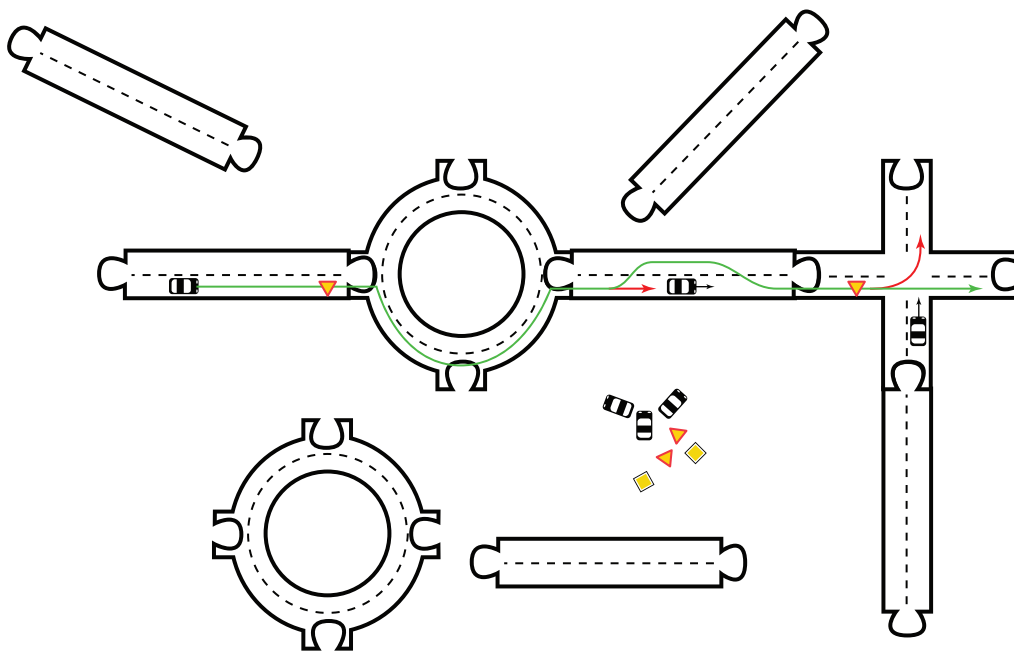


CHALMERS



Automated Drive Environments and Decision Making *Master's thesis in Systems, Control and Mechatronics*

NICKLAS GUSTAFSSON

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2013
Master's thesis EX009/2013

MASTER'S THESIS IN SYSTEMS, CONTROL AND MECHATRONICS

Automated Drive

Environments and Decision Making

NICKLAS GUSTAFSSON

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2013

Automated Drive
Environments and Decision Making
NICKLAS GUSTAFSSON

© NICKLAS GUSTAFSSON, 2013

Master's thesis EX009/2013
Examiner: Jonas Sjöberg (jonas.sjoberg@chalmers.se)
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Modelling traffic environments can be seen as a jigsaw puzzle, where pieces that are put together constitute to a traffic scenario. Automated drive control systems is provided with information, from the environment, to make correct decisions and execute the corresponding behaviour.

Chalmers Reproservice
Gothenburg, Sweden 2013

Automated Drive
Environments and Decision Making
Master's thesis in Systems, Control and Mechatronics
NICKLAS GUSTAFSSON
Department of Signals and Systems
Chalmers University of Technology

ABSTRACT

Recent advances in the automotive industry, within the area of active safety, has paved the way for sophisticated automated drive systems. That is, systems that help drivers to operate their vehicles. A great number of stakeholders have identified automated drive systems as important parts of future automotive systems, with respect to passenger safety, comfort and environmental impact. It is hence of interest to develop such systems efficiently, using modelling and simulation software. This work focuses on two areas, evaluating traffic environment modelling tools and developing a system for automated drive. Two tools, PreScan[®] and CarMaker[®], are tested by building a traffic model, containing roads, infrastructure and traffic flows. This thesis describes the tools' User Interfaces (UI) for specifying the model, -for connecting Simulink[®] developed control systems and -for visualizing simulation results. The study results in a table, stating the strengths of the two tools respectively; PreScan has an excellent drag and drop UI for specifying roads, infrastructure and traffic flows; CarMaker has an advanced default vehicle dynamics model, and a great UI for visualizing simulation results. An automated drive control system is developed, implemented and connected to the traffic environments. A three-layered planning architecture allows the system to sense, plan and execute a legal and efficient behaviour. An Adaptive Cruise Control (ACC) component is used to execute longitudinal decisions and prevent collisions. The resulting system drives a host vehicle successfully, through the traffic environments developed using the modelling tools.

Keywords: Modelling traffic environments, Decision making, Efficient overtake and merging, Intersection driving, Roundabout driving

ACKNOWLEDGEMENTS

I would like to express my appreciation to Volvo Car Corporation, for providing me the opportunity to perform my Master's Thesis in a stimulating environment. A special thanks goes to Dr. Stefan Soylo, who has been my supervisor throughout the last twenty weeks. Your advice and guidance has been of great importance for my work. I would also like to acknowledge TNO and IPG Automotive for providing me the opportunity to use their software, PreScan and CarMaker respectively, to achieve my results.

NOMENCLATURE

Host car

The host car, or ego vehicle, is the car for which the automated drive system is designed and implemented.

Target

A target is an object (infrastructure or vehicle) that typically should be considered by the automated drive algorithm, e.g. a speed sign indicating a new set speed, or a vehicle navigating through a roundabout ahead of the host car.

Target angle

The target angle $\alpha_{tar} \in [-\pi, \pi]$ is defined positive counter clockwise from the host car's heading to the target's position.

Target range

If $\mathbf{v}(t)$ is a vector from the host car to a specific target at time instance t , the range is given as

$$d_r(t) = \sqrt{\mathbf{v}(t)^\top \mathbf{v}(t)} \quad (0.0.1)$$

Target range rate

The target range rate at time instance t is given by

$$d_{rr}(t) = \frac{d}{dt} d_r(t) \quad (0.0.2)$$

Hence a negative range rate translates to a target approaching the host car.

Time-to-collision

Time-To-Collision (TTC) is given by

$$t_{ttc} = -\frac{d_r}{d_{rr}}. \quad (0.0.3)$$

It is the time to collision if host and target continue with constant velocities. Systems developed in this work use TTC to distinguish which targets are of most importance to the host car. TTC should not be confused with *inter vehicle time*, which assumes that the target vehicle is stationary. It is straight forward to include acceleration in Eq. 0.0.3 (Gietelink, 2007). However, it can be hard to estimate target vehicles acceleration and is hence not included in this work.

Lateral direction, longitudinal direction and yaw angle

Figure. 0.0.1 illustrates direction- and angle conventions related to the host vehicle.

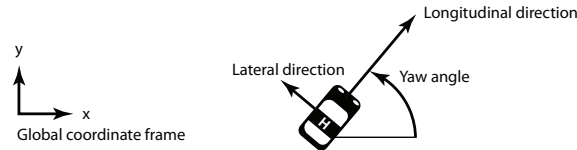


Figure 0.0.1: *Conventions of directions and angles related to the host vehicle.*

CONTENTS

Abstract	i
Acknowledgements	i
Nomenclature	iii
Contents	v
1 Introduction	1
2 Thesis outline and contributions	2
3 Modelling tools for traffic environments	3
3.1 Scenaria descriptions	3
3.2 PreScan evaluation	3
3.2.1 Building a PreScan traffic environment model	3
3.2.2 Modelling sensors to access control signals	5
3.2.3 Adding Simulink control systems	6
3.2.4 Simulating a scenario using PreScan	7
3.3 CarMarker evaluation	8
3.3.1 Building a CarMaker traffic environment model	8
3.3.2 Accessing signals in Simulink	10
3.3.3 Adding Simulink control systems	11
3.3.4 Simulating a scenario using CarMaker	12
3.4 Summary of software strengths	13
4 Decision making for automated vehicles	14
4.1 Software architecture for automated drive	14
4.2 Perception	15
4.2.1 Road model	15
4.2.2 Targets	16
4.3 Mission planner	16
4.3.1 Road sign information	17
4.3.2 List of precedence	17
4.4 Behaviour planner	18
4.4.1 Precedence based decision making for legal behaviour	18
4.4.2 Cost evaluation for planning of efficient overtake manoeuvres	18
4.5 Motion planner	22
4.5.1 Lane defined lateral motion	22
4.5.2 Trajectory-defined lateral motion	24
4.5.3 Longitudinal motion	24
4.6 Execution of motion goal	24
4.6.1 Lateral control	24
4.6.2 Longitudinal control	25
4.7 Results of the automated drive system	26
5 Conclusions	30

1 Introduction

Volvo Car Corporation's (VCC) vision is that "by 2020, nobody shall be seriously injured or killed in a new Volvo" (Volvo Car Corporation, 2013b). Hence, research and development efforts are needed, to increase passenger safety, in future automotive systems. It can be done by improving three factors: infrastructure, vehicle and driver (HAVEit, 2012b). For instance, improvements to passive, vehicle related, safety functions such as air-bags and deformation zones are needed. However, since the great majority of accidents are caused by the driver (HAVEit, 2012b), safety functions that actively prevents accidents are of great importance to reduce traffic casualties and injuries. A particular area of interest, related to active safety, concerns the development automated drive systems. That is systems that use sensors to perceive the surrounding traffic environment, and help drivers to operate their vehicles; for instance, by taking direct control of the vehicle's throttle and steering. Recent advances within the area has paved the way for systems such as Adaptive Cruise Control (ACC). It is a cruise control with the addition that it has sensors to get information of the traffic ahead. It can hence conclude that a vehicle in front drives too slow- or close to the host vehicle, and act by adapting the host vehicle's velocity (Rajamani, 2012). Original Equipment Manufacturers (OEM) such as Volvo and BMW use the technology in their vehicles (Volvo Car Corporation, 2013a; BMW, 2013). The importance of automated drive systems is acknowledged by a great number of industry and science representatives (HAVEit, 2012a; interactIVe, 2012), not only due to safety reasons, but also as key components in the goal of reducing environmental impact and increasing passenger comfort.

