of a two-lane road at a velocity of approximately 5.5 m/s. Then an obstacle appears on the same lane. The car automatically stops right behind the obstacle, while the driver is driving hands-off and feet-off.

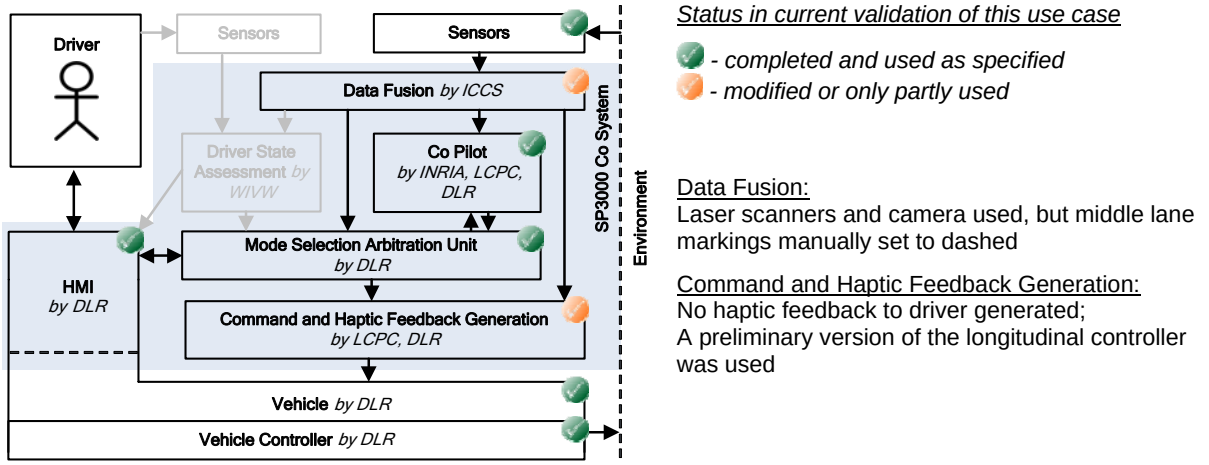


Figure 28: Driving and detected obstacle in HA: Used system components

Frames 1-6 of the video sequence in Figure 29 show how the ego vehicle was driving in the automation level Highly Automated in the right lane while approaching an obstacle ahead. The ego vehicle maintained the desired velocity and compensated lateral deviations. When the obstacle came closer as in frame 4 to 5, it was necessary either to brake or to brake and change the lane to the left, which can be seen also in the computed trajectories and in the action grid in Figure 31. Since the driver did not initiate a lane change, the co-system braked and came to a stop after frame 6. The planned and driven velocity profiles are shown in Figure 30.

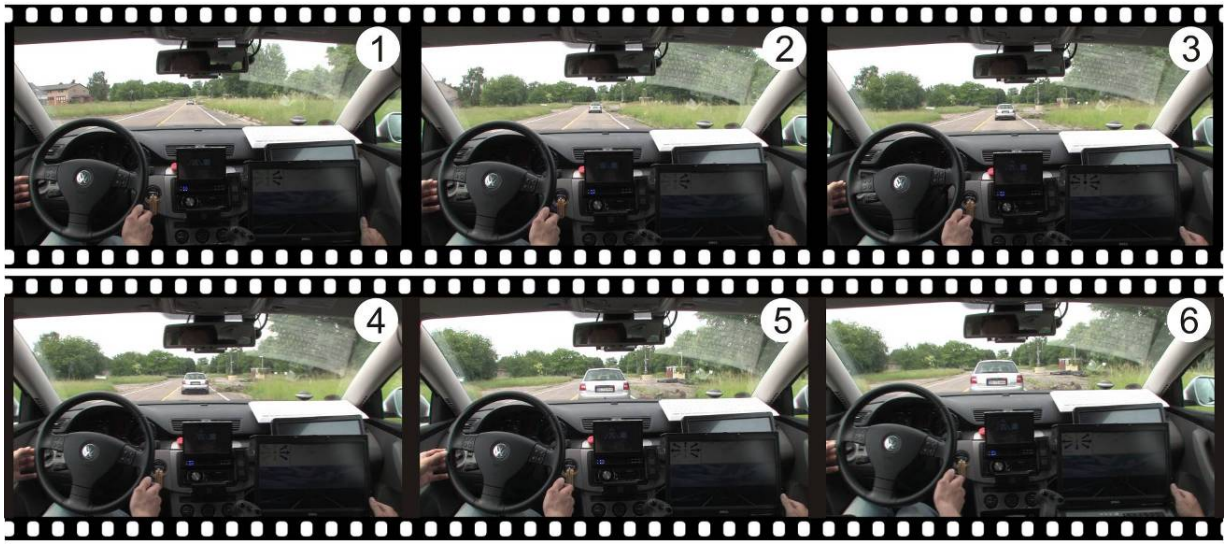


Figure 29: Driving and detected obstacle (HA): Video sequence

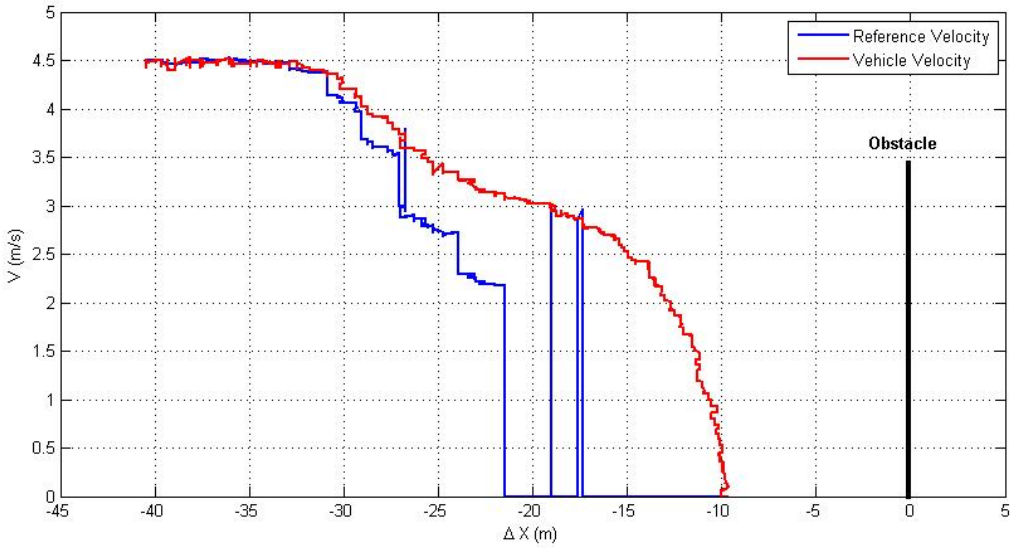


Figure 30: Driving and detected obstacle (HA): Planned and real velocity

Figure 30 shows the performance of the longitudinal controller in terms of the velocity control. In this figure the reference velocity to be followed by the vehicle demonstrator, blue curve, is computed from the planned velocity profile by taking the velocity of the closest point of the planned trajectory to the vehicle at each longitudinal controller cycle. For this reason, the profile of the reference velocity is not continuous. Sometimes peaks of the reference velocity occur when the data fusion model detects false obstacles or looses obstacles (see peaks between -20m and -15m). We will address and improve this issue in the next integration phase.

The red line in Figure 30 signifies the actual vehicle velocity that is controlled by the longitudinal controller. We can see that the actual longitudinal controller has a significant time reaction but the whole system is able to stop the vehicle sufficiently behind the obstacle (10m of distance). However, we should take a deeper look at this situation to tune the longitudinal controller to

enhance the accuracy of the controller while at the same time preserving the passengers' comfort.

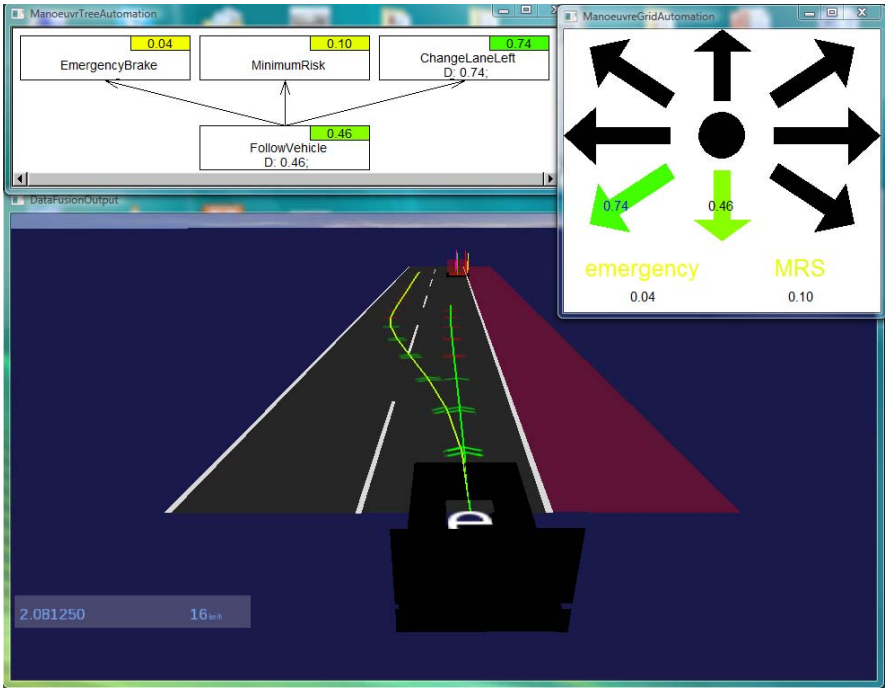


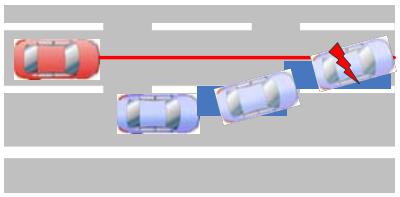
Figure 31: Driving and detected obstacle (HA): Planned trajectories, manoeuvre tree, and manoeuvre grid

Figure 31 illustrates the manoeuvre tree (grid on the upper left and right) and the planned trajectories; further, it provides a rough representation of the environment (bottom part). It shows that the standing lead vehicle has been detected by the laser scanners and the road has been correctly observed by the lane camera. The manoeuvre tree shows the currently performed manoeuvre "FollowVehicle". Three future manoeuvres are available with different valentials: "EmergencyBrake", "MinimumRisk", and "ChangeLaneLeft". The manoeuvre grid depicts the options to either brake or to change lane to the left and brake. (For more details on the interpretation of this representation see Chapter 1 and 3.)

For the selected manoeuvres, detailed trajectories are calculated. In case of a lane change, the corresponding trajectory indicates an acceleration and some deceleration later on, whereas the trajectory for the "FollowVehicle" manoeuvre indicates to first accelerate and then brake stronger behind the obstacle. As observed in the video sequence in Figure 29, this trajectory is executed, finally coming to a stop behind the stopped lead vehicle.

The use case "Driving and detected obstacle" has been successfully validated in the FASCar. Switching between different automation levels was possible. In the automation level Highly Automated, the vehicle successfully drove in its lane and stopped behind a standing lead vehicle. Further tests are planned to demonstrate the same use case at higher velocities.

2.2 Use case “Driving and Detected Obstacle - Emergency Brake in Driver Assisted”



In contrast to the previous chapter, which deals with normal driving and an obstacle detected far enough ahead, this use case of the use case class “Driving and detected obstacle” deals with an emergency situation, which is triggered by an obstacle suddenly appearing in front of the ego vehicle.

In this section, the validation of the use case in automation level Driver Assisted is described, at which the reaction of the automation in automation level Driver Assisted was tested. (For the validation in level Highly Automated, please refer to section 2.3.) The expected result was that the automation intercedes by an emergency braking manoeuvre only, if the driver does not react in time to the sudden obstacle. In the test scenario, it was sufficient for that purpose to replace the suddenly appearing dynamic obstacle by a virtual static obstacle at a fixed GPS position. The FASCar was driven on the right lane of a two-lane road in Driver Assisted at about 11 m/s. The virtual static obstacle was programmed to stand on the right lane and - for visualization purposes - additionally marked on the lane by traffic cones. When the driver did not react to resolve the situation, the automation level Emergency was activated and an emergency braking manoeuvre was performed to mitigate the collision.

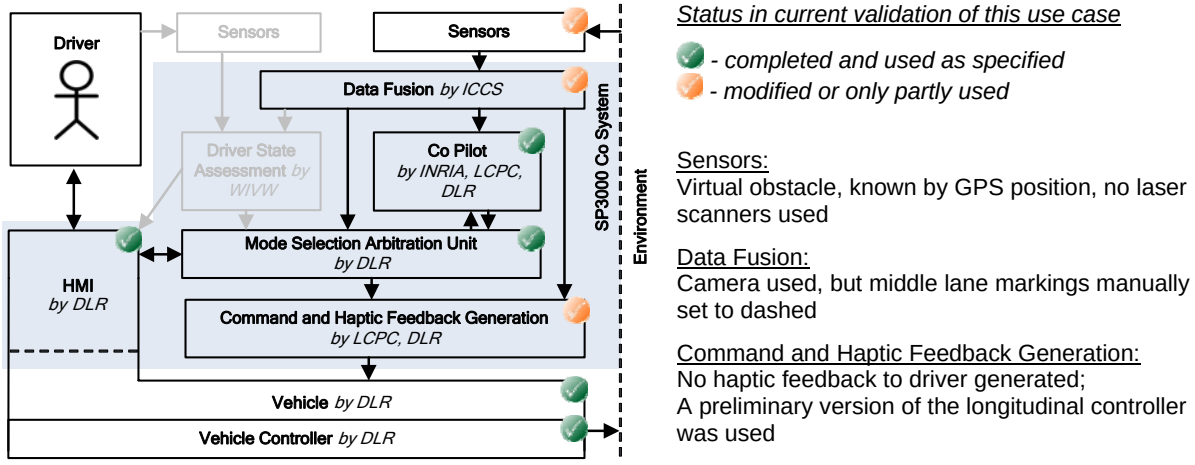


Figure 32: Driving and detected obstacle - emergency brake in DA: Used System Components

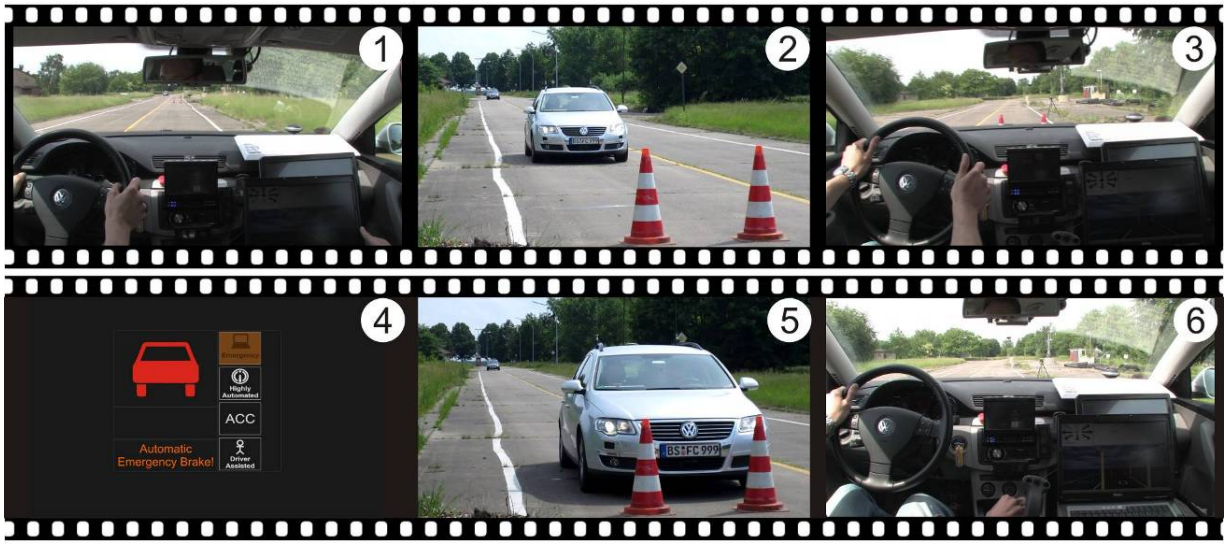


Figure 33: Driving and detected obstacle - emergency brake in DA: Video sequence

As illustrated in Figure 33, the vehicle was driving in automation level Driver Assisted and approached a static obstacle (frame 2). As the distance to the obstacle decreased below a safe threshold, the HMI issued a warning to the driver (frame 3) and activated the automation level Emergency as can be seen in the display shown in frame 4. Thus, the vehicle performed an automatic emergency braking manoeuvre to avoid a collision (frame 5). Once the vehicle came to a stand still, the warning shut off and control was given back to the driver in automation level Driver Assisted (frame 6).

In the displayed video sequence, a virtual obstacle was programmed at a fixed GPS position, such that the traffic cones were located about 30cm after the rear bumper of the virtual obstacle. As can be seen by the small distance to the cones in frame 5, the final distance of the ego vehicle to the obstacle was approximately zero, such that there probably was already some contact at very low velocity.