Developing automated drive systems is a time- and resource intense process. They have to be tested rigorously in a variety of scenaria, to ensure that the systems behave as desired. Often, car OEMs have a fleet of test vehicles, to validate systems' functionality. Hence a significant development cost is spent on equipment, driver hours, analysis of results and even on finding a road that is suitable for a desired scenario test. This strongly motivates a need for computer aided development tools that can reduce the cost carriers. MATLAB's[®] toolbox Simulink (MathWorks, 2013) is a modelling and simulation tool, widely established in industry, for design of control systems. However, it is not convenient to specify traffic environments in Simulink. For that cause, specialized Simulink compatible traffic environment modelling tools are needed to realize specific scenaria, i.e. make the automated drive system behave as intended in specific traffic situations.

As mentioned, automated drive systems can assist the driver by supporting at specific situations, and hence keep the driver in the loop. Going further towards totally excluding the driver from the loop, to avoid human mistakes, a fully automated drive system is needed to navigate the vehicle. Such an implementation puts higher demands on the system's framework. It has to be modular and scalable to handle the vast amount of traffic situations that a driver usually deals with. For instance, Urmson et al. (2008) uses a three layer planning framework that separates tactical planning from making decisions and executing the corresponding behaviour. The system manages to drive a host vehicle through an urban traffic environment successfully, without human interventions. Such fully automated vehicles might possibly be a part of the solution to fulfil VCC's vision of zero deaths and injuries in future vehicle systems.

2 Thesis outline and contributions

The contributions of this work concern traffic environment modelling tools (described in Section 3) and the implementation of an automated drive system (described in Section 4). Two commercial modelling tools, PreScan (Tass, 2013) and CarMaker (IPG Automotive, 2013), are examined. A set of scenaria (described in Section 3.1) is realized and the tools' following features are considered:

- User interface for specifying traffic models, i.e. how different kinds of roads, road signs, vehicles and their movements are specified.
- Signal interface between traffic environment model and an automated drive control system developed in MATLAB/Simulink.
- Visualization of simulation results.

The resulting first contribution, of this work, is a table of the two tools' strengths (presented in Section 3.4) with respect to the features considered.

An automated drive framework (described in Section 4) is developed and implemented. Its modular design (illustrated in Section 4.1) is a contribution itself, separating tactical-, behavioural- and motion planning layers (described separately in Section 4.3–4.5). The third contribution, of this work, is a road model parametrization (described in Section 4.2) that allows the planning layers to prepare a precedence list and make context-based decisions. Further, an Adaptive Cruise Control (ACC) component is implemented (described in Section 4.6) to execute the decisions issued. However, it is also used to prevent collisions with vehicles that are in the host car's navigation path. The last contribution of this work is an algorithm (described in Section 4.4.2) that evaluates a cost function to make efficient overtake decisions with respect to acceleration and velocity deviations from desired reference values. The complete system successfully drives a host vehicle through the traffic environment models built for the scenaria in Section 3.1, the results are described and depicted in Section 4.7.

3 Modelling tools for traffic environments

This section describes the evaluation of two traffic environment modelling tools, PreScan and CarMaker, selected after a limited market survey. The tools' abilities to model traffic environments, connect with control systems and visualize simulation results are specifically considered. A set of scenarios is realized to evaluate the tools' strengths, with respect to the features considered. The results are presented in the end of this section.

3.1 Scenarios descriptions

A set of scenarios is selected to put demands on specifying complex road shapes, timing traffic flows and routing signals to- and from the automated drive control system. The following scenarios are examined:

- Straight- and curved highways:
The host car should follow roads that are curved or straight with a set speed.
- Traffic signs:
The host car should be able to handle speed signs, stop signs and rights of way.
- Overtaking:
Two situations are of interest:
 - The host car approaches a slow driving car in the right lane of a highway. The automated drive system should decide if an overtake is safe and efficient. If so, the overtake is performed.
 - The host car approaches a slow driving car in the right lane; another car is approaching fast enough from behind for an overtake manoeuvre to be considered unsafe. The host car should then decide to adjust its speed to the slow driving vehicle in front.
- Highway exit:
The host car should leave the highway by taking an exit.
- Roundabout:
The host car should be able to drive through a roundabout.
- Intersection:
The host car should be able to drive through an intersection.

3.2 PreScan evaluation

This section describes a four step modelling and simulation process using PreScan with MATLAB/Simulink. It describes how roads and traffic flows are defined, how signals are routed to the control system and how the resulting model is simulated.

3.2.1 Building a PreScan traffic environment model

PreScan uses a Graphical User Interface (GUI) with drag and drop functionality to import infrastructure- and actor objects (e.g. cars) from an object *library* to a *build area*. Most objects that are placed in the build area are customizable via a *property editor* on the right hand side of the GUI. In addition, objects can be configured by right clicking on the specific object and selecting *object configuration*. The PreScan GUI is depicted in Fig. 3.2.1.

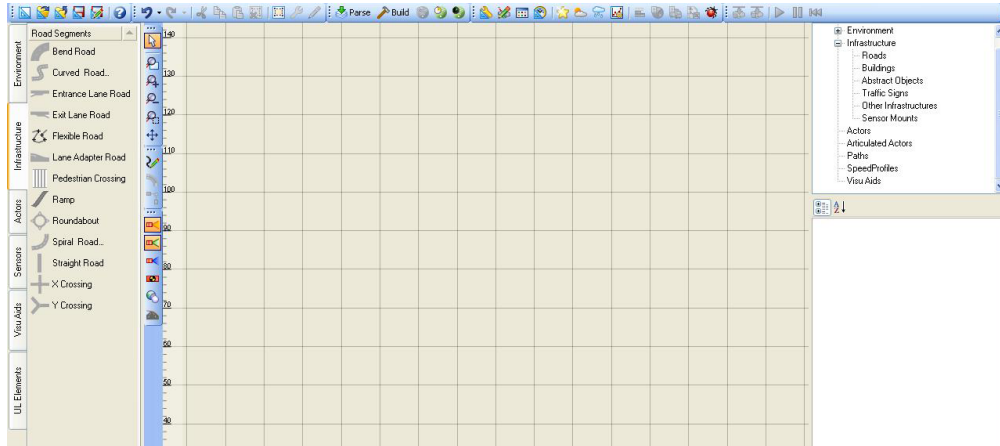


Figure 3.2.1: The PreScan GUI consists of an object library, toolbars, a build area and a property editor.

Infrastructure

PreScan comes with a large set of predefined and customizable infrastructure objects including road segments, buildings, traffic signs etc. In this work, road segments and traffic signs are of most importance. The type of road segments used are:

- Straight road
- Roundabout
- Intersection
- Curved road
- Highway exit
- Lane adapter (merges two lanes into one)

The appearance of predefined infrastructure is customized, to the extent needed, by dragging the road segment's ends with the mouse cursor, or by changing parameters in the property editor. Road segments are put together by connecting road joints, which appear as green dots when a segment is selected (see Fig. 3.2.2). Additional road joint connection points can be added to the ends of road segments. In that way diverging lanes, depicted in Fig. 3.2.3, can be achieved.

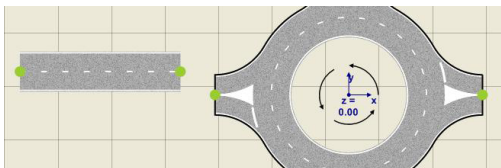


Figure 3.2.2: When a road segment is selected and moved, green connection dots appear in the ends of the segments. If two connection points approaches each other, they snap and connect.

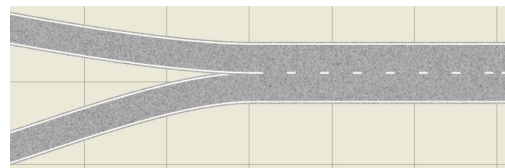


Figure 3.2.3: Road joints can be edited to add additional connection points, which enables constructions such as diverging roads.

The best practice, using PreScan, is to start by adding pieces of infrastructure whose desired positions and orientations are known (usually *straight roads* and different types of *crossings*). These road segments can then be fixated to the build area and connected using the *Curved Road* type, which is a flexible road segment that automatically creates a smooth intermediate road

between segments. Traffic signs are added by dragging and dropping a *sign object* to the build area. PreScan prompts the user to specify the appearance of the sign’s plate and pole. A library of hundreds of signs is available and it is also possible to add custom images to the sign plate. The final road model is shown in Fig. 3.2.4. It contains all traffic contexts required by the scenario in Section 3.1.

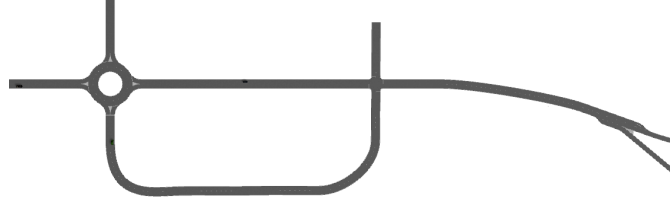


Figure 3.2.4: A top-view of the resulting PreScan model. Starting from the leftmost point, the model contains a roundabout, a highway strip, an intersection and a highway exit.

Actors and trajectories

Cars and pedestrian objects are called *actors*, in PreScan. They are typically dynamic objects that interact with the infrastructure. In this work actors are used as both host- and target cars. An actor *trajectory*, i.e. movement definition, is given by a *path* and a corresponding *speed profile*. The actor’s path is most conveniently defined using the inherited path tool found in the left toolbar. Using the tool, it is sufficient to specify the end joints of a specific road segment, PreScan automatically calculates a path between the points. When an actor is dropped on a path, PreScan generates a default, but configurable, speed profile and assigns the resulting trajectory to the actor. In this work, the host car is not assigned a trajectory since it is controlled by an automated drive system.

3.2.2 Modelling sensors to access control signals

Sensors are used to obtain information about the traffic environment’s state, in order to facilitate for the host car’s control system. They are added to actors- and infrastructure by drag and drop. Objects of sensory interest have to be configured as detectable, in order for the sensors to register them. It is done by selecting a ‘sensor detectable’-checkbox in the specific object’s configuration. There are three main types of sensors available in PreScan:

- Idealized
- Detailed
- Ground truth

In this work, *ideal antennas* are used for one-way Infrastructure to Vehicle- (I2V) and Vehicle to Vehicle (V2V) communication, to send information to the automated drive control system. Antenna transmitters allow infinite range of detection and are put on infrastructure and actors that should be detectable by the host car. The transmitters send general information about the object such as a unique identifier (ID) and global coordinates. In addition, the sensor allows custom signals to be used. This was used to send information about target type (e.g. speed sign) and an arbitrary associated parameter (e.g. speed limit). Transmitters send information over a specified channel; to lump all data on a single bus, all transmitters are connected to the same channel. The host car uses an antenna receiver to obtain the information broadcasted on the channel. The antenna set-up is illustrated in Fig. 3.2.5.

Unfortunately, targets can change place on the Simulink bus during simulation. It might confuse control systems that assume that signal number n on the bus always corresponds to target n . This phenomenon, illustrated in Fig. 3.2.6, is due to PreScan that orders targets by range to the host vehicle. The solution approach used in this work is to use an algorithm, that sorts the bus signals by their unique identifiers. As a consequence, signal number n always corresponds to target n throughout the entire simulation.

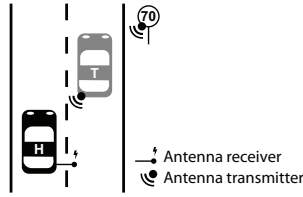


Figure 3.2.5: All targets (infrastructure and actors) are equipped with antenna transmitters that broadcast information on a data channel. The host car (H) uses an antenna receiver to obtain the target information.

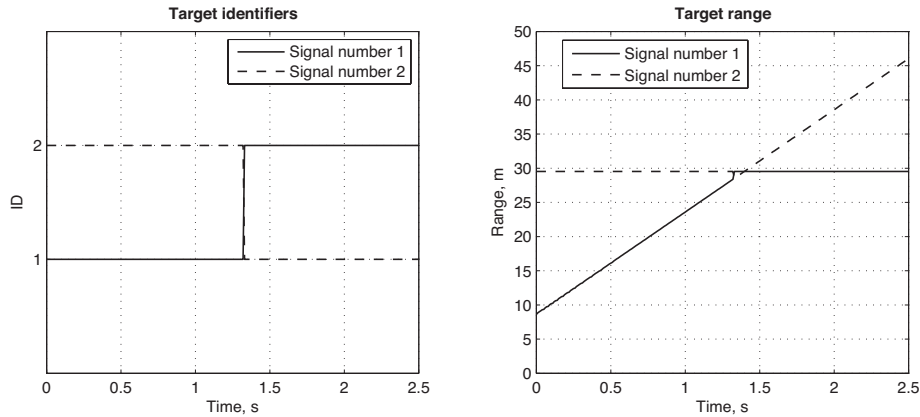


Figure 3.2.6: **To the left:** Two targets, with identifiers 1 and 2, changes place on the Simulink bus, coming from the sensor interface. **To the right:** PreScan orders the targets on the bus by range. Hence signal 1 always corresponds to the target closest to the host vehicle. For that cause, targets can change place on the bus during simulation.

A lane marker sensor is used to obtain the host car’s lateral state on the road. It is a so called ground truth sensor, i.e. a sensor with an associated algorithm. It outputs information regarding distance to nearby road markings; and it operates by using a customizable number of scan lines, depicted and described in Fig. 3.2.7. Even though this work is limited to ideal sensors, it should be mentioned that PreScan supports customizable detailed sensors such as Camera, LIDAR, radar and Ultrasonic.

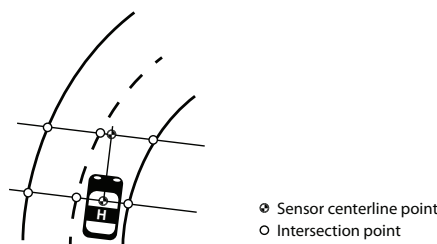


Figure 3.2.7: The host car is equipped with a lane marker sensor that outputs coordinates of nearby lane markers at intersection points. It utilizes a customizable number of scan lines, perpendicular to the sensor heading, to search for intersecting lane markers. In addition, the sensor outputs information regarding its own position and heading (same as the host car).

3.2.3 Adding Simulink control systems

MATLAB/Simulink is invoked from PreScan to initialize variables and launch a 3D-viewer called *VisViewer*. A *compilation sheet* is generated by PreScan when the model is compiled. It is a Simulink .mdl-file that contains an interface between the traffic environment model and user specified Simulink blocks. The compilation sheet is divided into blocks of the individual actor-

and infrastructure objects that have associated sensors. The compilation sheet for this work is illustrated in Fig. 3.2.8. The host car (upper left block) has a lane marker sensor and an

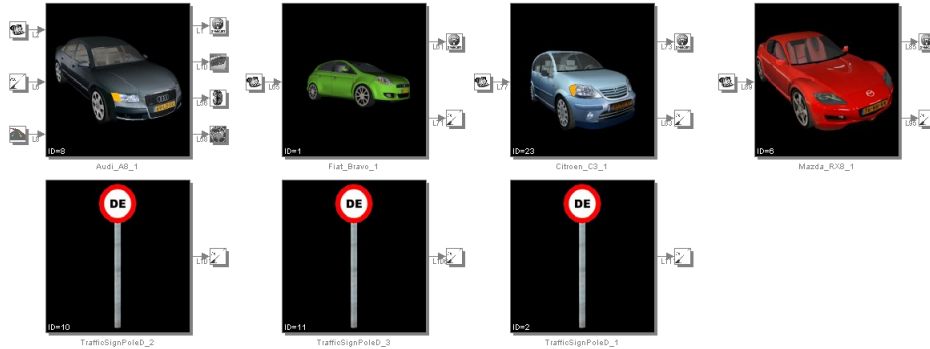


Figure 3.2.8: A PreScan compilation sheet consist of actors- and infrastructure that have sensors attached. Sensor and visualizer interfaces are automatically generated when a PreScan model is compiled.

antenna receiver as input; and interfaces to VisViewer as output. The traffic signs on the other hand, solely has an antenna transmitter output. The general interface, between PreScan and user defined Simulink blocks, is illustrated in Fig. 3.2.9. The host car block, with its content depicted

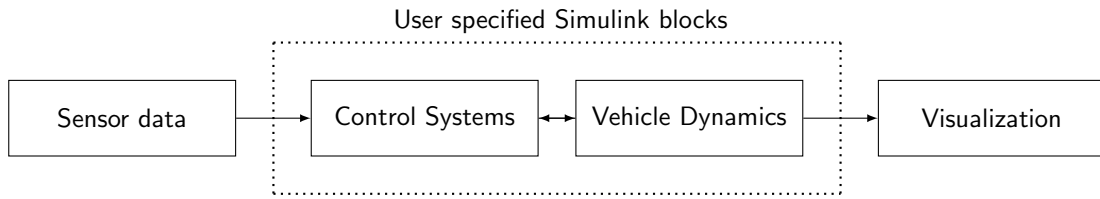


Figure 3.2.9: Sensor information is available to user specified Simulink blocks for control and vehicle dynamics. State data from the vehicle dynamics is muxed and sent to a visualizer.

in Fig. 3.2.10, shows the PreScan-Simulink vehicle control interface. The interfaces to sensors and visualizer are marked with a gray overlay color. Actor states and sensor data is generated by PreScan to the left; to the right is a state interface for user specified vehicle dynamics. The state interface to the left is just a copy of the state interface to the right, for convenient signal routing to the control system.