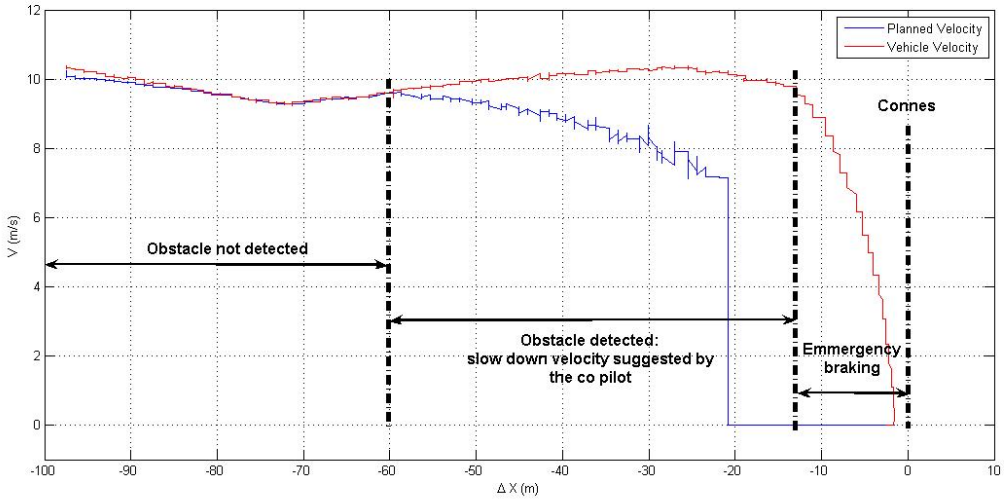
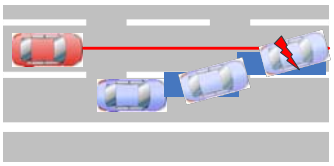


Figure 34: Driving and detected obstacle - emergency brake in DA: Planned and real velocity

Figure 34 shows the actual and the planned velocity profile of the vehicle demonstrator versus the distance separating the vehicle demonstrator and the obstacle. In this test, the driver controls the longitudinal velocity by himself with the assistance of the co-pilot. In the beginning, the system does not detect the obstacle and the planned velocity correlates with the driven speed. Once the obstacle is detected (at -60m distance), the co-pilot plans to slow-down (velocity profile blue) until it requests to stop the vehicle, but the driver ignores this suggestion (as can be seen by the gap between the planned velocity dropping to zero (at distance of -21m) and the initiation of the emergency brake). When the situation becomes critical, the emergency braking is triggered by the system and the vehicle is stopped.

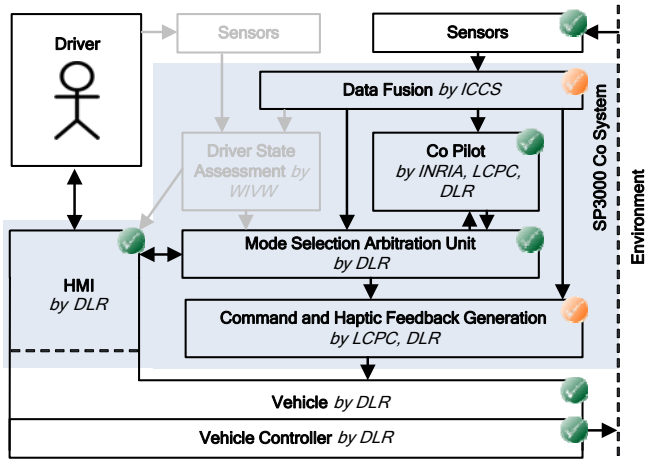
A preliminary version of the use case “Driving and detected obstacle - emergency brake in Driver Assisted” has been validated successfully. As shown, the vehicle performs an automatic emergency brake to mitigate rear-end collisions even in automation level Driver Assisted. In the final version a haptic feedback to the driver will be added to enhance the warning of the driver before the emergency brake is activated. This haptic feedback includes e.g. a vibration on the gas pedal and a brake jerk.

2.3 Driving and Detected Obstacle - Emergency Brake in Highly Automated



As described before, this use case is derived from the use case class “Driving and detected obstacle” deals with an emergency situation triggered by a detected obstacle in front of the ego vehicle. The emergency interaction can be applied to all automation levels.

Regarding the validation of this use case class, in this section, the reaction of the automation in automation level Highly Automated is tested. (For the validation in Driver Assisted, please refer to the previous section.) The FASCar drives autonomously on the left lane of a two-lane road in Highly Automated at about 2.5 m/s (for safety reasons). A team member suddenly holds an obstacle in front of the vehicle. In consequence, the automation level Emergency is activated and an emergency brake is performed to avoid the collision.



Status in current validation of this use case

- completed and used as specified

- modified or only partly used

Data Fusion:
Laser scanners and camera used, but middle lane markings manually set to dashed;

Command and Haptic Feedback Generation:
No haptic feedback to driver generated;
A preliminary version of the longitudinal controller was used

Figure 35: Driving and detected obstacle - emergency brake in HA: Used System Components

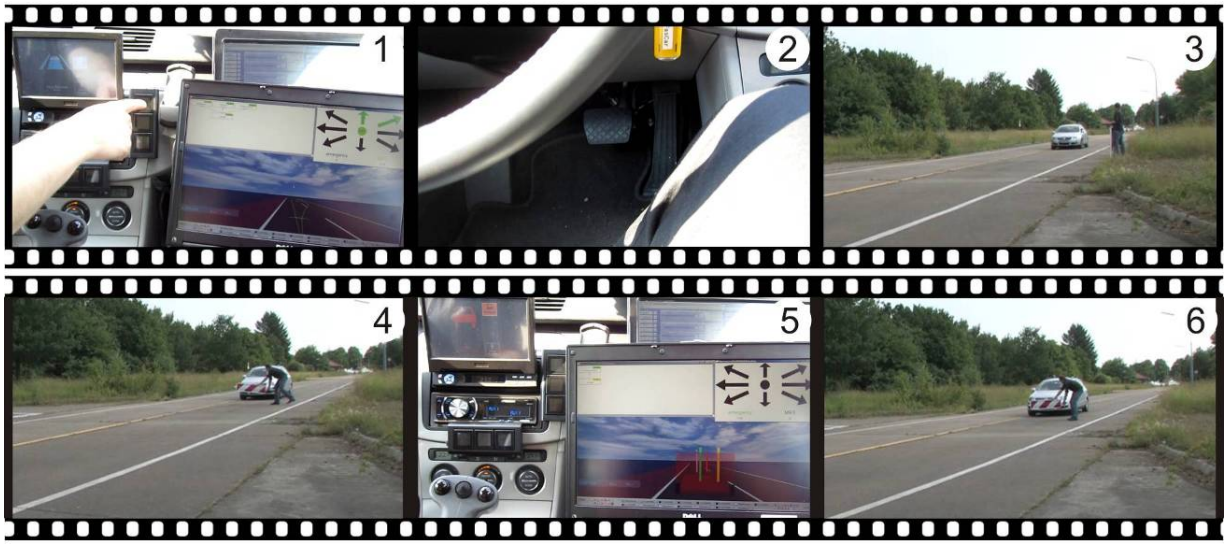


Figure 36: Driving and detected obstacle - emergency brake in HA: Video sequence

As illustrated in Figure 36, the ego vehicle is driving autonomously in automation level Highly Automated (frame 1, 2) in the left lane. On the left road side there is an obstacle (frame 3) which suddenly moves into the lane directly in front of the vehicle (frame 4, time 0.6s in Figure 37). As shown in frame 5, the obstacle is detected and the automation level Emergency is activated, because of the high criticality of the situation which could not be resolved by comfortable braking. Therefore, an automatic emergency brake is performed and the vehicle comes to a complete stop (frame 6).

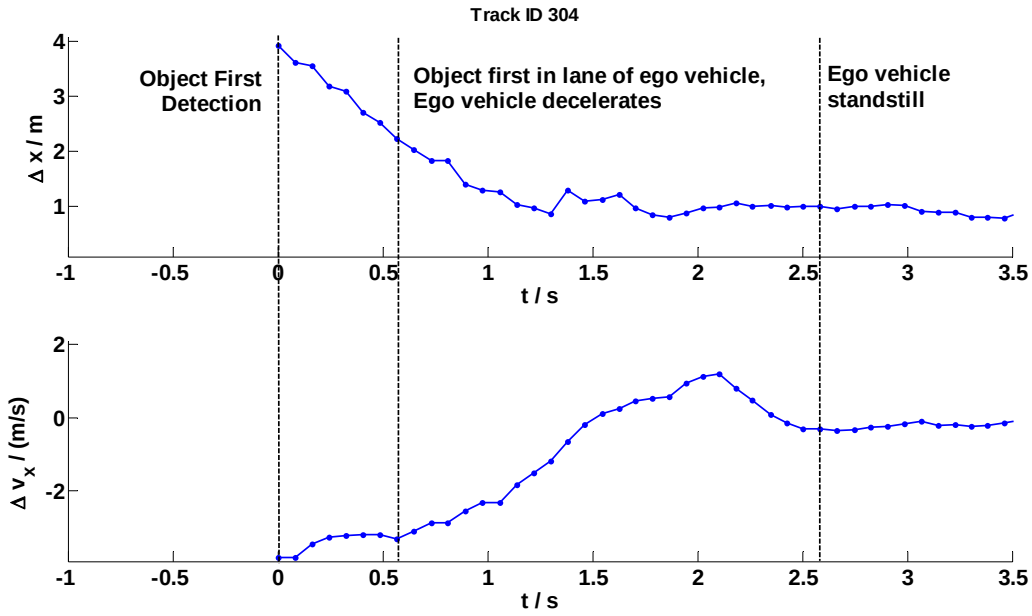


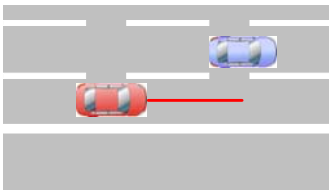
Figure 37: Driving and detected obstacle - emergency brake in HA: Relative distance and velocity of obstacle as determined by data fusion component

Figure 37 shows the relative distance and velocity of the tracked obstacle that appears suddenly in front of the ego vehicle as determined by the data fusion component. It can be seen that the object is detected steadily while the ego vehicle decelerates (relative velocity is negative and finally becomes approximately zero).

When an unexpected obstacle (a pedestrian for example) appears it usually comes from outside the road. The data fusion tracking component tracks the object even if this is outside of the road, but it doesn't report it to the other components. As soon as the object moves in the road it is included in the output of the data fusion, so the Co-Pilot and control components can take proper action. As shown in Figure 37, immediately after the object has entered the lane, the automation level Emergency is activated and an automatic braking is performed, which can be seen by the fact that the relative longitudinal velocity is decaying to zero, avoiding the collision with the obstacle.

A preliminary version of the use case "Driving and unexpected, detected obstacle in front (HA)" has been validated successfully. The vehicle stopped automatically in front of a suddenly appearing obstacle to avoid the collision.

2.4 Driving in a Lane Avoiding Right Overtaking



The use case class “Driving in a lane avoiding right overtaking” covers a specific driving situation, where the ego vehicle is driving on the middle or right lane and a slower vehicle is on the left. The reaction of the ego vehicle depends on the technical capabilities and can reach from no reaction, to some kind of information for the driver up to an automatic avoidance of such situations. For the demonstrator tested here, the expected behaviour is that the automation does not take over the other vehicle.

The use case was tested in the FASCar on a two-lane road, where the FASCar drove in automation level Highly Automated. It followed the right lane at initially approximately 5.5 m/s. A slower vehicle ahead on the left lane drove first at about 2.7 m/s, then accelerated up to 7 m/s and finally stopped on the left lane. It should not be passed on the right at any time (despite the low velocities, which only were due to restrictions of the test track).

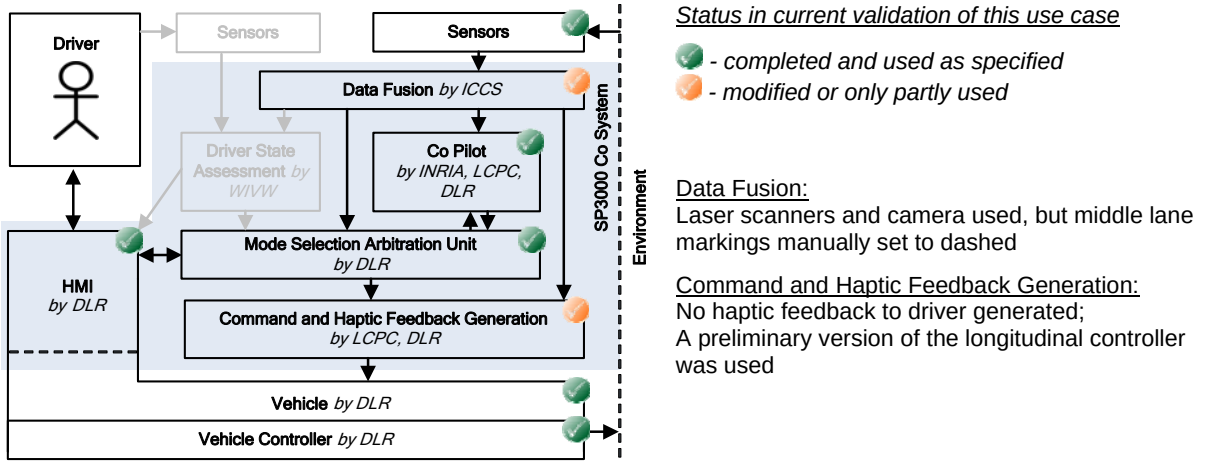


Figure 38: Driving in a lane avoiding right overtaking: Used System Components

As shown in the video sequence in Figure 39, the demonstrator vehicle is driving in Highly Automated mode on the right lane. As displayed in frame 2, there is a slower vehicle ahead in the left lane. The automation recognizes the situation and - in order to avoid right overtaking - offers either to change the lane to the left or to slow down, which is executed (frame 3). Once the demonstrator vehicle is driving at the same speed as the vehicle ahead, the planned trajectories offer to follow the lane or to change to the left lane (frame 4). As the vehicle on the left lane accelerates slightly, the ego vehicle matches this speed up to 5.5 m/s on the right lane. Finally, the vehicle on the left lane brakes to a standstill and the ego vehicle does the same to avoid right overtaking (frames 5, 6).

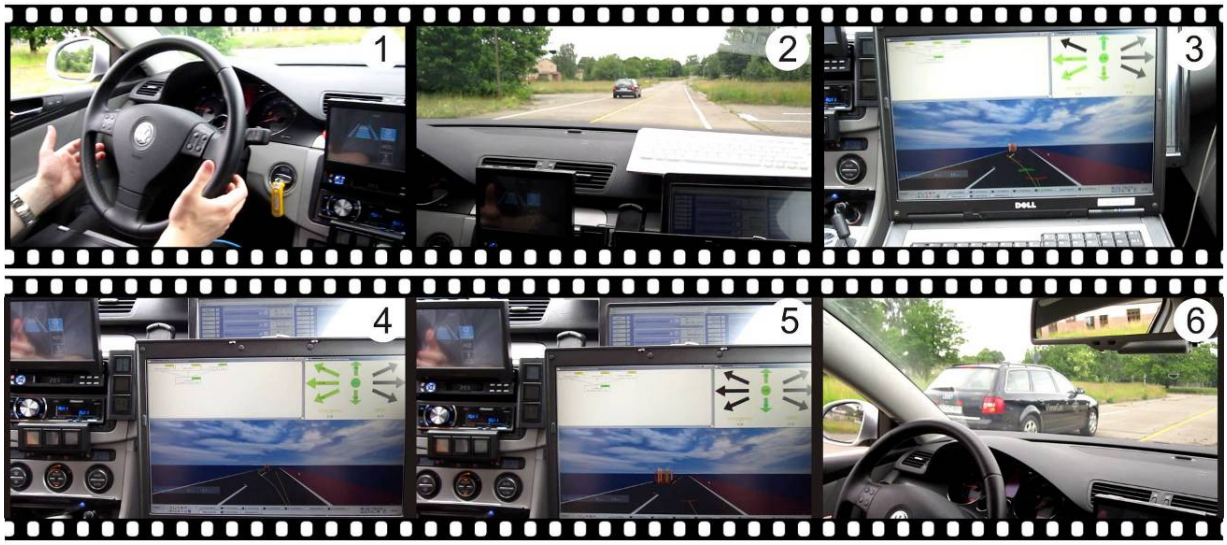


Figure 39: Driving in a lane avoiding right overtaking: Video sequence

Figure 40 shows the situation where the ego vehicle currently performs the manoeuvre “FollowLane” and the vehicle ahead on the left lane has already been detected by the laser scanners. Optionally available manoeuvres are “EmergencyBrake”, “minimumRisk” and “ChangeLaneLeft”. The computed trajectories indicate that in case a lane change was selected, the vehicle should first accelerate briefly and then decelerate to follow the other vehicle in the left lane. In case the current manoeuvre “FollowLane” is pursued further, the corresponding trajectory indicates that immediate braking is necessary in order to avoid right overtaking

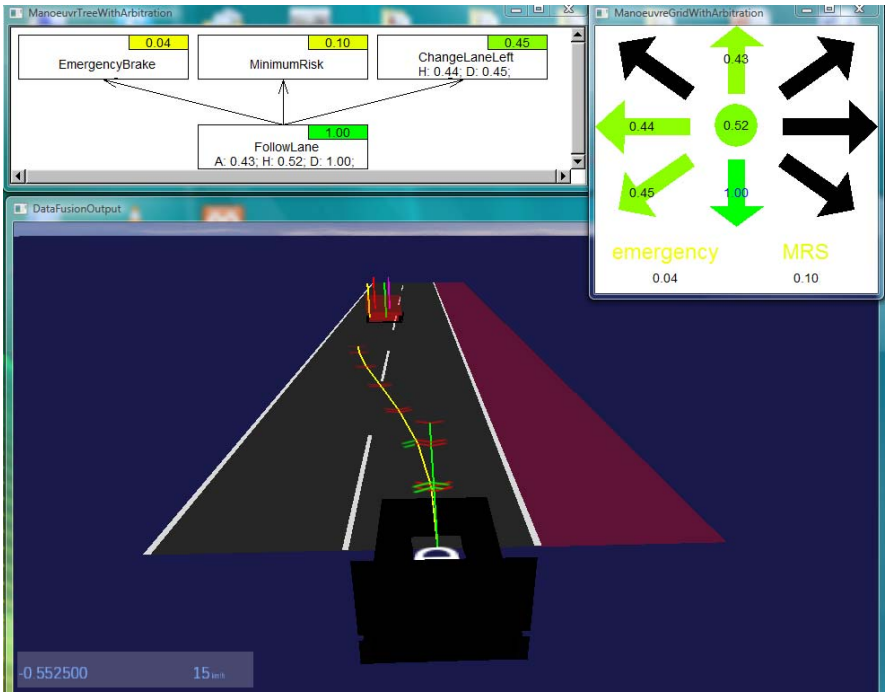
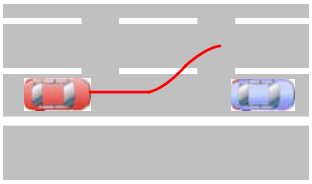


Figure 40: Driving in a lane avoiding right overtaking: Planned trajectories, manoeuvre tree and manoeuvre grid

The use case “Driving in a lane avoiding right overtaking” has been demonstrated successfully. The manoeuvre planning creates appropriate manoeuvres and trajectories to avoid right overtaking and in Highly Automated the vehicle follows the planned trajectories well. Some adaptations are still necessary in order allow right overtaking at very low speeds according to the traffic rules, such that the road is blocked by a standing or very slowly moving vehicle on the left lane.