A VCC vehicle dynamics model was incorporated in the control system design by copying it into the host car's block. The visualizer needs vehicle states, such as pitch, yaw, roll, position and velocity, in order to display a correct representation of the simulated dynamics. A *state mux block* to do so, is provided as interface by PreScan. Not all states have to be given, but naturally the visualizer will not display a turning vehicle correctly if no yaw information is given.

3.2.4 Simulating a scenario using PreScan

The simulation of a scenario is invoked from Simulink and VisViewer visualizes it during runtime. A set of mouse commands can be used to manipulate the viewing angle and zoom level. It is also possible to add custom camera views and record the results as a movie file. However, the recording has to be done during simulation and can not be done offline.

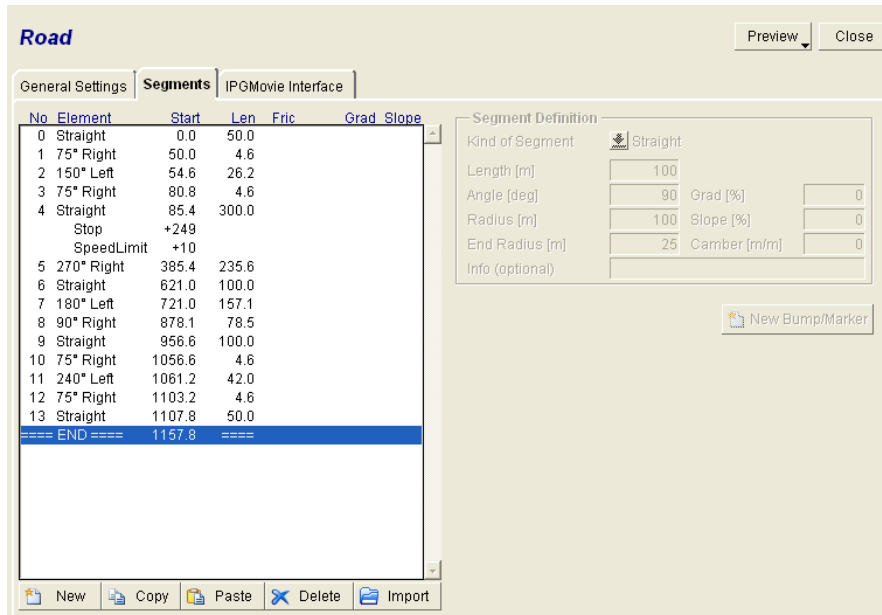


Figure 3.3.1: CarMaker GUI for specifying a TestRun road model. The segments (No 0–13), specified in the list to the left, makes up for the pieces of road and road signs that are needed to fulfil the desired scenaria. Associated parameters, such as length, radius and angle, are used to customize the segments' appearance.

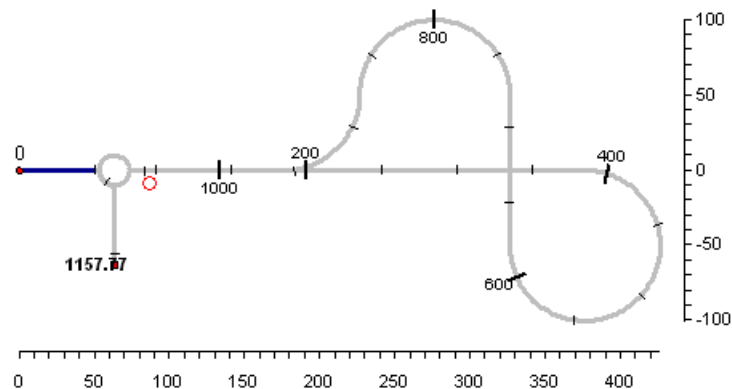


Figure 3.3.2: A bird's eye view, of the road model, constructed to realize the desired scenaria. Starting from the leftmost point, the road extends straight trough a roundabout and an intersection. It turns back, passes the intersection once more and turns left in the roundabout. It is also possible to understand the road's extent by following the distance markers (0-200-400-600-800-1000-1157.77) from start to finish. A speed sign is placed in the beginning of the highway strip after the roundabout; a stop sign is placed on the west entry of the intersection.

not possible to zoom. It is therefore not easy to accurately e.g. inspect roundabout's midpoint coordinates. An intersection was realized by letting the road cross itself, thus making a 270 degree turn. Similarly, the roundabout was realized by letting the road turn back and create a third exit (south). The roundabout's curvature was achieved by a series of right- and left turns with appropriate radii. A highway exit was not implemented due to the limitations described above.

CarMaker comes with a few pre-defined road signs, but it is also possible for the user to add their own. They are added to the road segments individually by specifying a longitudinal offset from the start of a segment. In this work, a speed sign was added to the highway strip after the roundabout; a stop sign was added to the intersection entry connecting from west.

Adding target vehicles to the traffic model

Traffic objects (actors) are added to the environment by specifying their type and motion in a list interface. A few pre-defined types are available, e.g. compact car, truck, bus, motorcycle and pedestrian. The motion of an actor is tightly connected to the road model's definition. It is given by a lateral- and longitudinal offset, from the road's centreline and -start respectively, together with a time of arrival. Hence, traffic vehicles follow the road model's segment list sequentially, either as oncoming vehicles or the opposite. This caused limitations to the actors' possible movements, e.g. target vehicles have to pass straight through the intersection, since the road does so. Figure 3.3.3 exemplifies the GUI for specifying traffic. Three target vehicles (T01, T02 and T03) and their movements have been added to realize the desired scenaria.

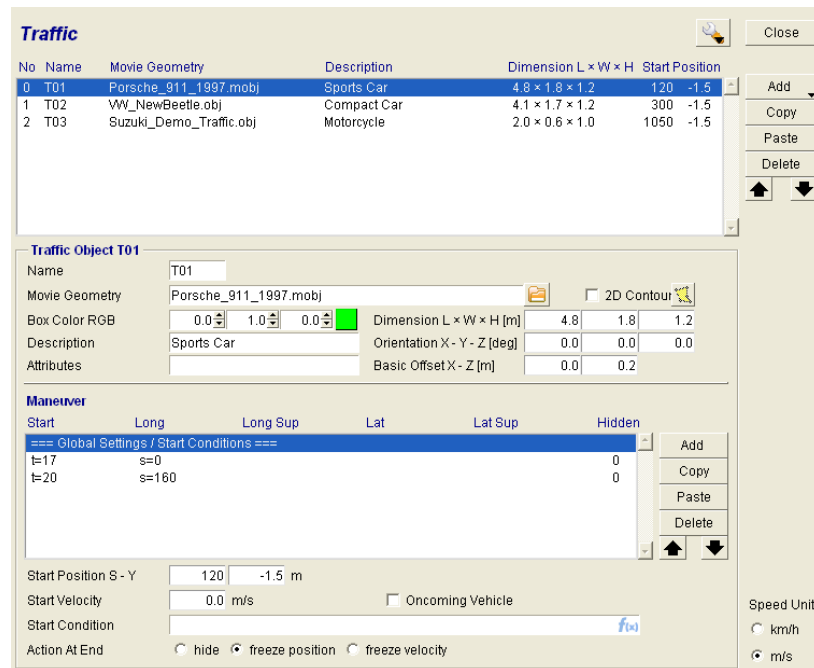


Figure 3.3.3: CarMaker uses a list interface to specify traffic, i.e. to define target vehicles and their movements over time.

Host vehicle dynamics

It has to be mentioned, even though it is not the main focus of this work, that CarMaker comes with an advanced default vehicle dynamics model for the host car. The model is specified within the CarMaker car GUI; it is highly customizable through great many parameters concerning systems such as engine, suspension, tires, brakes and powertrain. This has a great positive effect on the realism of the simulation visualization.

3.3.2 Accessing signals in Simulink

It is possible to equip the host vehicle with different types of sensors, such as radar, LIDAR or sonar. CarMaker does not automatically generate a Simulink interface to access the sensor signals. It has to be built manually using a set of blocks provided by CarMaker. However, since this work assumes ideal sensors, a simpler way of accessing relevant signals for automated drive is available. In fact, all relevant signals for this work is available directly in Simulink as exemplified in Fig. 3.3.4, where the global coordinates of three targets (T01, T02 and T03) are read. It is sufficient to specify a signal's name to import it to the Simulink workspace. It is also possible to write to the signals, e.g. a lateral offset parameter was overwritten during overtake manoeuvres, thus making the host car change lane.

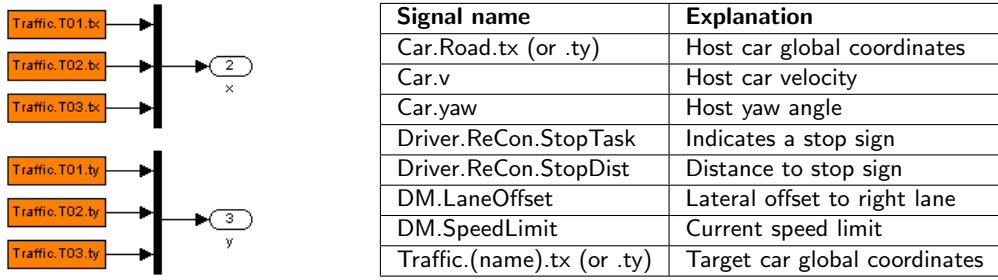


Figure 3.3.4: **To the left:** Three targets' (*T01*, *T02*, and *T03*) global coordinates are accessed by specifying the signals' names. All signals available during simulation can be accessed in the same way. It was therefore not necessary to equip the host car with sensors such as radar or LIDAR. **To the right:** The set of signals that was used in this work.

3.3.3 Adding Simulink control systems

When a new TestRun project is created, CarMaker prompts the user to select whether Simulink is to be used in the project or not. Selecting yes generates a folder called *src_cm4sl*, containing a default Simulink .mdl-file which is connected to the TestRun. The .mdl-file contains an interface to the host car's vehicle dynamics as illustrated by Fig. 3.3.5. CarMaker has a built in driver

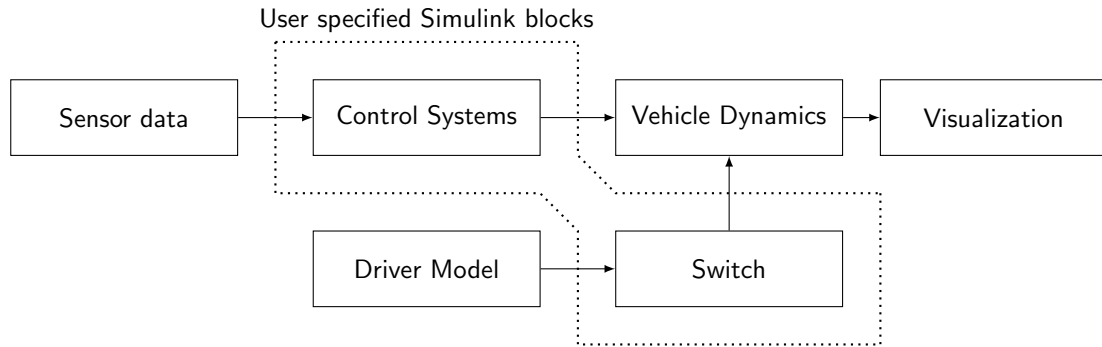


Figure 3.3.5: A driver model controls the host vehicle by default. Its signals has to be disabled (illustrated by a switch) in order to enable a user specified control system to govern the vehicle dynamics.

model, called *IPGDriver*, to govern the vehicle dynamics. It follows a list of manoeuvres that specifies the host car's desired lateral- and longitudinal position on the road, similar to the actors added in the traffic model. Hence, a simulation of the vehicle dynamics can be invoked without adding any custom control systems to the Simulink model. In this work, the driver was told to follow the right lane using a set speed. Consequently, the host car drives through the roundabout and adjusts its speed to the traffic regulations given by the road signs. However, it was desired to add an user specified control system, hence the *IPGDriver* model was disconnected from the gas and brake signals in the Simulink vehicle control interface, depicted in Fig. 3.3.6. Thus, still allowing it to steer the car to follow the road model. An offset parameter was used to make the driver change lane during overtake manoeuvres. This way of peeling of- and adding custom control systems to the model can be quite advantageous when compared to building a complete system from scratch, especially if it is desired to examine some minor sub-system for automated drive.

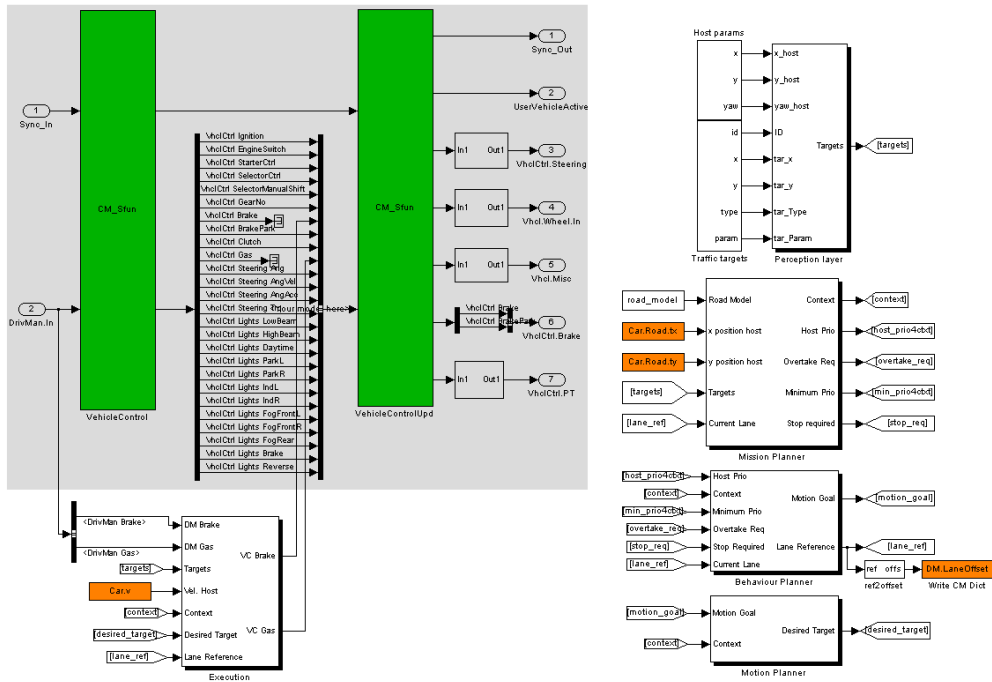


Figure 3.3.6: Vehicle control systems are added by substituting IPGDriver signals. The interface to the host car's vehicle dynamics is highlighted in grey colour.

3.3.4 Simulating a scenario using CarMaker

TestRun simulations are invoked from CarMaker, which then calls for the control systems specified in Simulink. A GUI component called *IPGMovie* (depicted in Fig. 3.3.7) visualizes the simulation



Figure 3.3.7: A GUI component, *IPGMovie*, visualizes the scenario simulation. It is possible to rewind and replay the simulation, which can be useful for debugging the control system's behaviour. *IPGMovie* also allows the user to record videos of the simulation post runtime.

in 3-D, from a camera angle that can be adjusted by using mouse commands. The visualization results are saved to a buffer, thus enabling the user to rewind and record movies post runtime. This function is particularly useful when the control system does not work properly during simulation. It is visually possible to see when the host car does not behave as expected. Then, by

rewinding and noting the simulation time, signals that affects behavioural decisions are examined and the control is system corrected.

The advanced host car vehicle dynamics affects the simulation greatly. The simulation runs relatively fast, often faster than real time. In addition, the visualization of the host vehicle driving through the traffic model appears very detailed. IPGMovie visualizes tire forces, pitch and roll.

3.4 Summary of software strengths

This section summarizes the strengths of PreScan and CarMaker, with respect to modelling of traffic environments, Simulink interface and simulation visualization. The result is presented in Tab. 3.4.1

Table 3.4.1: Summary of software strengths.

PreScan Strengths	
Modelling	Drag & drop for building road networks and adding actors
	Highly customizable road segment types (e.g. roundabout)
	Flexible specification of actor trajectories, not necessarily tied to road definition
	Great capabilities of specifying sensors
Simulink interface	Simulink interface to sensors and actors is auto-generated
	Easy to incorporate Simulink control systems
	Straight forward to use user specified Simulink designed vehicle dynamics
Simulation	3-D visualization of traffic environment during simulation

CarMaker Strengths	
Modelling	Advanced default (customizable) host vehicle dynamics
Simulink interface	Built in driver model enables fast sub-system tests
	Easy to access simulation signals
Simulation	3-D visualization of traffic environment during simulation
	Fast simulation
	Buffers the visualization results, hence enabling rewind post runtime

4 Decision making for automated vehicles

This section presents the implementation of a three-layered planning framework, used to achieve automated drive for the scenario described in Section 3.1. The traffic environment model that is used to obtain the results, related to this section, is developed using PreScan. As a result Section 4.5, that describes the implementation of a motion planner, is related to PreScan sensors. However, similar results can be obtained by using a model developed using CarMaker. It is assumed that the system has access to all signals it needs, this since the development starts from an early stage and mainly focuses on the actual framework.

4.1 Software architecture for automated drive

The software architecture used in this work, illustrated in Fig. 4.1.1, is inspired by contributions to DARPA’s Grand- and Urban challenges (Urmson et al., 2009; Kammel et al., 2008; Urmson et al., 2008; Thrun et al., 2007). It uses a perceived representation, of a static- and a dynamic environment, to plan and govern an automated vehicle. The architecture is comprised of three planning layers:

1. Mission planner
2. Behaviour planner
3. Motion planner

They operate on different levels of control abstraction. The mission layer provides tactical information to the behaviour, in order for it to make correct decisions. The decisions result in motion goals that are decoded to a desired path, by the motion planner. The path is executed if