2.5 Driving and Lane Change



The use case class “Driving and lane change” covers a lane change manoeuvre in different automation levels. In lower automation levels the driver is supported e.g. by a blind-spot warning, in highly automated driving there is the possibility of a manual lane change or a highly automated lane change that is either initiated by the driver or done automatically.

With regards to the validation of this use case in the FASCar, a driver initiated lane change in automation level Highly Automated is tested. A standing lead vehicle on the right lane of a two-lane road was passed by a lane change to the left lane. The FASCar was driving autonomously in Highly Automated mode at a speed of about 10 m/s.

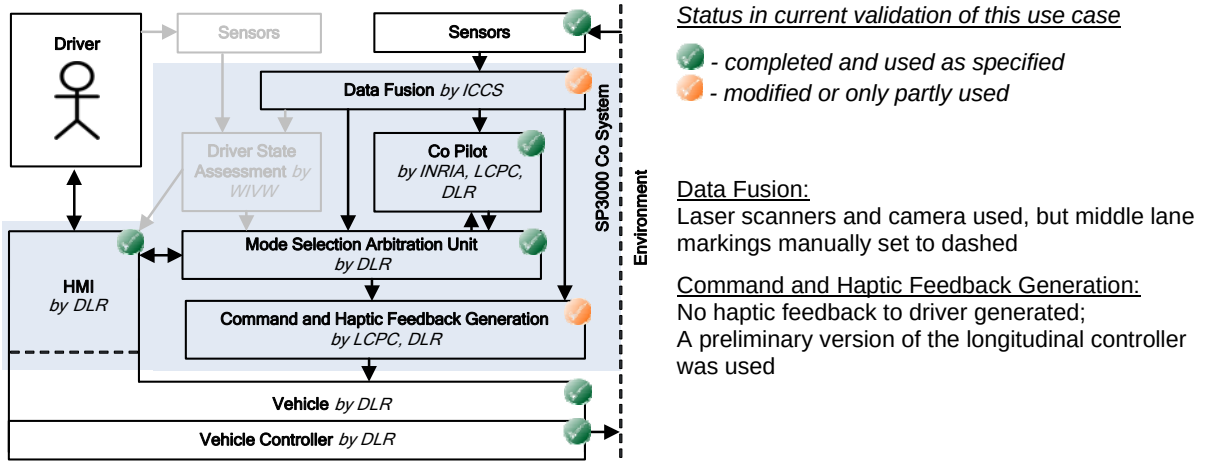


Figure 41: Driving and lane change: Used System Components

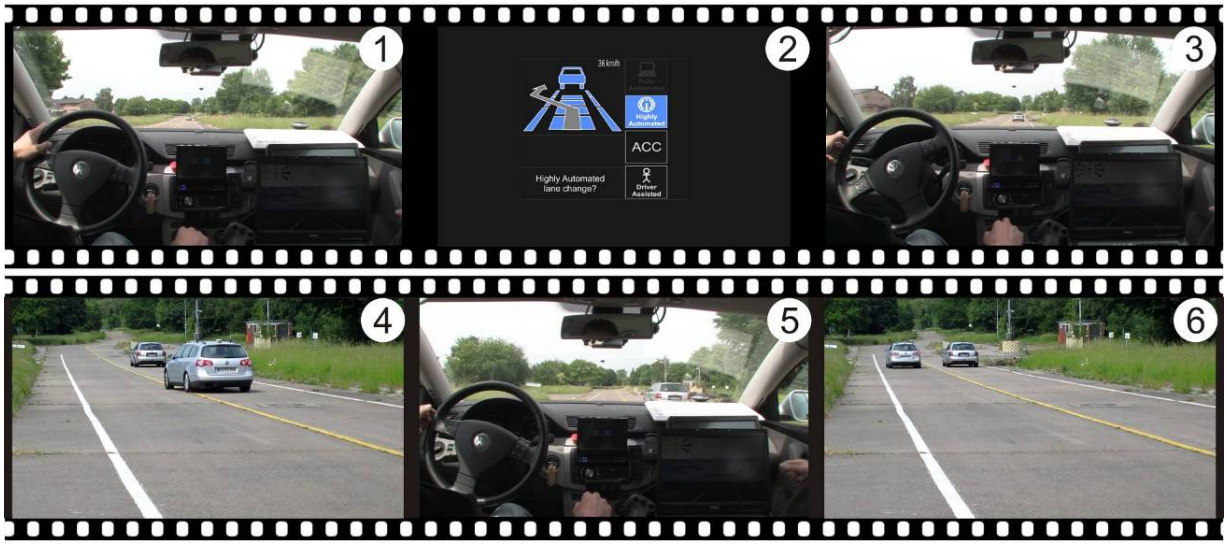


Figure 42: Driving and lane change: Video sequence

The video sequence in Figure 42 shows how the demonstrator vehicle approaches a standing lead vehicle in the right lane. The demonstrator vehicle drives in automation level Highly Automated. As the distance decreases, a lane change is suggested in the display shown in frame 2, which the driver activates by pushing the indicator button to the left, (frame 3). Immediately, the vehicle initiates a lane change (see frames 3 and 4) and then passes the standing vehicle (frames 5, 6) on the left lane.

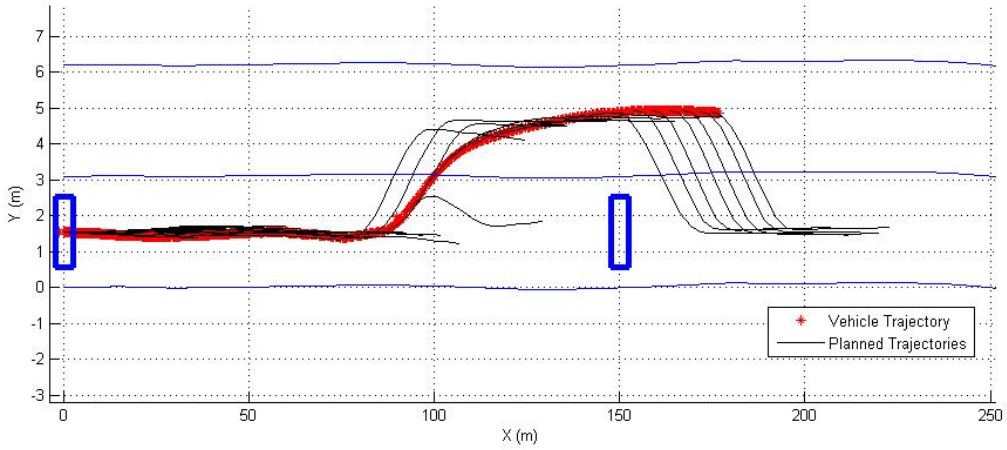


Figure 43: Driving and lane change: Planned and driven trajectory

Figure 43 shows the behaviour of the vehicle with respect to the planned trajectories. The two boxes show the initial position of vehicle demonstrator and the obstacle placed in the right lane, ahead of the initial position of the demonstrator. Note that the axes are scaled differently to allow observation of more details. At the beginning of this plot, the planned trajectories keep the vehicle in the middle of the lane. When the system detects the obstacle, the planned trajectories are deformed in order to bring the vehicle to the adjacent lane to the left to avoid the collision to the obstacle. When the obstacle is passed, the planned trajectories try to bring the vehicle again to the right lane (see at the end of the plot) but this lane change again requires confirmation of the driver (as the driver did not react, the vehicle did not change to the right lane again).

The use case “Driving and lane change” has been validated successfully, as described above. The lane change is currently initiated by an indicator button due to limited access to the indicator signals in the preliminary test vehicle. Once the Joint System will be transferred to the FASCar Joint System demonstrator, the indicator lever will be used.

2.6 Driving and Activation Not Possible

This use case class is about situations in which the driver wants to activate a specific automation level but this transition is refused due to unfulfilled preconditions, e.g. driving too fast or not on a highway. Examples that are covered are the refused transition from Driver Assisted to Semi-Automated or to Highly Automated as well as the refused transition from Semi-Automated to Highly Automated

In order to validate this use case class in the FASCar, the precondition of maximum speed for automation level Highly Automated was manually decreased in order to make this automation level unavailable during the test. In a later product, the maximum speed allowed for level Highly Automated of course might be higher. The driver tried to switch from Driver Assisted to Highly Automated, which was refused. Then the driver took over by selecting a lower automation level again. Meanwhile, the driver was following the right lane of a two-lane road at approximately 6.0 m/s.

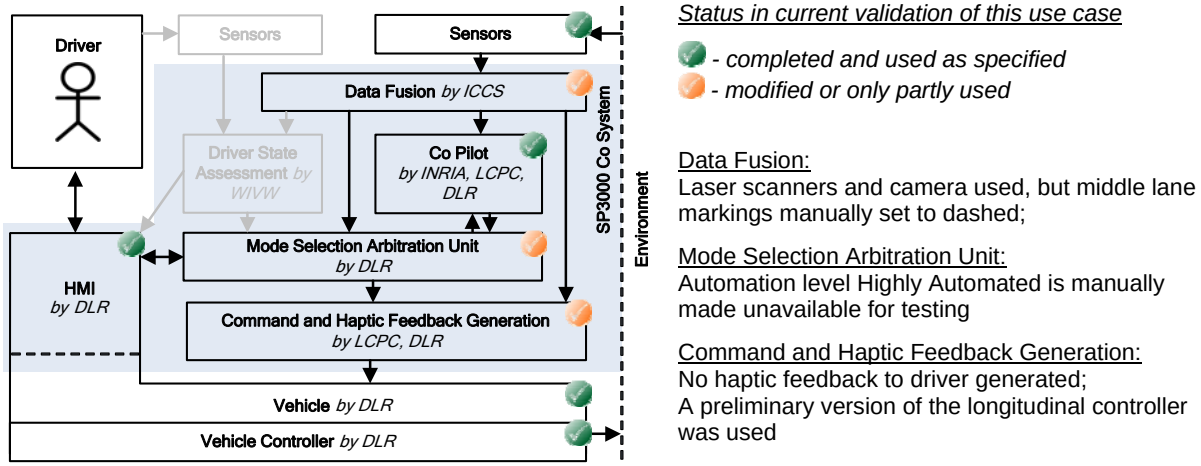


Figure 44: Driving and activation not possible: Used System Components

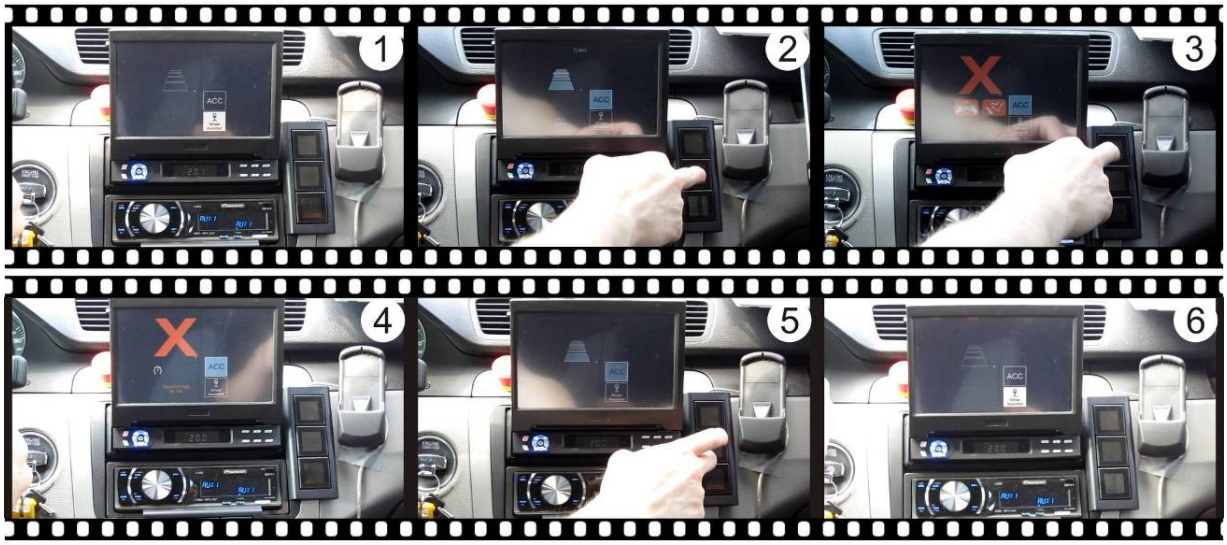
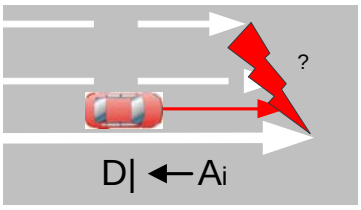


Figure 45: Driving and activation not possible: Video sequence

Figure 45 illustrates the HMI display when the activation of the automation level Highly Automated is impossible. First, the demonstrator vehicle is in Driver Assisted (frame 1). Subsequently, the driver switches to automation level Semi Automated (frame 2). When trying to activate Highly Automated (frame 3) the display indicates the refused transition by a large orange “X” and symbols to communicate the driver that he must remain in control. After this initial warning, the symbols below the “X” are replaced by additional information regarding the reason for the refusal (frame 4) – in this case that the speed is too high. In frames 5 and 6 it can be seen how the driver switches back down to Semi Automated and finally Driver Assisted.

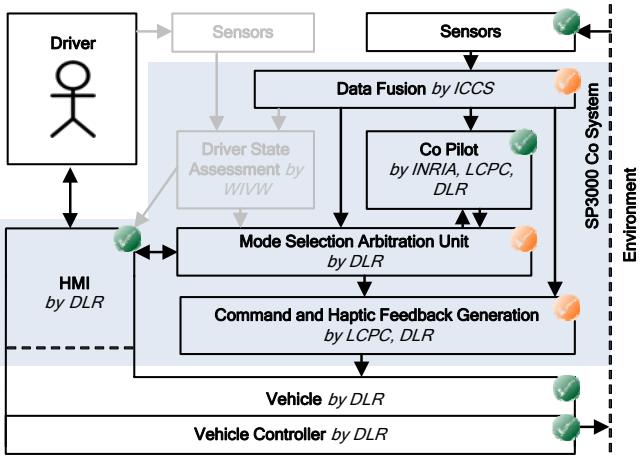
The use case “Driving and activation of automation level impossible” has been validated successfully for the case that the speed was too high to activate Highly Automated. The speed limit for this refusal was reduced for this purpose. All visual HMI components worked properly.

2.7 Driving, Driver Unresponsive and Transition to MRM



This use case class is about a situation in which the driver does not take over control as requested by the system. After a warning period the co-system starts a manoeuvre that is the manoeuvre with the minimum risk in that specific situation to avoid serious damage.

To validate this use case class in the FASCar, the vehicle was following the left lane of a two-lane road in automation level Highly Automated at approximately 6.0 m/s. This automation level was then manually made unavailable, which triggered an escalating take over request to the driver and ultimately led to a transition to the MRM. During the MRM the FASCar changed to the right lane, slowed down and then stopped completely.



Status in current validation of this use case

- completed and used as specified

- modified or only partly used

Data Fusion:
Laser scanners and camera used, but middle lane markings manually set to dashed

Mode Selection Arbitration Unit:
Highly Automated is manually triggered to be no longer available for testing

Command and Haptic Feedback Generation:
No haptic feedback to driver generated;
A preliminary version of the longitudinal controller was used

Figure 46: Driving, driver unresponsive and transition to MRM: Used System Components

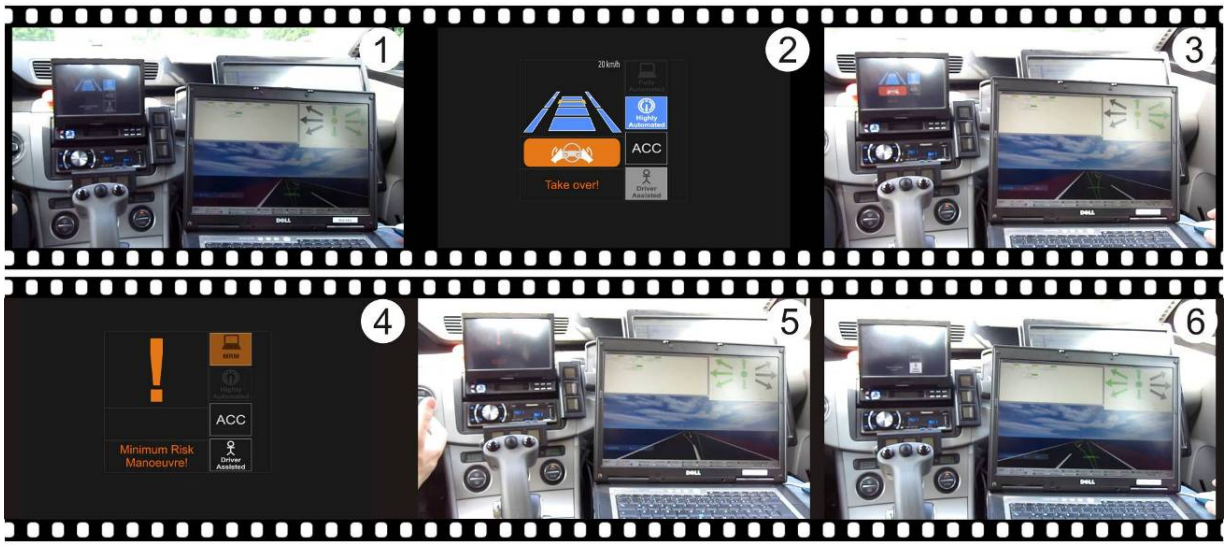


Figure 47: Driving, driver unresponsive and transition to MRM: Video sequence

Figure 47 shows the necessary deactivation of the automation level Highly Automated. First the demonstrator vehicle is driving in Highly Automated (frame 1). Then this level is no longer available and a take over request is issued, as shown on the display in frame 2 and 3. The take over request is escalating by blinking with increasing frequency. Since the driver does not react on the take over request, the minimum risk manoeuvre (MRM) is triggered, as shown in frames 4 and 5, and the field for Driver Assisted is still blinking to indicate the automation level requested by the take over request. The MRM is executed and the vehicle changes to the right lane, decelerates and finally comes to a complete stop. At this point, the driver takes over which terminates the MRM and the automation level Driver Assisted is activated (frame 6).

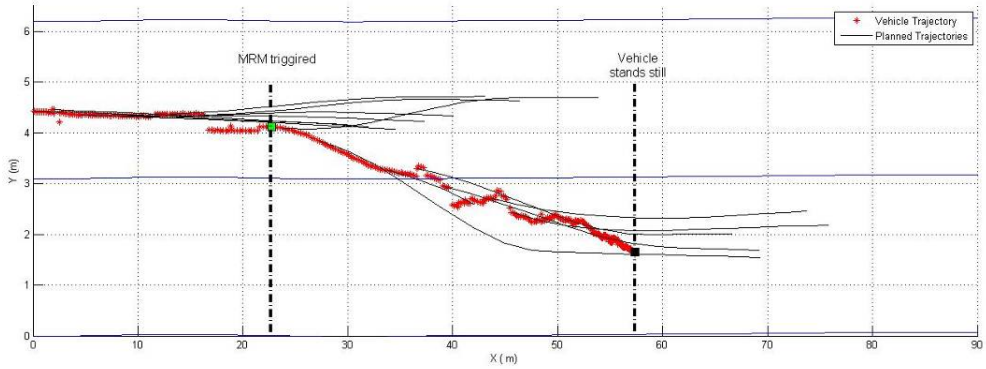


Figure 48: Driving, driver unresponsive and transition to MRM: Planned and driven trajectories

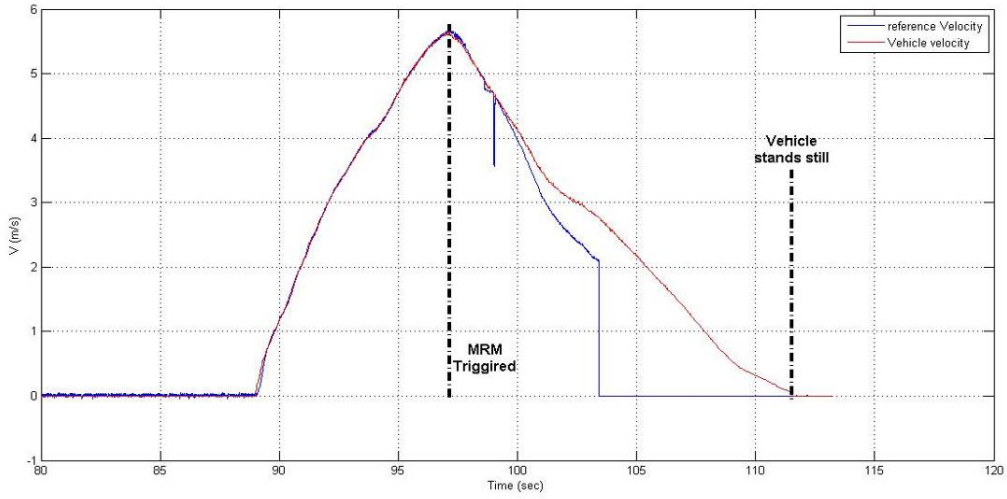


Figure 49: Driving, driver unresponsive and transition to MRM: Planned and real velocity

Figure 48 shows the planned (thin black lines) and actually driven trajectories (red stars) for this use case. The actual positions plotted in the figure stem from the recorded GPS values. At some points, the GPS data included jumps and other irregularities. These errors, however, only show up in the above plot and have no influence on the actual performance of the demonstrator

vehicle, since the GPS measurements are not used by the system and only exploited for the visualisation of the use case in this diagram.

Figure 48 shows that at the beginning, lane keeping is requested by the system and the co-pilot plans the adequate trajectories for this manoeuvre. The vehicle follows these plans and follows the left lane. After some time the MRM manoeuvre is triggered because the driver fails to comply with the take over request. The system commands the vehicle to change a lane to the right and then to decelerate to standstill.

In Figure 49 we see the profiles of the reference velocity and the real vehicle velocity. In a normal driving scenario, the reference is planned to reach its desired value and the vehicle tracks this velocity profile very closely. After triggering the MRM manoeuvre, the reference velocity is planned in a way to bring the vehicle to standstill. We observe that in the deceleration phase, the vehicle only follows the reference profile with a certain delay. This is because we are using the same controller to control the acceleration and deceleration dynamics, although they are different because of different actuators used to affect these dynamics (engine to accelerate and brakes to decelerate). We also experienced a peak artefact in the reference velocity, which is generated by the co-pilot. The peak did not have any effect on the vehicle behaviour, but we will nevertheless investigate this issue in the next integration phase.

A preliminary version of the use case “Driving and deactivation necessary (MRM)” has been validated successfully with manually triggered unavailability of the automation level Highly Automated. In future, the availability of the automation levels has to be determined automatically upon the fulfilment of the various preconditions.

3 Validation and Status Report of System Components

While Chapter 2 focussed on the overall result for each use case, in this chapter, the validation of the individual Joint System component is discussed in more detail.

3.1 System Hardware

The current system hardware setup was established in the first year of the project and has been improved and validated in the second year during many testing activities in an earlier military area which was used exclusively as drive-by-wire test track.

All system hardware components have reached a stable and reliable state, proved by several driving experiments (compare Chapter 0, Section 1.2). In the vehicle there are two different types of sensors and actuators. On the one hand there are the vehicle sensors which were already integrated in the serial car. On the other hand there are the sensors that were added to the car especially for HAVEit. In detail these are the laser scanners, the lane detection camera and the differential GPS system.

The main results of the sensor commissioning phases are:

Lane detection camera:

- Stable and satisfying precision regarding lateral deviation, heading and lane properties
- Lane curvature is precise enough in the range up to 50m, but in the wider range the curvature has to be estimated using another sensor (here digital map and GPS)

Laser scanner

- Several updates and adaptations to the HAVEit requirements were performed and tested
- The algorithms to distinguish between relevant and not relevant objects were improved under several aspects
- The basic software has been extended with lane recognition functionality
- The last testing phase demonstrated a satisfying tracking performance of the manoeuvre relevant objects

D-GPS

- In dependence on the current satellite configuration the system operates at a precision of 1-10 cm. By using a digital map the sensor data fusion is enhanced with the positioning signals, e.g. the street lane curvature is calculated in this way.
- The according reference station is a mobile unit which is ready for operation all over Europe.

All sensors are running in a real-time environment which provides the according environment perception to the Co-Pilot algorithms. An example for the synchronized sensor data is shown in Figure 50 where the vehicle heading in the lane is plotted for both, lane detection camera and GPS system. Both values are showing good accordance and a satisfying precision.

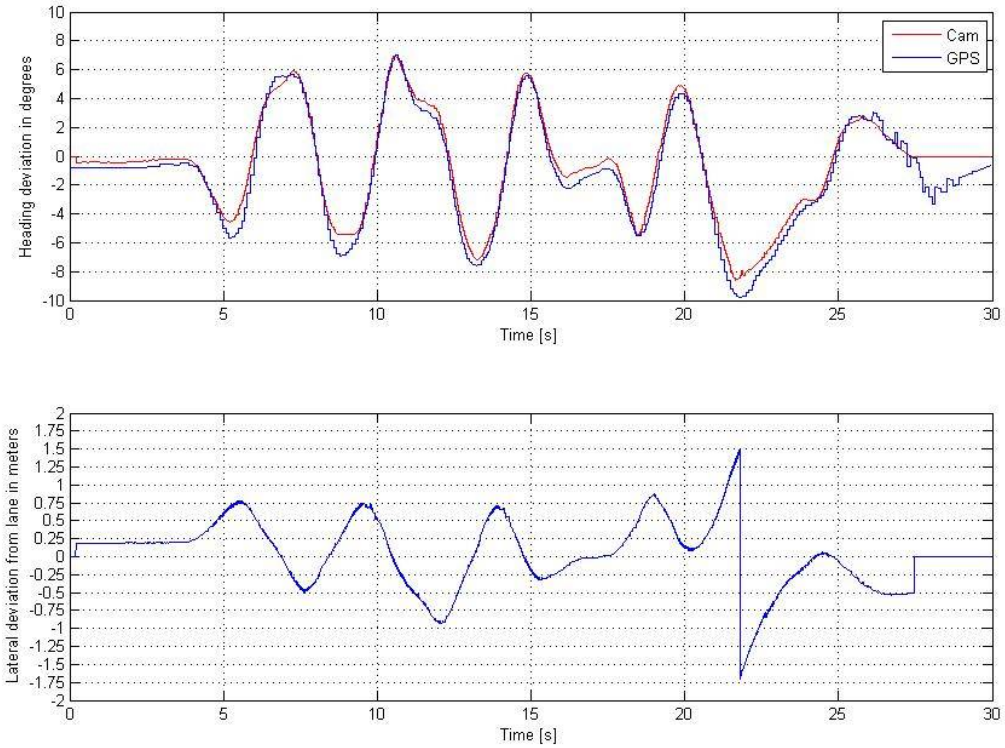


Figure 50: Heading of both, lane detection camera and DGPS (upper part). Heading of the vehicle in the lane (lower part).

3.2 Data Fusion

The data fusion component has been tested both separately and in conjunction with the other Joint System components. Object tracking performance was good and consistent allowing the Co-Pilot and control components to take proper action in the test scenarios. Also stationary objects (like grass) were properly filtered, so the vehicle did not react on erroneously detected objects. Lane estimation was performing well up to medium ranges and moderate dynamics. In hard manoeuvres the lane detection camera lost the lane tracking, but this error is minimized with proper tuning of the rest of the system. Filter components that were used for ego vehicle state estimation also performed well and did not reveal any abnormal behaviour during scenario testing.

Figure 51 shows the tracker output during the scenario of avoiding right overpass (see chapter 2.4) in the longitudinal (upper two diagrams) and the lateral dimension (lower two diagrams). We see that relative velocity increases to zero, and that the ego vehicle stops behind the standing object. Track ID of the standing object is maintained throughout the scenario until the ego vehicle comes to a standstill (Figure 52).

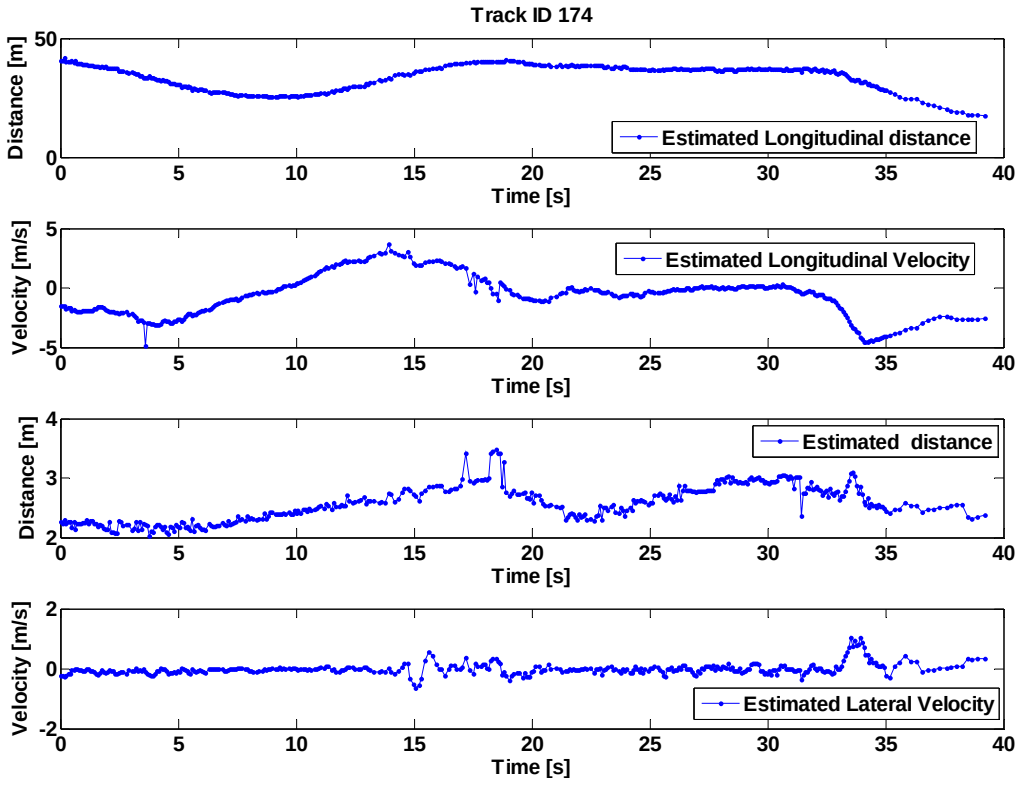


Figure 51: Tracked object estimation in right lane

Figure 52: Scenario: avoid right overpass

In Figure 53 track estimation of a standing object is presented. The standing vehicle is first detected at ~50m. The automation action can be seen after 5 seconds where the relative velocity increases until it becomes zero and the ego vehicle comes to a standstill (Figure 54).