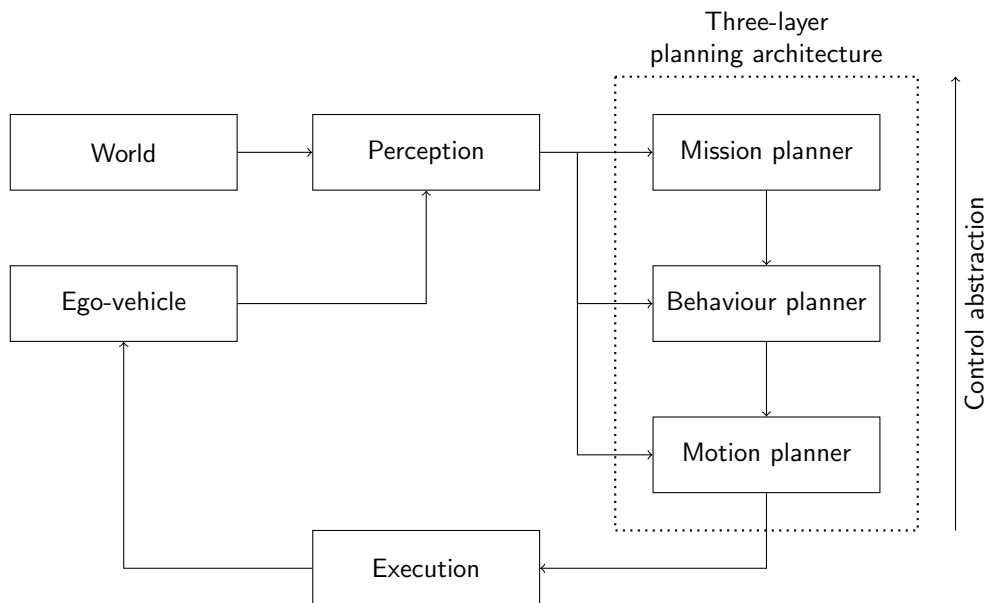


Figure 4.1.1: A perception layer fuses data to a format that fits a three-layer planning architecture; consisting of a mission-, a behaviour- and a motion planner. The mission layer provides information about the current traffic context (situation), in order for the behaviour planner to interpret the perception data, and make correct decisions. The motion planner generates the desired path needed to conform to decisions made. If the path is clear of target objects, the execution module generates the control input to the vehicle dynamics needed; if not, an ACC component intervenes to brake the host car to avoid collision.

no obstacles are in it, in which case the host car adjusts its velocity by using an ACC component. In that way, a potential collision is avoided.

4.2 Perception

This section describes the perception module, which generates a traffic environment model consisting of:

- a static road model
- targets vehicles and -infrastructure

It also provides host car information to the planning layers, such as local- and global position on the road, velocity, yaw etc.

4.2.1 Road model

A road model structure is defined, in order for the planning layers to be aware of the road geometry and prevailing traffic regulations. It contains parametrizations of multiple choice *traffic contexts*, i.e. road hubs such as roundabouts and intersections, that have multiple connecting roads. Highway is a third context type considered in this work. However, it is not included in the road model since it is defined to be the default context. The multiple choice traffic contexts in the road model have a set of general- and type specific parameters. The general parameters specifies:

- type, e.g. roundabout
- extent, defined as a midpoint and an outer radius
- inner zone, defined as a midpoint and an inner radius
- a speed limit v_{ctx}

Type specific parameters are used to describe the connecting roads and their traffic regulations. The road model used in this work is illustrated in Fig. 4.2.1. The roundabout does not have any type specific parameters. It is fully defined by its general parameters. On the other hand, the intersection has four connecting roads, each with a specific traffic regulation. Hence two parameters were used to describe each connecting road. An angle parameter tells where the road's centreline is located; a traffic regulation type tells whether the road is major or associated with a yield sign.

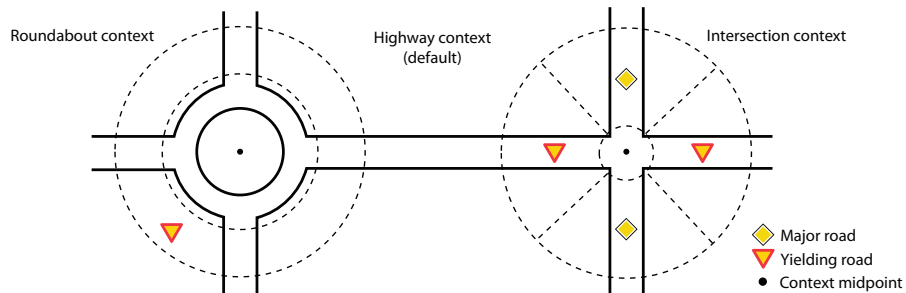


Figure 4.2.1: The road model holds geometry definitions for traffic contexts that are in the host car's navigation route. Both roundabouts (to the left) and intersections (to the right) have an extent (dashed outer circle) that defines when vehicles are inside the context. A inner zone (dashed inner circle) defines the limits where the actual intersection (or roundabout) starts. Context specific parameters define additional zones with associated traffic regulations, e.g. if any road is major.

4.2.2 Targets

All objects that are of interest to the planning algorithms, such as vehicles and traffic regulation signs, are considered as targets. Naturally, the targets have to be treated differently depending on their type, e.g. a stop sign does not necessary generate the same control signals as a yield sign. A general target structure, or bus protocol, is defined to have the following signals:

- Unique identifier
- Target type
- Type specific parameter
- Target range, -range rate and -angle
- Position in global coordinate frame

The types, with their corresponding parameters, appear in Tab. 4.2.1. An *idle target* is defined to represent a point in the far horizon.

Table 4.2.1: A type and a parameter is specified for each target. Thus enabling the decision making algorithm to distinguish between different targets and treat them accordingly. An idle type is specified to represent a target in the far horizon.

Type	Parameter
Idle (no target)	Not defined
Speed sign	Speed limit
Stop sign	Not defined
Car	Lane state (left or right)
Yield sign	Not specified

4.3 Mission planner

The mission planner has the objective to provide following set of current tactical information to the other planning layers:

- Host vehicle's current traffic context
- Entry- and exit points
- Overtake detection
- Road sign information
- List of precedence

The host vehicle's current context is found by examining the extents of contexts within the road model. If the host car is not found in any of the contexts, the default context (highway) is selected. If the current context is of hub type (multiple choice), entry- and exit points are calculated by the mission planner. The points specify where the host car should enter and exit the inner zone of the context. Thus, governing which navigation route to take. In this work, the route (and consequently the points) is assumed to be known in advance.

An overtake detection algorithm is activated during highway drive. It outputs a boolean that indicates if there is a car in the right lane, in front of the host vehicle, with a time-to-collision less than a specific time constant. The resulting overtake required signal makes the behaviour planner aware of the situation.

4.3.1 Road sign information

The mission planner keeps track of speed-, yield- and stop signs that are approached by the host car. A stop sign is essentially a yield sign with a requirement to unconditionally stop for a certain amount of time, t_{stop} . After the time has passed, the stop sign can be treated as a yield sign. A discrete event system was used to implement the function, thus only allowing the mission planner to inform that a stop sign is present for t_{stop} after reaching a standstill pose by the sign. The prevailing set speed v_{set} is given by

$$v_{set} = \min(v_{sign}, v_{ctxt}), \quad (4.3.1)$$

where v_{sign} is the current speed restriction given by speed signs; v_{ctxt} is the speed limit associated with the current context's road model.

4.3.2 List of precedence

The mission planner is responsible for preparing and updating a list of precedence, that specifies the priority for vehicles operating in the current context. The process of generating the list is similar for all contexts. All vehicles (including the host car) within the current context are paired with a traffic regulation, given by the road model; and an intended event, e.g. right turn or lane change. Together, they give a priority. Table 4.3.1 shows how priorities are assigned to vehicles (a lower number is higher priority) for different traffic contexts. Note that the assignment is independent of whether the car is a target or host. A vehicle is said to interfere with other traffic

Table 4.3.1: Intended events and prevailing traffic regulations are used to assign a priority number to all vehicles (including host) in the current context. A lower number represents a higher priority.

Context	Regulation and intended event	Priority
intersection	inside inner zone	1
	major road, right turn	1
	major road, heading straight	1
	major road, left turn	2
	yield, right turn	3
	yield, heading straight	3
	yield, left turn	4
roundabout	major road (inside inner zone)	1
	yield (in outer zone)	2
highway	follow lane	1
	change lane, no interference with other traffic	1
	change lane, interference with other traffic	2

if its intended behaviour disturbs other vehicles, either by directly ramming them or by blocking their way unsafely. For instance, if a car is approaching from behind with

$$0 \leq t_{ttc} < \tau, \quad (4.3.2)$$

where τ is a time constant. Then, a lane change is considered to interfere with other traffic.

4.4 Behaviour planner

The behaviour planner is responsible for making correct decisions, concerning how to respond to desired behaviour. The resulting decisions are issued as *motion goals*, later decoded by the lower level motion planner. The motion goals are either *longitudinal*, e.g. to yield at an intersection; or *lateral*, e.g. to change lane at an overtake manoeuvre. Motion goals have to be safe to execute, and for overtake manoeuvres, they have to be motivated by an efficient behaviour. The following sections presents approaches for achieving both a legal and an efficient behaviour.

4.4.1 Precedence based decision making for legal behaviour

Decisions are supported by the precedence list issued by the mission layer. The host car is defined to have precedence if

$$prio_{host} \leq prio_{min}, \quad (4.4.1)$$

where $prio_{host}$ is the host car's priority number. $prio_{min}$ is the smallest priority number of all target vehicles in the current context. Having precedence allows the behaviour planner to respond positively on desired behaviour. Figure 4.4.1 exemplifies the behaviour for two situations, in a highway- and a roundabout context. Table 4.3.1 is used to generate the corresponding precedence lists shown in Tab. 4.4.1a and Tab. 4.4.1b. Consequently (for the highway case), if the host vehicle is $C2$, its behaviour planner issues a motion goal to stay in the right lane. It is a correct decision given $C1$ approaching from behind. For the roundabout example, given that the host car is $C2$ with entry- and exit point as depicted in Fig. 4.4.1; since Eq. 4.4.1 is false, the behaviour planner issues a motion goal to yield at the entry point.