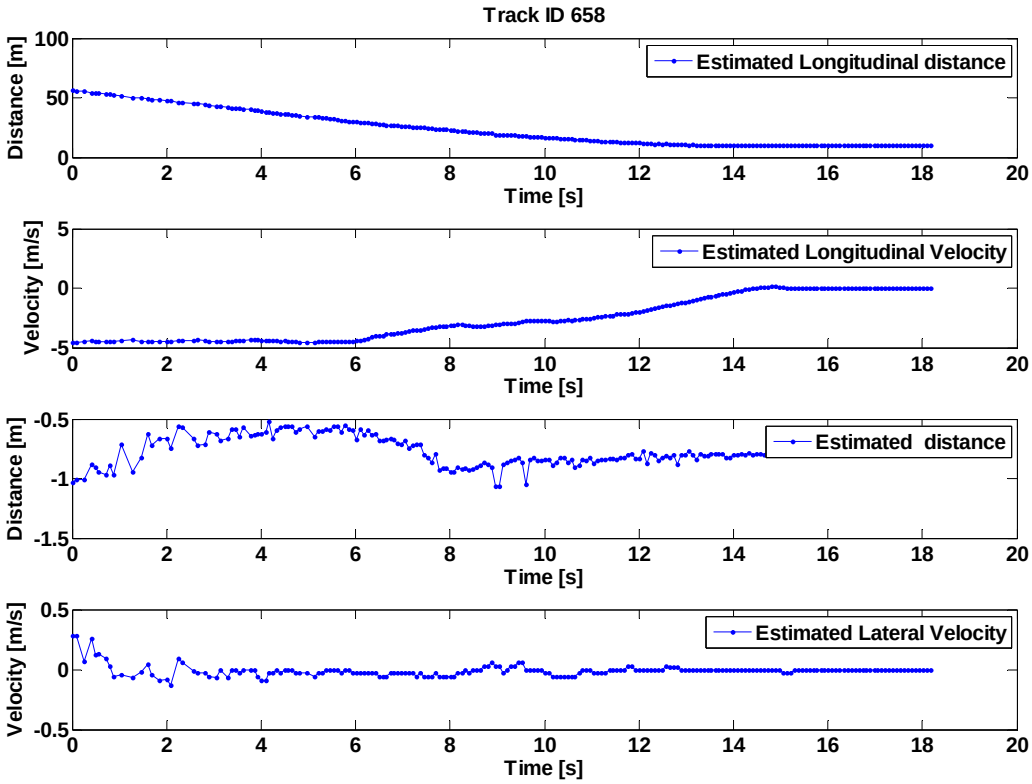


Figure 53: Tracked object estimation in the same lane

Figure 54: Scenario: Stopping behind object

In Figure 55 track estimation of a standing object is presented. The standing vehicle is first detected at ~70m. The automation action can be seen after 3.5 seconds where the lateral distance (3rd plot) decreases indicating a lane change manoeuvre (Figure 56). Notice that in contrast to the previous scenario, relative longitudinal velocity does not become zero, since the automation avoids the object with a lane change.

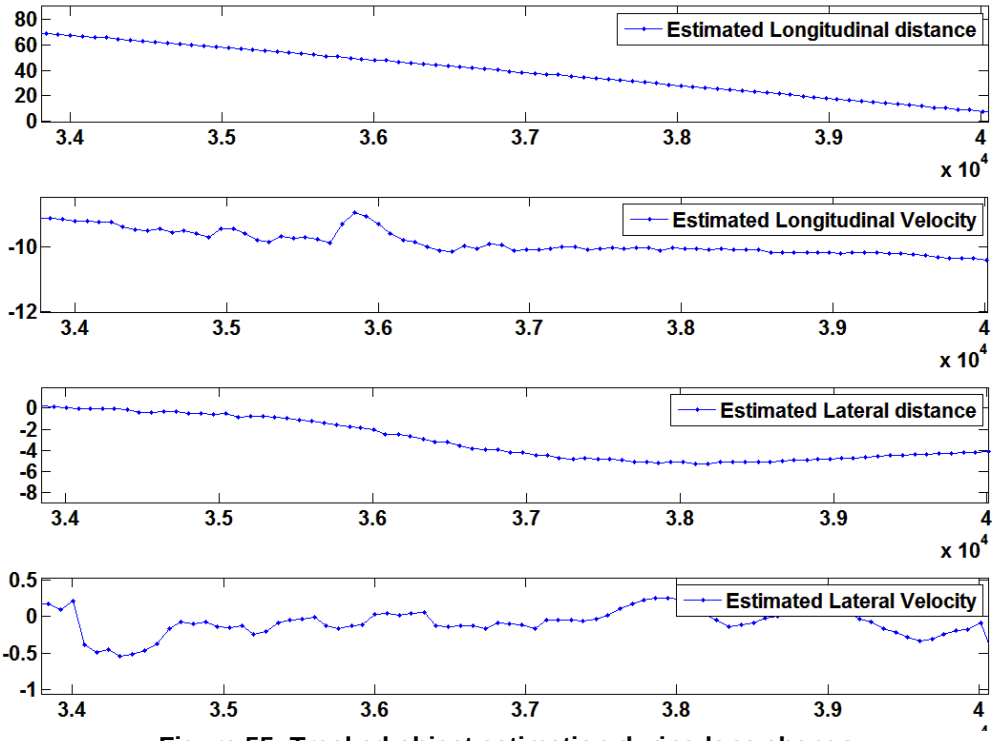


Figure 55: Tracked object estimation during lane change

Figure 56: Scenario Lane change

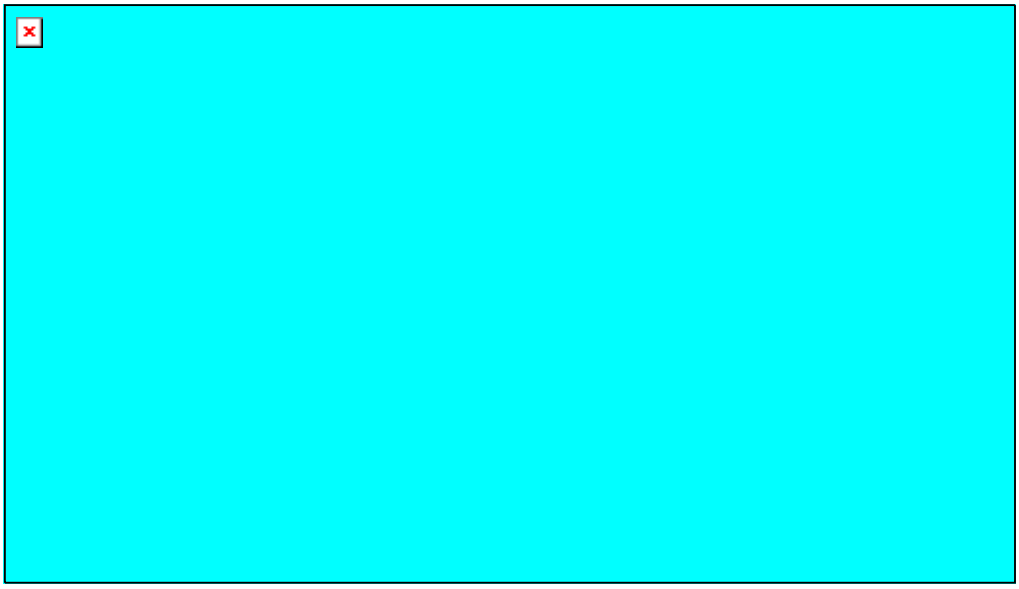


Figure 57: Lane estimation lateral error

Following topics will be addressed in the last year of the project.

- Lane estimation using the lane detection camera is not sufficient, so a fusion of camera, map and laser scanner data will be implemented in order to increase the reliability of the estimation in ranges above 50 meters.
- Another important issue is the sensors failure management. Appropriate strategies will be designed and implemented in cooperation with the other components. For example in the event of failure of lane detection camera, an appropriate strategy has to be designed depending on how fast the system can pass the control to the driver and how long the data fusion can provide accurate enough information.

3.3 Co-Pilot

The Co-Pilot has two main functionalities: The first functionality is the manoeuvre generation which contains several manoeuvre algorithms and the fusion of their outputs (see Section 3.3.1). The second functionality of the Co-Pilot is the trajectory generation (see Section 3.3.2) which, based on the current environment and the available manoeuvres, calculates one trajectory for every available lane. Further, a Minimum Risk Manoeuvre trajectory is always generated.

3.3.1 Manoeuvre Tree, Manoeuvre Grid, and Manoeuvre Fusion

As explained above (see Section 1.1.2) the manoeuvre planning component consists of three different components whose outputs are fused in order to generate a common behaviour. In the following the behaviour of each algorithm as well as the fused output in FASCar will be shown in detail for an exemplary driving situation. For this purpose, the use case "Driving and Lane Change" which is already explained above (see Section 2.5) is adopted. In addition to this detailed depiction of the behaviours in this situation some general comments are given regarding the current status of each component.

Manoeuvre Tree algorithm

The algorithm covers all situations which occur within the use cases which were defined for the Joint System. Further work will improve the behaviour of the algorithm so that it is more compatible to the driver. Also the general quality of the output of the algorithm will be improved by means of further tests in simulation and FASCar.

Figure 58 shows the output of the manoeuvre tree algorithm in the use case "Driving and Lane Change" as well as the output of the data fusion for some points in time. The plots in the lower part of the figure show the automation preferred manoeuvre and its longitudinal parameterization. The preferred manoeuvre is the one with the highest valential of the feasible manoeuvres in the manoeuvre tree. The whole manoeuvre trees for some specific points in time are also depicted. In the manoeuvre tree there is also the valential for each possible longitudinal parameterization of a manoeuvre depicted (A: "Accelerate", H: "Hold velocity", D: "Decelerate").

At the beginning the vehicle is driving along the right lane of the road. The automation preferred manoeuvre is "Follow Lane". Because the vehicle is slower than the target velocity (10 m/s) the longitudinal parameterization of the manoeuvre is "Accelerate". At time t =93.60s the output of the data fusion as well as the manoeuvre tree for this situation are shown. The root manoeuvre of the tree indicates the manoeuvre which is currently driven by the vehicle ("Follow Lane"). The other feasible manoeuvres are "Emergency Brake", "Change Lane Left" and "Minimum Risk Manoeuvre". The "Follow Lane" manoeuvre is the preferred one, because there is no reason for the automation to prefer one of the other feasible manoeuvres. The longitudinal parameterization of the automation preferred manoeuvre changes some time later to "Hold Velocity" because the target velocity is reached.

At about t = 95s the "Change Lane Left" becomes the automation preferred manoeuvre. This is founded in the detection of an obstacle in front of the ego vehicle. Because the velocity of this obstacle is slower than the target velocity of the ego vehicle (10 km/h slower at minimum), the automation wants to overtake this obstacle to hold the target velocity. As no obstacle is located on the left lane, the "Change Lane Left" manoeuvre is feasible and can become the preferred one. As preferred longitudinal parameterization "Decelerate" is chosen to avoid a longitudinal

collision with the obstacle ahead. For t = 97 s a screenshot of the data fusion output for this situation is given. The obstacle is located at the end of the known lanes. The manoeuvre tree for t = 97 s shows that also the "Follow Vehicle" manoeuvre is feasible, but less preferred than the "Change Lane Left" manoeuvre. Beside that the "Follow Vehicle" manoeuvre is recognized as currently driven manoeuvre by the vehicle and thus located in the root of the tree.

Based on the output of the manoeuvre planner the HMI gives the information to the driver, which the Co-Pilot prefers to do a lane change. Because of this information the driver activates the highly automated lane change at about 97.2 s (see Section 2.5). Thus the vehicle starts to perform this manoeuvre.

Short time later the obstacle disappears from the data fusion output although it has not moved in reality. The reaction of the manoeuvre tree algorithm is that the preferred manoeuvre changes to "Follow Lane" because there is no reason any more to prefer the lane change. For t = 98.07 s the data fusion output is depicted. This screenshot shows that the obstacle is missing and that the ego vehicle is currently doing the lane change manoeuvre. In the manoeuvre tree for t = 98.07 s the "Follow Lane" manoeuvre is the preferred manoeuvre, the lane change to the left is the second one. The "Lane Change" manoeuvre is correctly recognized as currently driven manoeuvre.

The highly automated execution of the lane change manoeuvre is not influenced by the loss of the object. This is based on the fact that the possibility to do a highly automated lane change is only depending on the feasibility of the manoeuvre not on the preference of the automation.

When the vehicle has passed the lane markings the preferred manoeuvre changes for a short time to "Change Lane Right" (t = 98.3 s). Because of the obstacle is not detected yet, the automation tends to go the rightmost lane to fulfil the traffic rules.

Some time later the obstacle is detected again (t = 98.4 s). The automation prefers now the "Follow Lane" manoeuvre to overtake the obstacle on the left lane. For t = 103.05 s the data fusion output and the manoeuvre tree are shown. Only the "Emergency Brake" and the "Minimum Risk Manoeuvre" are feasible in this situation. When the obstacle is passed the automation prefers the "Change Lane Right" manoeuvre to fulfil the traffic rules like mentioned above. Thus the "Follow Lane" manoeuvre is also feasible (see data fusion output and manoeuvre tree for t = 105.90 s) the Co-Pilot does not directly start the highly automated execution of the lane change. It is shown to the driver that the Co-Pilot prefers the "Change Lane Right" manoeuvre and the execution will start when it is activated by the driver.

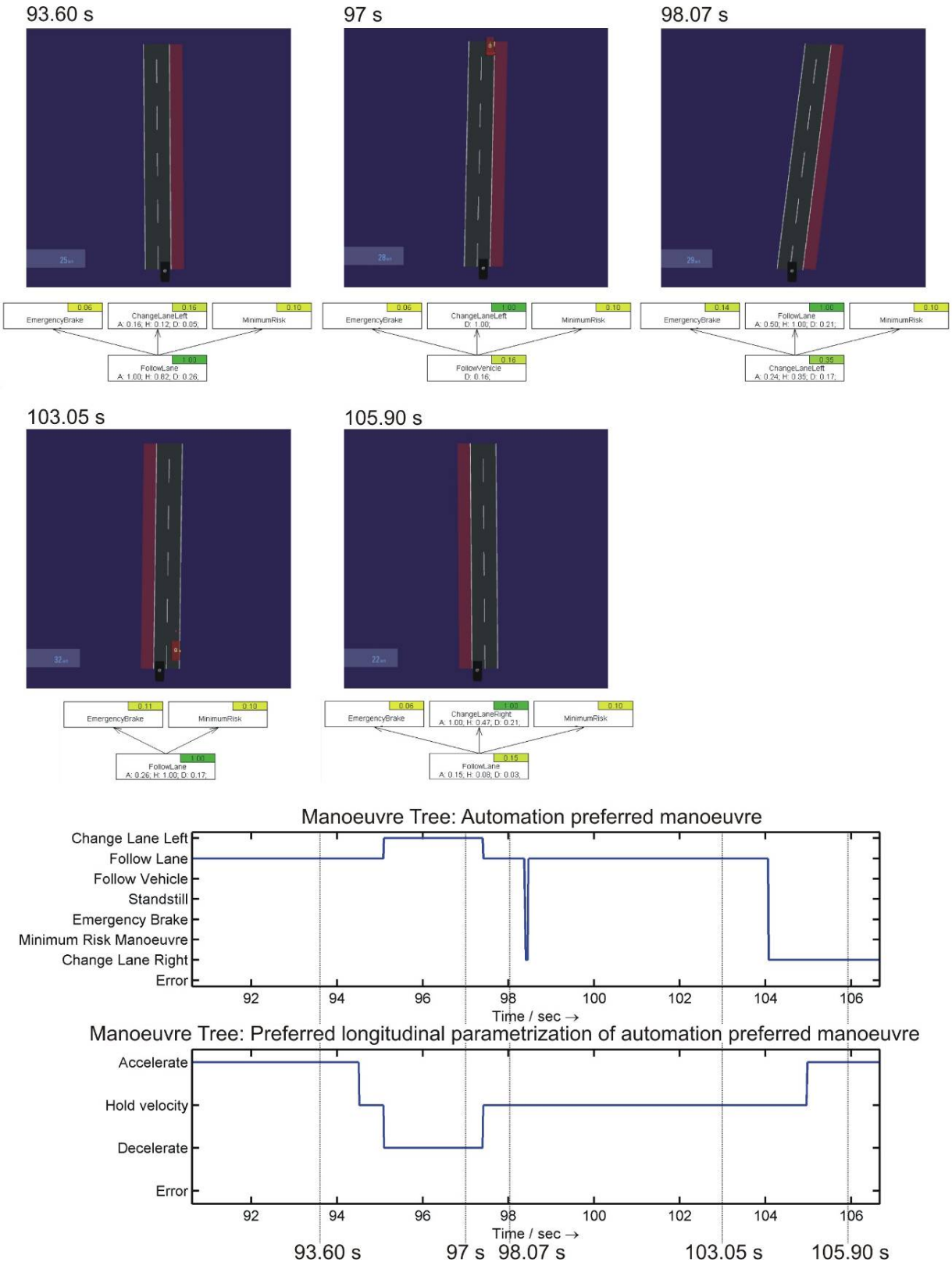


Figure 58: Output of manoeuvre tree algorithm in use case "Driving and Lane Change"

Manoeuvre Grid algorithm

The algorithm now integrates the full HAVEit functionality. The final year of development will be on the parameterisation of the algorithm to meet the needs of the different HAVEit demonstrators. Its structure will be made more universal to accelerate this parameterisation process and to open up to future applications.

Figure 59 shows a few snap shots of the environment and grid manoeuvre output for the “Driving and Lane Change” use case. At t = 93.60s the ego vehicle (black) is driving at 25km/h on the right most lane of the highway. There are no vehicles in the neighbourhood. The algorithm indicates that the manoeuvre corresponding with accelerating and staying on the lane (top - centre) is the best manoeuvre. It is attributed a maximal valential of 1 because it does not integrate any safety risk or traffic rule offence. The other manoeuvres have lower valentials. Changing lanes to the left is not recommended as traffic rules stipulate to stay on the right most lane whenever possible. Manoeuvres corresponding with a lane change to the right have a valential of zero as they mean driving off the road, indicated by the black arrows on the grid. The manoeuvres for holding velocity or decelerating are given a low valential because they are not optimal with respect to the average trip speed. The algorithm also indicates there is no reason to apply the emergency and minimum risk manoeuvres. At t = 97.00s an object (red) is detected in front of the ego vehicle. Now also the manoeuvres corresponding with staying on the lane have a zero valential, in black on the figure. The best solution proposed is now keeping the acceleration but changing lanes to the left in order to avoid the object. As this corresponds with a non-optimality with the previously mentioned traffic rule the valential has a value of 0.73, below the maximum.