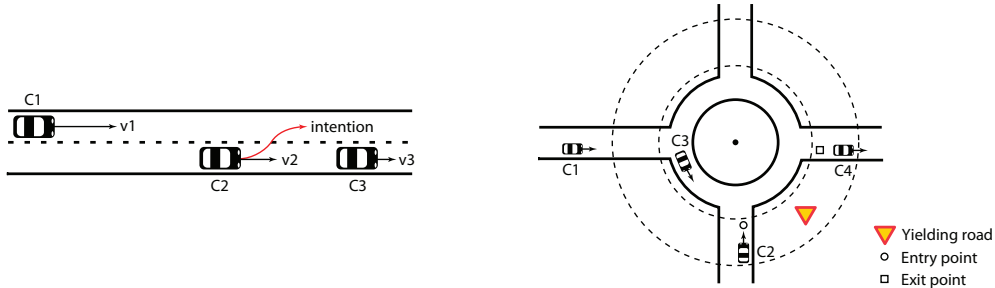


Figure 4.4.1: *Two scenarios are illustrated to exemplify precedence based decision making. **To the left:** A car ($C2$) approaches another car ($C3$) and determines that an overtake is desired. However, the manoeuvre is not possible without interfering with $C1$, hence $C2$ is assigned a priority of 2 and does not have the precedence to answer its desired behaviour positively, as a consequence it stays in the right lane behind $C3$. **To the right:** All vehicles are assigned a priority depending on their position. $C1$ is not included in the list since it is not in the context extent. $C2$ does not have precedence and it therefore not allowed to navigate through the context. It must yield at the entry point of the inner zone (denoted by a ring).*

4.4.2 Cost evaluation for planning of efficient overtake manoeuvres

The precedence list set-up (described in Section 4.4.1) enables safe decisions, regarding how to change lane, during an overtake manoeuvre. However, if the host car is about to enter a traffic context, let's say 500 meters from its current position, the time saved by performing an overtake might not motivate the resulting acceleration profile. This section develops a method that uses costs of acceleration- and velocity deviations from reference values, to make more efficient overtake decisions. As a first approach, overtaking a single vehicle is considered. The result is later used to solve overtaking and merging with a group of vehicles.

Table 4.4.1: Two precedence lists are built for a highway- and a roundabout scenario.

(a) A highway precedence list is built to support the behavioural planner. It is clear that the vehicle *C2* is not allowed to perform its intended behaviour (overtake *C3*).

Highway	
Vehicle	Priority
C1	1
C2	2
C3	1

(b) A roundabout precedence list is used to determine if the host vehicle (here *C2*) should navigate through the inner zone, or yield at the entry point.

Roundabout	
Vehicle	Priority
C2	2
C3	1
C4	2

Figure 4.4.2 illustrates a set of n target vehicles $\{T_1, T_2, \dots, T_n\}$, with corresponding constant velocities v_i , where $i \in \{1, 2, \dots, n\}$. Target T_i is positioned s_i meters in front of a host vehicle

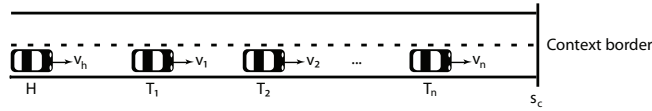


Figure 4.4.2: A host vehicle (*H*) drives behind n targets, in the right lane of an highway strip, leading to a traffic context s_c meters ahead.

(H). All targets are assumed to drive and stay in the right lane of a two-lane highway. The road leads to the border of a traffic context s_c meters from the host vehicle. It is initially assumed that $n = 1$. Hence, two decisions with the following outcomes have to be evaluated:

1. The host car adjusts its velocity and positions itself behind T_1 . Thus the host car's velocity deviates from the set speed v_{set} .
2. The host car changes lane and accelerates to v_{set} , passes T_1 and decelerates to v_c , a velocity suitable for the upcoming context. This decision gives a different distance to drive in order to reach the upcoming context; it might also require large accelerations that affect fuel consumption and passenger comfort. However, it allows the host car to drive closer to v_{set} compared to (1).

The differences between (1) and (2) are taken into account by a cost function

$$\mathcal{J}(a_x, v) = \int_0^{t_f} \kappa a_x^2 + \lambda (v - v_{set})^2 dt, \quad (4.4.2)$$

where κ and λ are tunable weights to value longitudinal acceleration- and velocity deviations. The time it takes, to reach the desired velocity at the traffic context ahead, is denoted t_f . The acceleration- and velocity profiles for (1) and (2) depend on:

- initial velocity v_h of the host car
- set speed v_{set} of the host car
- distance s_i to the target car that is evaluated for overtake
- velocity v_i of the target car that is evaluated for overtake
- distance s_c to the upcoming traffic context
- goal velocity v_c that the host car should use at the upcoming traffic context

Consider a highway example, where a host car should take an exit $s_c = 300$ m ahead, with velocity $v_c = 70$ km/h. The host car finds itself driving at $v_h = 100$ km/h, approaching a target vehicle $s_i = 20$ m ahead that drives at $v_i = 80$ km/h. The prevailing set speed is $v_{set} = 120$ km/h. Figure 4.4.3 depicts a trajectory, and acceleration- and velocity profiles that conform to the two cases, (1) and (2). The profiles were created under the assumption of piecewise constant

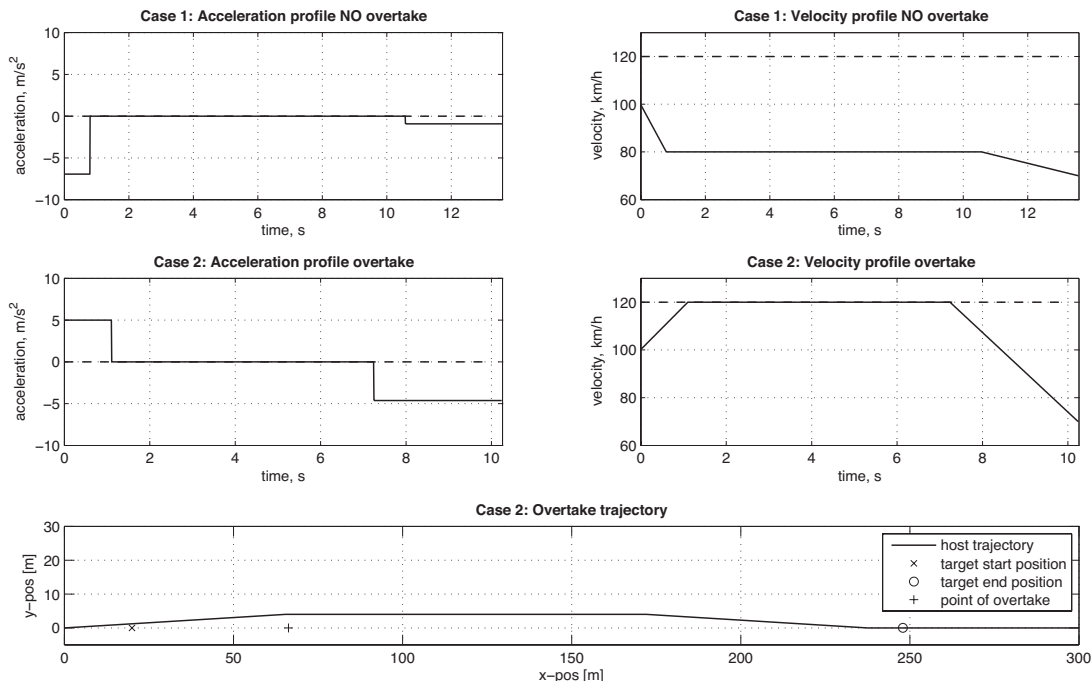


Figure 4.4.3: Acceleration- and velocity profiles (solid) for a highway example. The reference values (dashed) are zero acceleration and velocity v_{set} . Note that t_f differs between the two possible decisions. **Upper figures:** The host car decelerates and adjusts its velocity to the target vehicle in front. It reaches the upcoming exit and adjusts its velocity to 70 km/h. **Middle figures:** The host car accelerates to set speed, overtakes the target vehicle, and decelerates to adjust its velocity to the upcoming exit. **Bottom figure:** The host vehicle's trajectory, for the intended overtake, is given by a ramp shape that varies with the lane width and the host vehicle's velocity profile. Three target vehicle positions are depicted: (x) initial position, (+) position when overtaken by host vehicle and (o) position when host vehicle reaches the goal context ahead.

acceleration, a maximal acceleration of 5 m/s² and a deceleration time of 3 s to reach v_c . The trajectory consist of two ramps that varies by the lane with, host initial velocity and the goal velocity of the upcoming context. Equation 4.4.2 can be used to evaluate the resulting costs of the two cases exemplified, thus telling which decision is the cheapest. However, let's instead consider an example where the context distance is swept from 1000- down to 100 m. Figure 4.4.4 shows how the overtake decision varies with the distance, for two different target velocities. The host car's start velocity is 100 km/h, its set speed is 120 km/h and the target starts 20 m ahead. For this example $\kappa = 1$ and $\lambda = .1$ in the cost function (Eq. 4.4.2). When the decision is YES, an overtake manoeuvre is considered feasible and the host car starts the procedure if it has precedence. As expected, the system considers an overtake feasible, at a closer distance to the upcoming context, if the target car drives at a slower velocity.