At t = 98.07s the lane change manoeuvre has just been started when the object accelerates, It disappears from the horizon of sight of the ego vehicle and the manoeuvre grid is adapted, indicating again to stay on the right most lane, as previously at t = 93.60s. The object slows down and is overtaken by the ego vehicle at t = 103.05s. The manoeuvre chosen by the algorithm is to accelerate and stay on the lane. The ego vehicle speed is still far away from the legal speed limit and there is still a small penalty on the valential for not driving on the right most lane. Both going off-road and colliding with the object, respectively by a left lane change and a right lane change, are discouraged with a valential zero. At t = 105.90s the object has been passed. The optimal manoeuvre is now to change lane to the right and accelerate. This manoeuvre has a valential of 0.99. When changing lanes, the acceleration applied is a bit lower than while staying on the lane, for comfort and easier controllability reasons, which leads to a little penalty on the valential.

The lower part of Figure 59 gives a view on the evolution of the longitudinal and the lateral component of the best manoeuvre of the grid algorithm. The 5 snapshots described in this section are indicated on the time axes of the plots. Some changes of the best manoeuvre, not described by the snapshots can be spotted. At t = 95s, the object is detected for the first time. In the following seconds the algorithm hesitates between an emergency brake (staying in the lane) and accelerating (combined with a lane change to the left). This is an example for the need of parameterisation in the next development step. The weight of the traffic rules component (favouring the right most lane) and of the comfort and speed components within the algorithm (encouraging a lane change and acceleration) can be adapted so that overtaking is encouraged when there is no risk for a collision.

After t = 97.00s the obstacle stays during some seconds in the horizon of sight. After an acceleration, first a hold velocity and then a deceleration are proposed for changing lanes to the left. The algorithm adapts to the situation, as a lane change was not immediately executed by the human driver.

After t = 98.07s the vehicle went over to the left lane for just a moment. This changed a follow lane advice into a right lane change advice interrupting the acceleration with a slight braking to do so. The weight of the comfort component of the algorithm (penalising changes between acceleration and braking) could be adapted for a smoother longitudinal behaviour in this case.

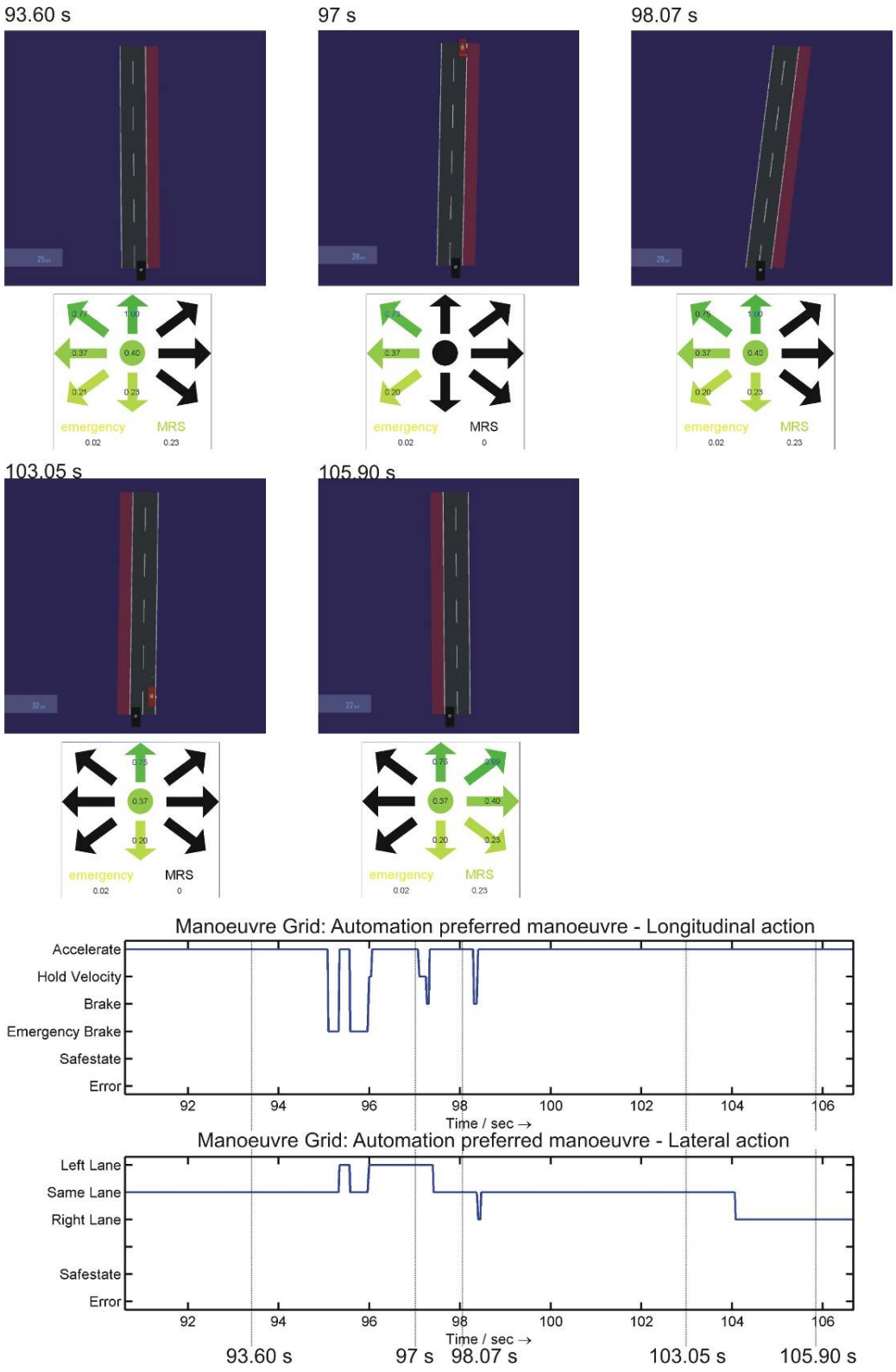


Figure 59: Output of manoeuvre grid algorithm in use case "Driving and Lane Change"

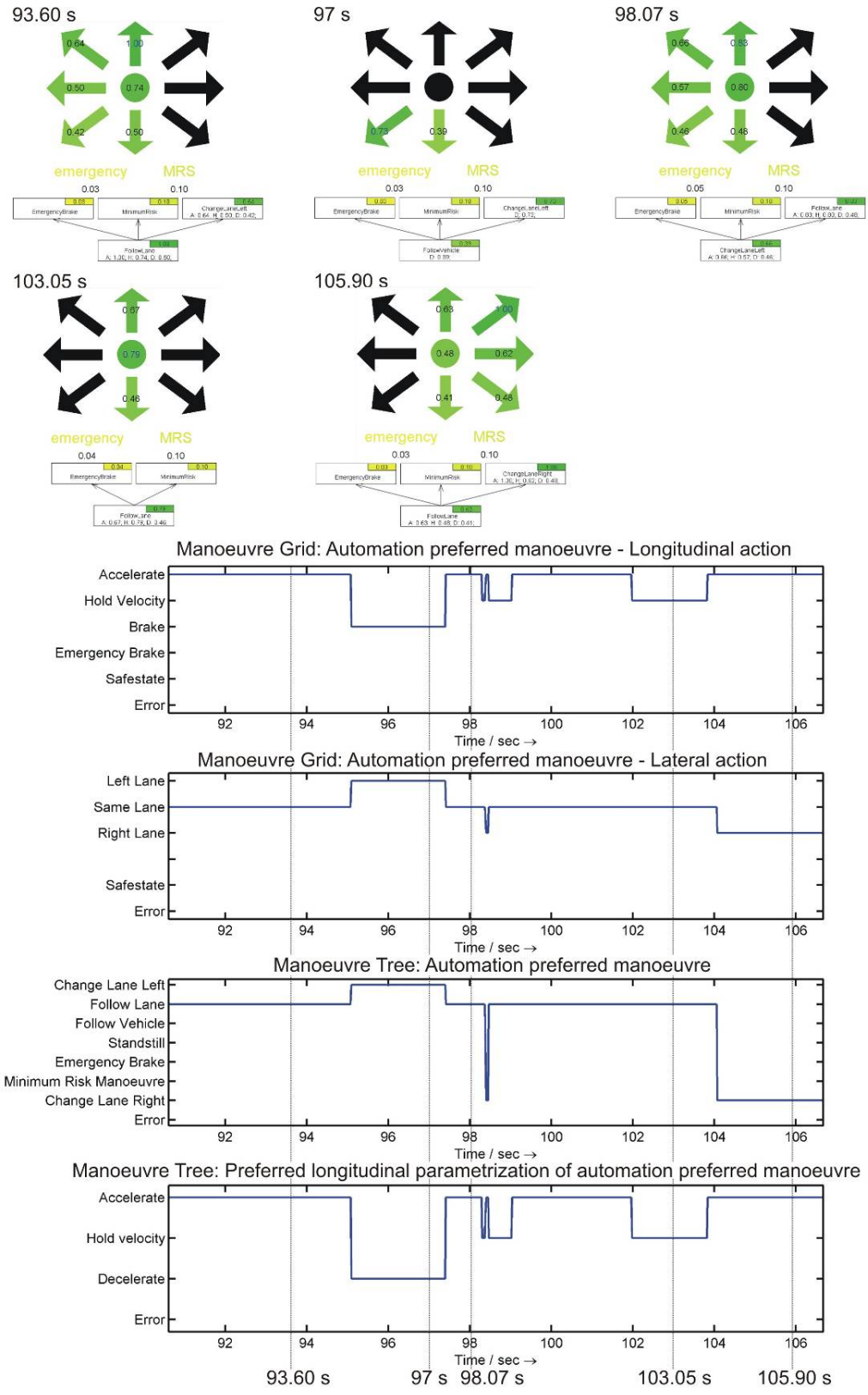


Figure 60: Fusion of the Manoeuvre Grid and Tree

Manoeuvre Fusion

Figure 60 shows the output of the fusion of the manoeuvre tree algorithm and the two manoeuvre grid algorithms. The fused output is depicted both as manoeuvre grid and manoeuvre tree. To do the fusion a congruency table is used. This table associates a tree manoeuvre to a corresponding grid manoeuvre.

As shown in Figure 60 the evolving of the preferred manoeuvre in the grid representation as well as the tree representation is the same. The output is stable and shows the same characteristic when the obstacle is lost as the three algorithms by themselves. Based on this figure, it can be concluded that the fused output is performing at least as well as the outputs of each single algorithm. Consequently, for all test runs presented in Chapter 2 the fused output was used.

3.3.2 Trajectory Generation

Several algorithms with a fundamentally different approach were developed for this function. All algorithms are using the same interactors (inputs and outputs), thus allowing easy exchange-ability in the development framework. A polynomial trajectory planner approach has proven to be more adequate for the HAVEit application during the validation in the FASCar demonstrator vehicle.

The polynomial trajectory planner uses a mathematical function that provides a geometric modelling (polynomial) of the vehicle trajectory that responds to the realistic demands of the manoeuvre to be performed. The advantage of this approach is that it is faster and easier adaptable to address different use cases.

The trajectory planner handles all the manoeuvres provided by the manoeuvre sub-component:

- Lane changes (left and right)
- Keep/follow lane or obstacle
- Emergency braking
- Minimum risk manoeuvre

The optimal trajectory is represented by a green line in the HMI (see Figure 61). The generated trajectories converge to the centre of the target lane and in general do not present overshoots for the addressed highway scenario and the use cases presented before.

The situations where an overshoot might occur are the following:

- When the vehicle is far from being aligned with the lane and the vehicle has high speed. For example: If the vehicle orientation relative to the lane is equal to 0.6 radians and the vehicle has a speed of 20m/s then an overshoot (slight) will occur even if the road has no curvature.
- When the road curvature is big and the vehicle has high speed.

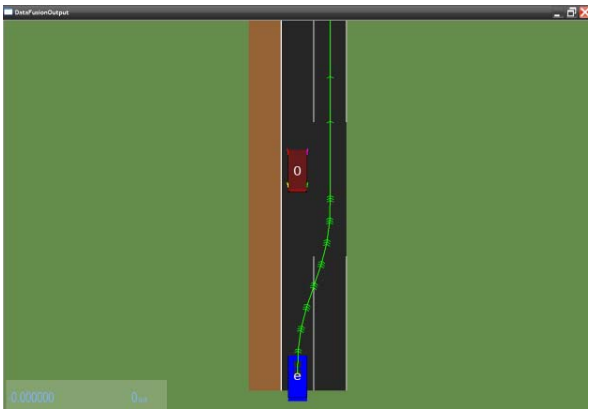


Figure 61: Change lane to the right to pass a stopped vehicle

The minimum distance to the obstacle in front can be adjusted by setting its value in the trajectory planner configuration file. At this moment this minimum distance to the obstacle in front is set to 17 meters. This value is high because of the following reasons:

It's necessary to have space to overtake the vehicle in front.

The lane detection by the camera doesn't work when the vehicle orientation relative to the lane is bigger than 0.26 radians (around 15 degrees). The lane detection doesn't work when the vehicle is close to the vehicle in front (less than 2 to 3 meters). Quick lane changes create oscillations in the lateral control.

The speed profile (see Figure 62) used can vary between a trapezoid (in red) and a sigmoid (in green) shape by using a sigmoid function as a filter which weight parameter can be set from a configuration file. The problem of using a simple trapezoidal speed profile is that the associated acceleration profile is not continuous. By using a sigmoid the acceleration profile is smoothed and high variations are eliminated. This improves the longitudinal control task.

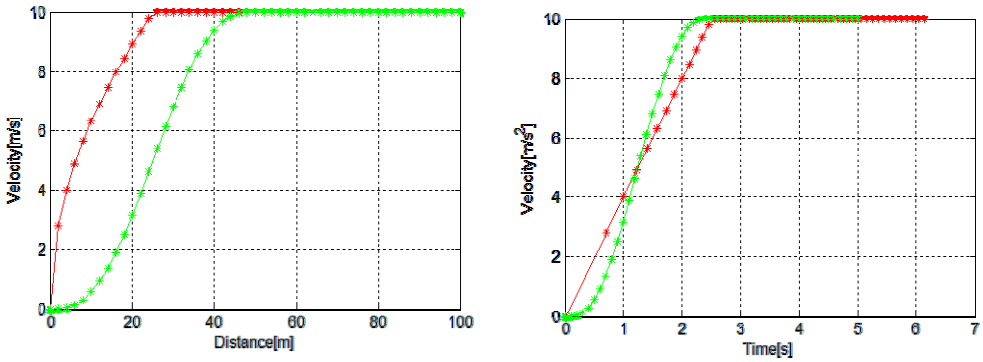


Figure 62: Speed profile

In Figure 63 the obtained trajectories for three different speeds (blue=5m/s, green=10m/s and red=20m/s) are represented for a typical lane change with a lateral displacement of 3.5 meters (lane width).

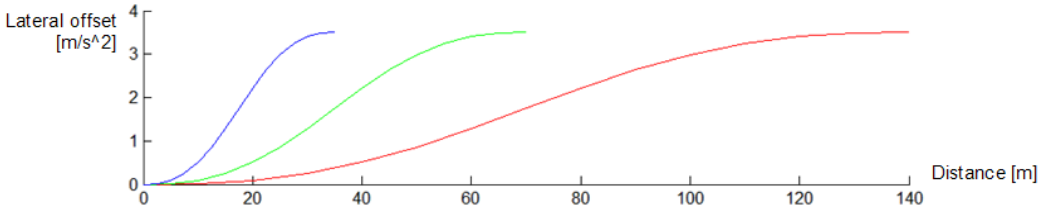


Figure 63: Lane change trajectories for different speeds

The value of the maximum lateral acceleration can be modified by changing the input parameters of the polynomial trajectory planner.

Figure 64 represents the lateral acceleration in m/s^2 corresponding to the trajectory at 20 m/s (red) presented in Figure 63.

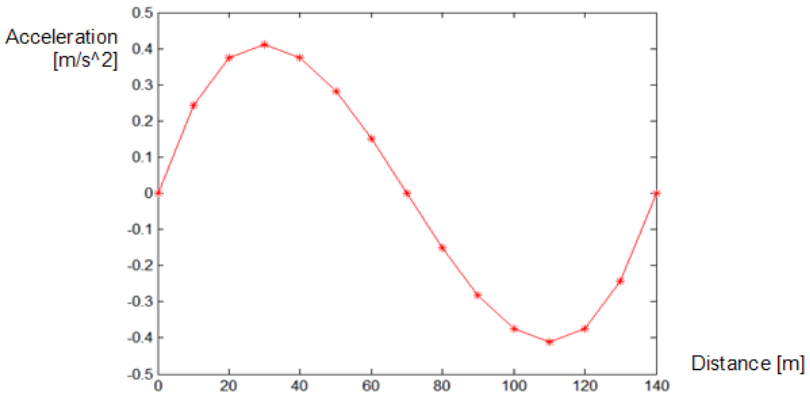


Figure 64: Lateral acceleration for a lane change at 20m/s

After a target lane is reached and the vehicle is over the lane and aligned, the remaining lane points are used to complete the trajectories.

3.4 Driver State Assessment

The current version of the DSA component is release number 3.0. It is available as an ELF file which can be flashed onto the CSC controller hardware. In addition, the DSA can be integrated into systems running on WIN32 and LINUX platforms via shared or static libraries.

So far, the DSA has been integrated successfully into the driving simulator of WIVW (CSC and WIN32 platforms), the VTEC demonstrator truck for WP5200 (CSC platform) and the Conti demonstrator vehicle for WP5100 (WIN32 platform). Integration work for the DLR demonstrator for WP4100 (CSC platform) is ongoing.

The following tests and validations have been performed during the development of the component:

- Study “DSA validation study” in the WIVW driving simulator: The intention of this study was to validate whether the first version of the DSA was successful to detect driver distraction and drowsiness in the driving simulator. 12 test drivers participated in the study. The study part that investigated distraction revealed important results for the further development of the distraction algorithms. It turned out that the definition of distraction has to be classified less rigorous as it was originally and that the thresholds for defining distraction have to be adapted to the current driving task demands and the automation level. Furthermore, it turned out that the DSA component is able to reliably detect different stages of drowsiness by both direct and indirect measures. Some modifications of the algorithms seem very promising in order to further reduce false alarm rates. The results from these simulator studies were considered in the subsequent versions of the DSA software.
- Integration tests in the WP5100 and WP5200 demonstrator vehicles: The goal of these tests was to validate whether the DSA component works correctly, whether all input signals are received properly, all internal calculations are done in the expected way and whether the DSA delivers meaningful output. The tests were performed in real traffic with a test driver following the instructions of a test leader sitting beside the test driver as a co-passenger. Besides observing the output of the driver monitoring camera, several specific driving manoeuvres had to be performed by the test driver to simulate an “abnormal” driving behaviour that might occur if the driver gets drowsy. The tests showed that the DSA correctly identifies relevant manoeuvres, correctly calculates respective internal signals and delivers meaningful output for drowsiness and distraction.
- Study “Attention Monitor” in the WIVW driving simulator: This study is currently ongoing. Its main goal is to define an interaction strategy for the use cases “driver drowsy” and “driver distracted” that will be carried out by the MSU based on the outputs of the DSA component. This will include information and warning messages as well as transitions to other automation levels. Beside this question the study will also be used to evaluate the newest version of the DSA software with regard to the detection quality for distraction and drowsiness

Further information on the DSA component can be found in the following HAVEit deliverable D32.1 [4] and D32.2 [5] as well as in publications [15], [16].

3.5 Mode Selection and Arbitration Unit

The current version of the Mode Selection and Arbitration Unit (MSU) component is 0.9. This version is developed in C-code and is available for the software framework and as ELF file for the CSC-ECU (see chapter 1.2.8).

During the validation at the FASCar the MSU runs within the software framework and manages the automation levels that are available in the current version (DA, SA, HA, Emergency, MRM, OFF), the transitions in between, as well as the visual HMI (Figure 13).

Figure 65 shows the current version of the MSU state machine as UML diagram. Automation level and transitions which are currently not preformed by the MSU are dyed in gray within the diagram. A detailed description of the MSU functionality is available in HAVEit deliverable D33.5 [7].

Next to the automation levels and transitions the MSU controls the HMI of the Joint System. During the validation only the visual HMI was active in the FASCar (see particularly Use Case “Driving and deactivation necessary” in Section 2.7). The acoustic and haptic HMI are also available by the MSU but not tested during the validation at the FASCar. A general validation of the MSU and the HMI was done by a usability study for HAVEit deliverable D33.3 “Validation by Simulation” [6]

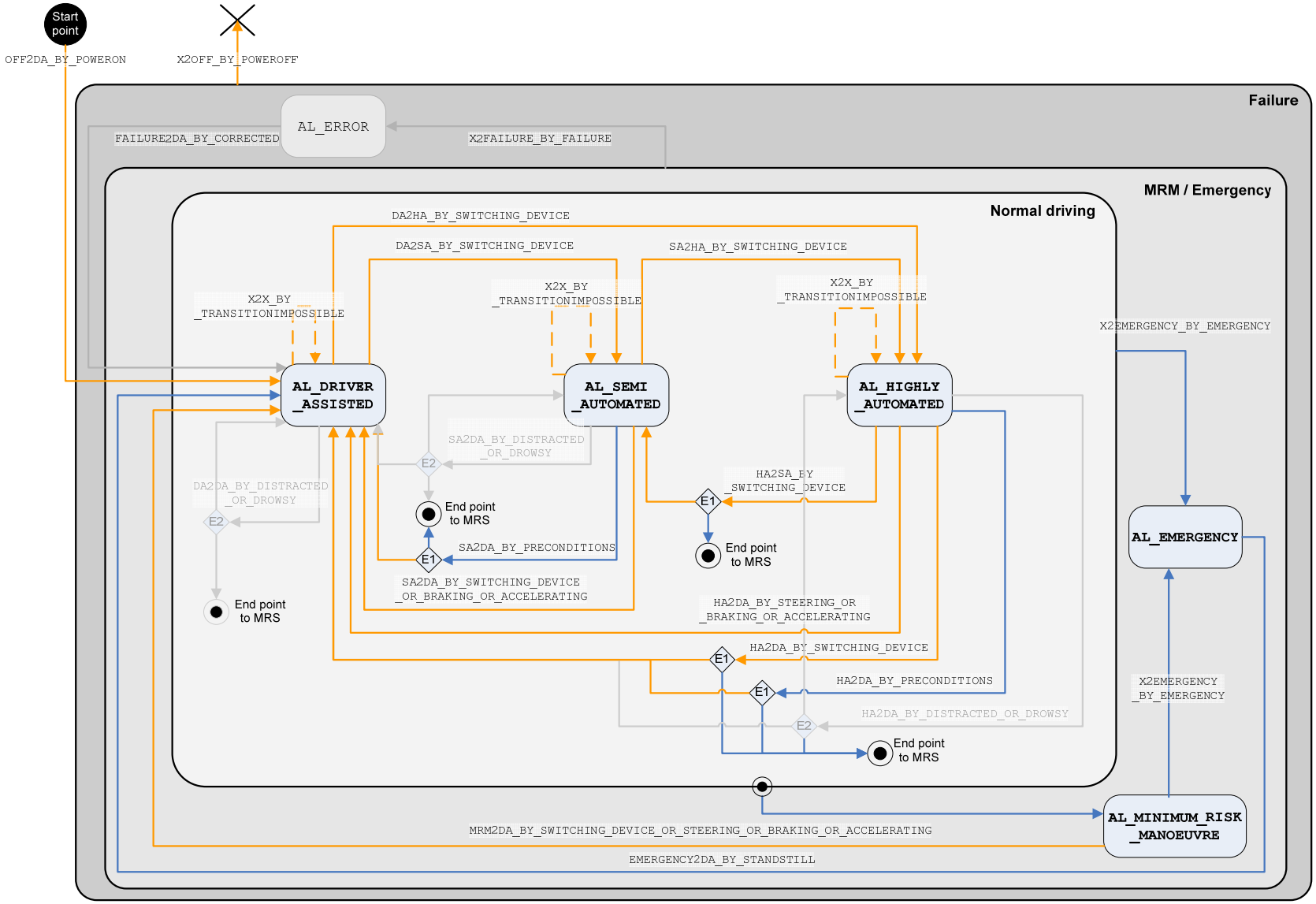


Figure 65: UML diagram of the current version 0.9 of the MSU state machine

3.6 Command and Haptic Feedback Generation

The component Command and Haptic Feedback Generation (see Figure 66) is comprised of two main functions: The Command Generation generates the acceleration and curvature that is necessary in order to follow the given trajectory. The second main function is the Haptic Feedback Generation, which generates the haptic HMI.

The Inceptor Input Translation translates the pedal and steering wheel inputs from the driver to an acceleration and curvature. Depending on the automation level input from the MSU, the Coupling Valve manages the lateral and longitudinal access of the driver and the automation to the vehicle.

The following figure shows the principal architecture of the component. A more detailed architecture description can be found in deliverable D33.5 [7].

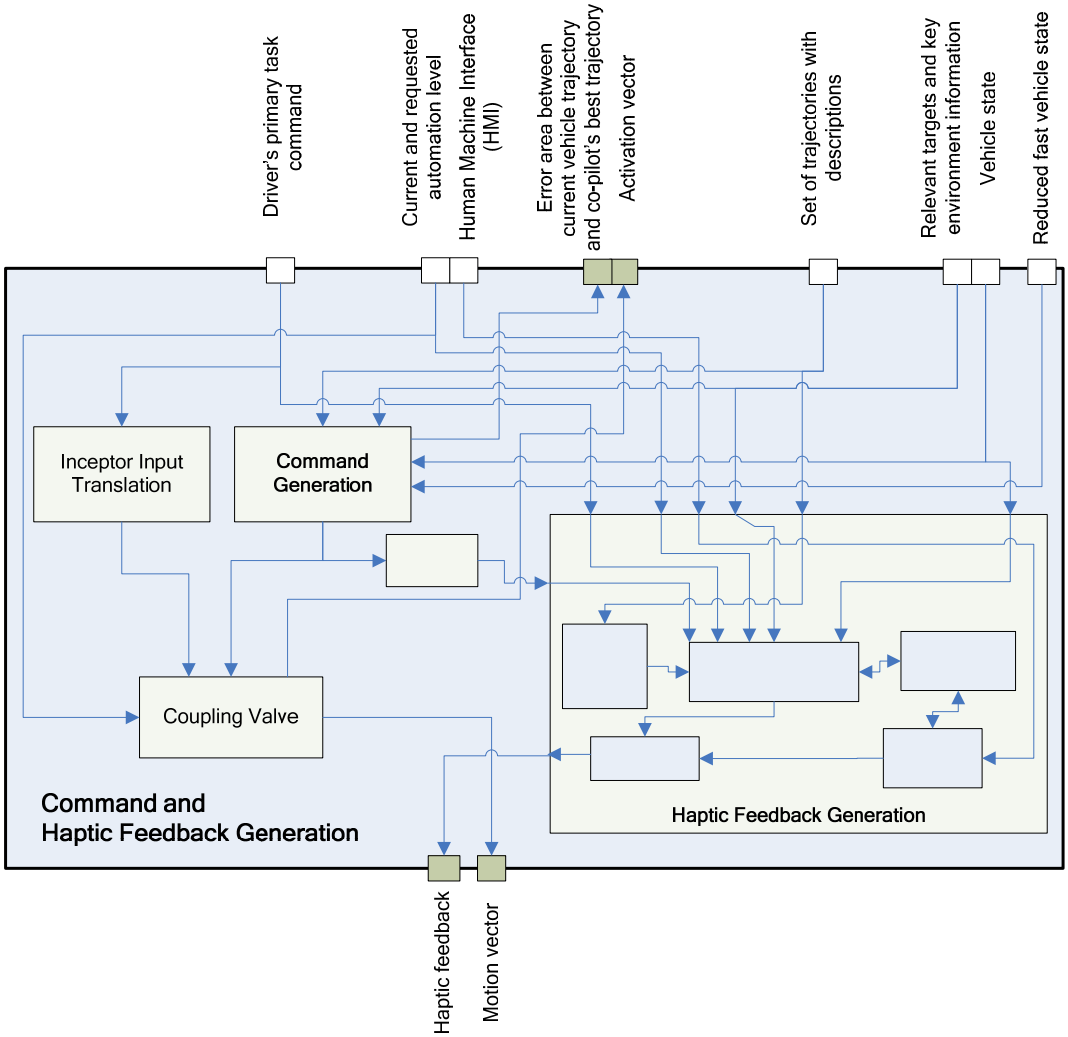


Figure 66: Principal architecture of the Command and Haptic Feedback Generation component

3.6.1 Command Generation

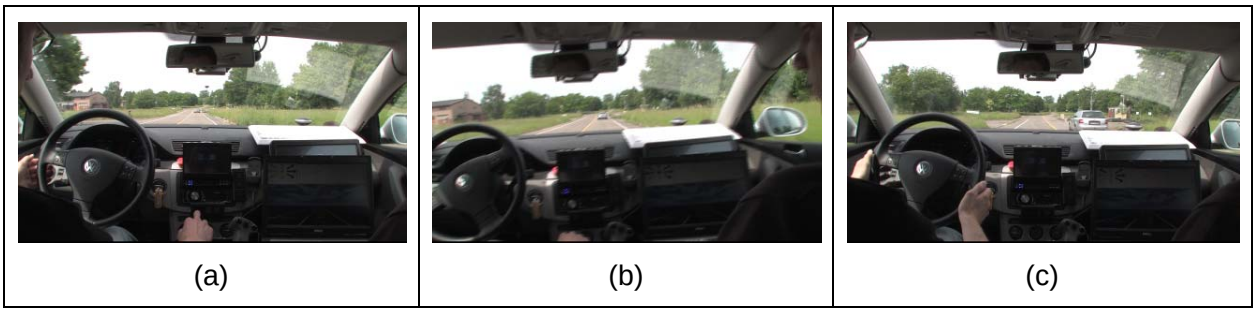
The main challenge of the command generation and validation sub-component is to handle the vehicle's dynamics. As mentioned in Section 1.1.5, the H-infinity controller design approach is chosen to develop the lateral controller. Regarding the complexity of vehicle dynamics, single H-infinity controller cannot handle the vehicle's lateral behaviour with the same performances level for the whole velocities range. For the actual controller development status, three controllers are designed around velocities 10, 15 and 20 m/s. However, a linear interpolation blending technique on controllers' outputs is used for controllers sequencing.

From trajectory data, the command generation and validation sub-component computes the lateral distance and heading errors from a point located at a constant distance ahead of the vehicle to the closest point of the trajectory. The controller uses these errors to correct the position and heading of the vehicle.

Besides, as the trajectory is generated at relatively low frequency, the command generation and validation sub component has an algorithm that allows the estimation of the vehicle position and orientation with respect to the first point of the trajectory. This component is well integrated but actually is disabled because it favours the apparition of controller oscillations. This issue will be investigated in depth.

In the next section we present an experimental test to validate the performances of the lateral controller which have a direct impact on vehicle stability.

In order to evaluate the performances of lateral controller, driving and lane change use case, described in Section 2.5, is used with controller (designed around 15 m/s). In this scenario, the vehicle starts on the same lane where a static obstacle is present. In the beginning, the vehicle keeps the lane because the perception component can't detect the presence of the obstacle yet, Figure 67-a. When approaching the obstacle, the perception component detects its presence. Then a lane change manoeuvre is initiated (Figure 67-b) to bring the vehicle to the adjacent lane (Figure 67-c).



(a) (b) (c)

Figure 67 : Lane change manoeuvre test on FASCar: (a) before lane change manoeuvre, (b) lane change manoeuvre engaged, (c) end of lane change manoeuvre.

Figure 68 sketches the velocity profile for this test case. It is noticeable that the choice of this low velocity is constrained by the test context, narrow track and presence of real obstacle. In deliverable D33.5 [7] we have reported test results without real obstacle where the lane change decision is controller directly from the control panel. In this test we have reached a velocity of about 20.5 m/s.

Figure 69 shows the lateral controller output. We observe at the first time that the control action is still constant up to 15s. This is caused by the fact that at the beginning the Joint System is running without connecting controllers' outputs to the vehicle actuators which can't bring the state of the system to change. When controllers are connected to the vehicle's actuators, the controllers affect the vehicle state and bring the vehicle to stay on the lane with lateral heading errors of about 0.2 (m) and 2 (deg) respectively, see Figure 70 and Figure 71. It is to be noticed that these errors are computed at a point located at a constant distance ahead of the vehicle (look ahead point).

When perception model detect the presence of the obstacle, the system engages a lane change manoeuvre which is accompanied with an abrupt change of the trajectory causing a discontinuous variation in vehicle to trajectory lateral and heading errors. Based on these errors, the controller computes the adequate control action in order to reduce this gap. It is interesting from Figure 69 to see that the controller is able to filter this critical variation and generates a smooth control action profile necessary for passenger comfort. The passenger comfort is measured from the lateral acceleration and should be, in general, less than 0.2*g in normal driving situation. Figure 72 shows the lateral acceleration factor (ay/g) for driving and lane change use case. It is interesting to see that the maximum value of the lateral acceleration doesn't exceed 0.1*g for this test use case, which is a good level for passenger comfort.

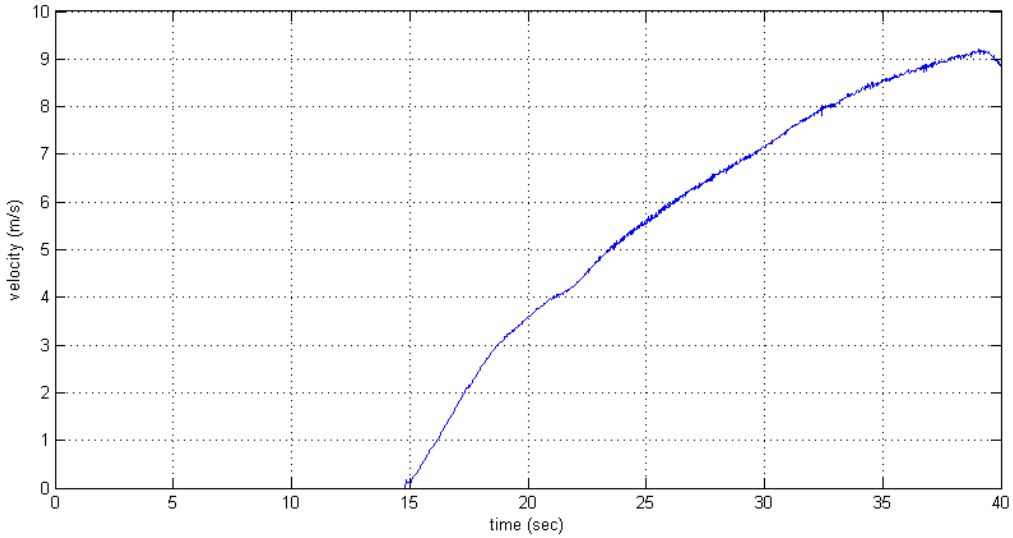


Figure 68: Vehicle velocity profile

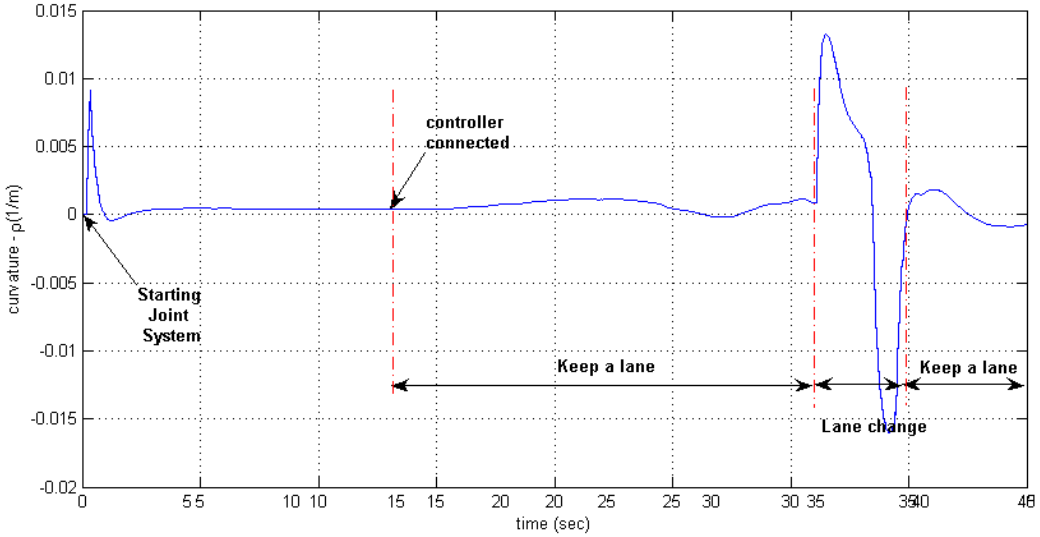


Figure 69: Lateral controller output expressed in terms of curvature

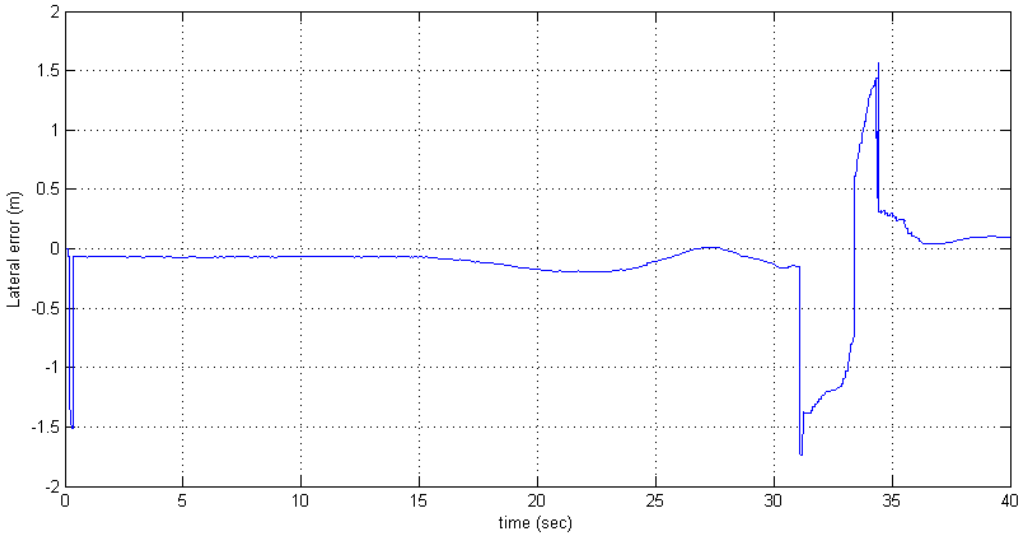


Figure 70: Vehicle look ahead point to trajectory lateral error

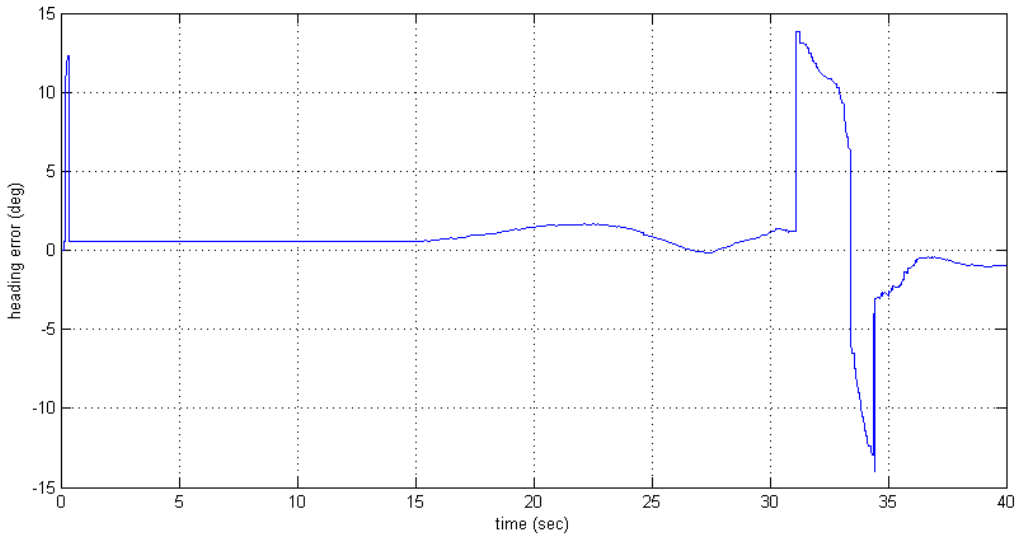


Figure 71: Vehicle look ahead point to trajectory heading error

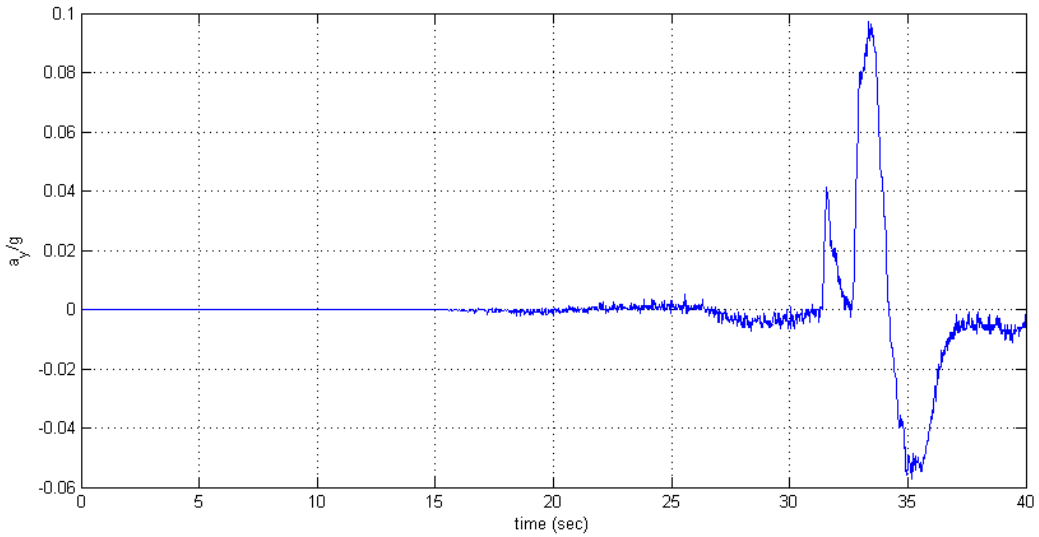


Figure 72: Lateral acceleration

3.6.2 Haptic Feedback Generation

At the current version of the Co-System component Command and Haptic Feedback Generation only a basic version of the Haptic Feedback Generation functionality is available. This means that the general software structure in C is already available and that some haptic feedback elements are already developed but not parameterised. The main integration of the Haptic Feedback Generation functionality will start this autumn as soon as the steer-by-wire system in the FASCar is available for haptic feedback tests.

4 Summary and Conclusion

This deliverable describes the validation of a research prototype system of highly automated driving in a test vehicle, one year before the end of the project. In the integration workshop in June 2010, all Joint System components (see Figure 2) except for the Co-System component Driver State Assessment were integrated and operated together in the test vehicle. During the validation the most relevant HAVEit use cases were tested in a real-world environment. The highest priority was assigned to the automation level Highly Automated because of the highest functional complexity of this automation level. Further, the same algorithms, functions and sensors are used as well in the other normal automation levels Driver Assisted and Semi Automated, even though in these levels the support of the driver is reduced compared to Highly Automated.

One year before the end of HAVEit the results of the first validated use cases (see chapter 2 and chapter 3) show that there are still many aspects to improve, but overall the Joint System is able to handle each of the use cases quite well.

- The environment sensors (lane camera and laser scanners) are available and deliver data which allow driving the use cases described in chapter 2.
- The Data Fusion component of the Co-System successfully generates a perception model and a vehicle state as input for the other algorithms within the Co-System.
- The Co-Pilot component as part of the Co-System interprets the output of the Data Fusion and chooses situation-specifically the available manoeuvres for the automation. By the chosen manoeuvre the Co-Pilot then calculates trajectories for all detected lanes and for the MRM.
- The Mode Selection and Arbitration Unit as component of the Co-System manages the transitions between the automation levels and controls the visual HMI.
- The Command and Haptic Feedback Generation of the Co-System follows the best trajectory generated by the Co-Pilot. In the automation level Highly Automated the driver is able to remove his hands from the steering wheel. Alternatively, the driver is able to keep his hands on the steering wheel to feel the lateral automation behaviour in HA. The driver is also able to influence the automation behaviour in HA via the steering wheel.

The overall concept and architecture hold up nicely to its expectations. The transfer of know-how and algorithm from the horizontal demonstrator described here to the vertical demonstrators is on its way and right on track. Besides the support for the vertical demonstrators that are now integrated, we will continue with the iterative refinement of concepts and prototypes: Until the end of HAVEit (July 2011) three additional vehicle integration meetings are planned. Between the planned integration meetings further development work will be performed in parallel at the partner sites. The main tasks during the next integrations are:

- The Co-System will be transferred from FASCarI to FASCarII
- Components of the Co-System will be migrated from the automotive PC to the CSC-ECUs.
- The Joint System demonstrator will be repeatedly tested and validated as the implementation and integration of improvements continues. Among others, coming validations will focus on the improvements to the Joint System sensors, and the validation of more complex use cases and higher speeds.

Overall, the subproject “Joint System” and the HAVEit project are on track. Highly automated driving is no longer a theoretical vision, but can be experienced now beyond simulators also in real vehicles.

References

[1] HAVEit Deliverable D12.1 “Architecture“, February 2009

[2] HAVEit Deliverable D21.1 “CSC & XCC HW-development completed”, July 2009

[3] HAVEit Deliverable D22.2 “Communication Hardware“, January 2010 2009

[4] HAVEit Deliverable D32.1 “Report on driver assessment methodology”, April 2009

[5] HAVEit Deliverable D32.2 “Model of driver behaviour for the assessment of driver’s state”, December 2009

[6] HAVEit Deliverable D33.3 “Validation by simulation”, November 2009

[7] HAVEit Deliverable D33.5 “Algorithm (C-code, 2st version) available for partners”, June 2010

[8] Boehm, B.W. (1986), A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes, v.11 n.4, pp.14-24

[9] Höger, R.; Amditis A. , Kunert M.; Hoess, A.; Flemisch, F.; Krueger, H.-P.; Bartels, A.; Beutner, A., Pagle, K. (2008). Highly automated vehicles for intelligent transport: The HAVEit approach; ITS World Congress, NY, USA.

[10] Flemisch, F.; Nashashibi, F.;, Rauch, N.;, Schieben, A.; Glaser, S.; Temme, G.; Resende, P.; Vanholme, B.; Löper, C.; Thomaidis, G.; Mosebach, H.; Schomerus, J.; Hima, S.; Kaussner, A.: Towards Highly Automated Driving: Intermediate report on the HAVEit-Joint System, “Transport Research Arena Europe”, Brussels, 2010

[11] Flemisch, F.; Schieben, A.; Kelsch, J.; Löper, C. (2008). Automation spectrum, inner / outer compatibility and other potentially useful human factors concepts for assistance and automation; In: de Waard, D.; Flemisch, F.; Lorenz, B.; Oberheid, H.; Brookhuis, K. Human Factors for Assistance and Automation; Shaker, Maastricht.

[12] Glaser, S.; Vanholme, B; Mammar, S; Gruyer, D; Nouveliere L.: “Maneuver based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction”, IEEE Transaction on Intelligent Transportation Systems, Jun 2010

[13] Löper, C.; Flemisch, F. O. (2009): „Ein Baustein für hochautomatisiertes Fahren: Kooperative, manöverbasierte Automation in den Projekten H-Mode und HAVEit“, Stiller, Christoph; Maurer, Markus (Hg.): 6. Workshop Fahrerassistenzsysteme. Karlsruhe: Freundeskreis Mess- und Regelungstechnik Karlsruhe e.V., S. 136–146

[14] Project page of the open source version control system “subversion”: http://subversion.tigris.org/

[15] Rauch, N., Kaussner A., Krüger, H.-P., Boverie, S. & Flemisch, F. (2010). Measures and Countermeasures for impaired driver’s state within highly automated driving. Paper presented at the Transport Research Arena Europe 2010 (TRA2010), Brussels, Belgium, 07.-10. June 2010.

[16] Rauch, N., Kaussner, A., Krueger, H.-P., Boverie, S. & Flemisch, F. (2009). The importance of driver state assessment within highly automated vehicles. Paper presented at the 16th ITS World Congress, Stockholm, Sweden, 21.-25. September, 2009

[17] Schieben, A., Heesen, M., Schindler, J., Kelsch, J., & Flemisch, F. (2009). The theater-system technique: Agile designing and testing of system behavior and interaction, applied

to highly automated vehicles. Paper presented at the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications, Essen, Germany.

[18] Vanholme, B.; Glaser, S; Mammar, S, Member, IEEE, Gruyer, D.: “Maneuver based trajectory planning for highly autonomous vehicles on real road with traffic”, In Proceedings of ECC’09, Budapest, Hungary, 23-26 August 2009.

[19] Vanholme, B; Gruyer, D; Glaser, S; Mammar, S:”Fast prototyping of a Highly Autonomous Cooperative Driving System for public roads”, In Proceedings of IV’10, San Diego, USA, 21-24 June 2010

[20] Flemisch, F.; Schindler, J.; Kelsch, J.; Schieben, A.; Damböck, D.: Some Bridging Methods towards a Balanced Design of Human-Machine Systems, Applied to Highly Automated Vehicles; Applied Ergonomics International Conference, Las Vegas, USA; 2008

Annex 1: Keywords

ACC	Active/Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
CAN	Controller Area Network
CMOS	Complementary Metal-Oxide-Semiconductor
Co-System	Software components within the Joint System
CSC	Chassis Safety Controller
DA	Automation level Driver Assisted
DDM	Driver's Drowsiness Monitoring
DIM	Driver’s Inattentiveness Monitoring
DMS	Driver Monitoring System
DSA	Driver State Assessment
ECU	Electronic Control Unit
ELF	Extensible Linking Format
FASCar	Advanced Driver Assistance System (German: “FahrerAssistenzSystem”) Car
GPS / D GPS	Differential Global Positioning System
HA	Automation level Highly Automated
HAVEit	Highly Automated Vehicles for Intelligent Transport
HMI	Human Machine Interface
I2V / V2V	Infrastructure to Vehicle / Vehicle to Vehicle communication
IMU	Inertial Measurement Unit
JS	Joint System
MRM	Automation level Minimum Risk Manoeuvre
MSU	Mode Selection and Arbitration Unit
NIR	Near-Infrared
OPENWRT	A Linux-based firmware program for embedded devices
RAM	Random Access Memory
RTK	Real Time Kinematics
SA	Automation level Semi Automated
SCOPE	Supervisory Control Of Program Execution
SILAB	Driving simulation software from WIVW
SMPL	Straight forward Modular Prototyping Library
UML	Unified Modelling Language
UTM	Universal Transverse Mercator

VWSCF	Web Service Communication Framework
XCC	X-by-wire Control Computer