The algorithm, described above, of evaluating an overtake decision for a target with respect to an approaching traffic context, can be denoted as a function

$$OE(v_h, v_{set}, s_i, v_i, s_c, v_c). \quad (4.4.3)$$

It returns a boolean value that tells if the overtake should be done (*true*) or not (*false*) from a efficiency perspective. Now, if we let the number of targets $n > 1$, the OE function can be used as a sub-problem solver to evaluate the feasibility to overtake each target car. Thus, enabling

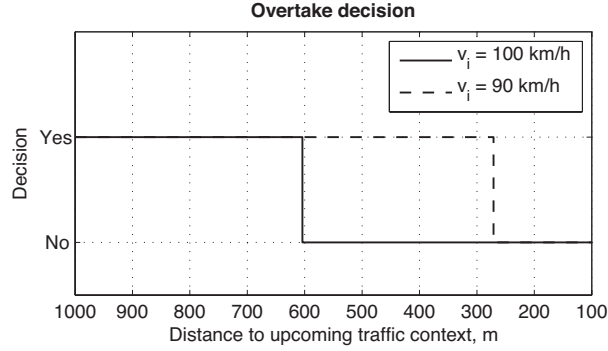


Figure 4.4.4: The positive decision, to overtake a target car, varies with the distance to a context ahead of the host car. Two different target velocities, 100 km/h (solid) and 90 km/h (dashed), are evaluated. As one can expect, the system decides that an overtake is feasible at a closer distance to the upcoming context if the target vehicle is driving slower. The final decision, however, requires that the host car has precedence in addition to having the lowest cost.

the system to decide if-, and in that case, how many target vehicles that should be overtaken (assuming that the host car has precedence). The solution must take two conditions into account, for each target vehicle evaluated:

1. if the target can be overtaken efficiently with respect to the upcoming traffic context
2. if it is possible to efficiently merge with respect to other traffic, after the target has been overtaken.

If both conditions above are true, it is considered efficient to overtake a target vehicle. The evaluation, of $n > 1$ target vehicles, is done for each target T_i , starting from $i = 1$ going to $i = n$ in an ascending order. However, if (1) or (2) is not true for T_i , $i - 1$ is returned as the number of cars that are feasible to overtake. Nevertheless, if (1) and (2) is true for all target cars, n is returned.

This section has previously discussed how condition (1) can be evaluated, using the *OE* function, for a specific target with respect to a context. However, the same function can be used to consider condition (2) as well. Observe the example illustrated in Fig. 4.4.5. The possibility

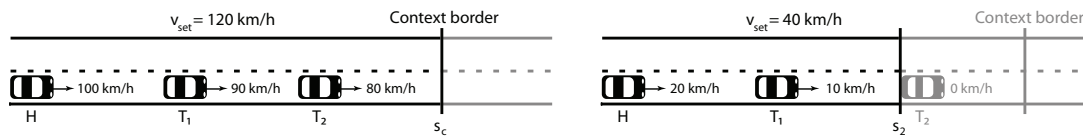


Figure 4.4.5: The feasibility of overtaking T_1 is evaluated in two steps: (1) **To the left:** with respect to the upcoming traffic context. (2) **To the right:** v_2 is subtracted from v_1 and v_{set} to make the calculations relative to T_2 's velocity. The context distance $s_c = s_2$ is used. The problem statement is hence the same as before, i.e. is it possible to overtake T_1 efficiently with respect to the acceleration needed to reach T_2 before T_1 ? Which is solved by the *OE* function.

to efficiently merge between two targets, T_k and T_{k+1} , $k \in \{1, 2, \dots, n - 1\}$, is evaluated by executing the following two steps:

1. Let v_{k+1} of T_{k+1} be used as a reference velocity. Hence, v_h , v_{set} and v_k are specified in relation to v_{k+1} .
2. Call *OE* using the new velocities and $s_c = s_{k+1}$, $v_c = 0$.

The function call generates the acceleration- and velocity profiles needed to evaluate the merge, using the cost function $\mathcal{J}(a_x, v)$ in Eq. 4.4.2, with respect to T_{k+1} . In the worst case, from a computational point of view, all cars in front of the host vehicle are considered feasible to overtake.

As a result, OE is called $2n - 1 \sim \mathcal{O}(n)$ times, i.e. two times for each target, except for T_n that does not have a vehicle in front.

Until now, only two neighbouring vehicles, T_k and T_{k+1} , have been considered when evaluating an overtake merge procedure. However, it is straight forward to include a larger set of target vehicles in front of T_{k+1} . In that way, a better prediction of the merge possibility can be obtained. To exemplify this, imagine a third vehicle that drives relatively slow in front of T_{k+1} . Its presence might force T_{k+1} to brake, in order to avoid collision. Hence, the merge gap between T_k and T_{k+1} is affected. The proposed solution to this is to extend the merge condition to look at all cars in front. For instance, when T_k is evaluated for overtake, the set of targets $\{T_{k+1}, \dots, T_n\}$ should be used as context references, as described above. The resulting algorithm is shown in Alg. 1. In the worst computational case, OE will get called $\frac{n^2+n}{2} \sim \mathcal{O}(n^2)$ times.

```

Data: host, n targets and upcoming traffic context
Result: number of target vehicles to overtake
for  $i=1$  to  $n$  do
  /* Evaluate overtake of target  $i$  w.r.t. traffic context */
  if  $!OE(v_h, v_{set}, s_i, v_i, s_c, v_c)$  then
    | return  $i-1$ ;
  else if  $i==n$  then
    | return  $i$ ;
  /* Evaluate overtake of target  $i$  and merging w.r.t. all cars in front
  of it */
  for  $k=i+1$  to  $n$  do
    | if  $!OE(v_h - v_k, v_{set} - v_k, s_i, v_i - v_k, s_k, 0)$  then
    | | return  $i-1$ ;
    | end
  end
end

```

Algorithm 1: Target vehicles are evaluated, for an eventual overtake manoeuvre, using a function (OE), that solves the sub-problem of overtaking a single target w.r.t. an upcoming traffic context.

4.5 Motion planner

The motion planner’s objective is to generate trajectories that conform to the motion goals issued by the behaviour planner. It operates in two different modes, depending on whether lane markers are present or not. This section describes the two different modes and their output to the execution layer. In general, for both modes, a heading error angle α_e to a point in front of the host is given as lateral output. It is later used to generate a Steering Wheel Angle (SWA). Gehrig and Stein (1998) stress the importance of selecting the point cautiously, from a stability point of view. Selecting a point too close to the car can result in a SWA that makes the car slip, which is not desired. The proposed idea is to not select navigation points closer than a certain lookahead radius r_{la} .

4.5.1 Lane defined lateral motion

A *lane defined mode*, for lateral motion, is used when there are lane markers available, i.e. during highway driving. A heading error angle α_e , to a desired point p_d for lateral navigation, is calculated to make the host vehicle follow a lane reference index $i_{lane} \in \{1, 2\}$, given by the behaviour planner. For a geometry illustrated in Fig. 4.5.1, two points, p_{ls1} and p_{ls2} , that defines the vector \mathbf{v}_{ls} , are used to estimate the road’s heading. The host car’s heading is given by the two points p_{c1} and p_{c2} , that defines the vector \mathbf{v}_c . A curb distance d_c , where

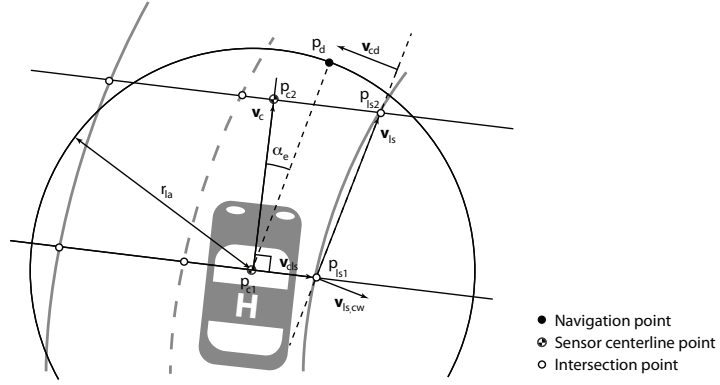


Figure 4.5.1: Lane marker information is used to calculate an heading error angle α_e to a desired navigation reference point p_d . The point is given by a lookahead radius r_{la} and a vector $\mathbf{v}_{cd} \perp \mathbf{v}_{ls}$ whose length d_c varies as a function of the current lane reference index.

$$d_c = \|\mathbf{v}_{cd}\|_2 \quad (4.5.1)$$

$$\mathbf{v}_{cd} \perp \mathbf{v}_{ls},$$

defines a desired lateral position on the road with respect to \mathbf{v}_{ls} . It depends on i_{lane} and the lane width of the road. To avoid feeding discontinuous control signals to the vehicle actuators, a low-pass system S_{LP} is used to smooth d_c variations. Hence d_c is given by

$$d_c = S_{LP} \left(i_{lane} - \frac{1}{2} \right) d_{lanewidth} \quad (4.5.2)$$

Figure 4.5.2 shows d_c for a lane change decision. A lookahead radius r_{la} is used, for stability

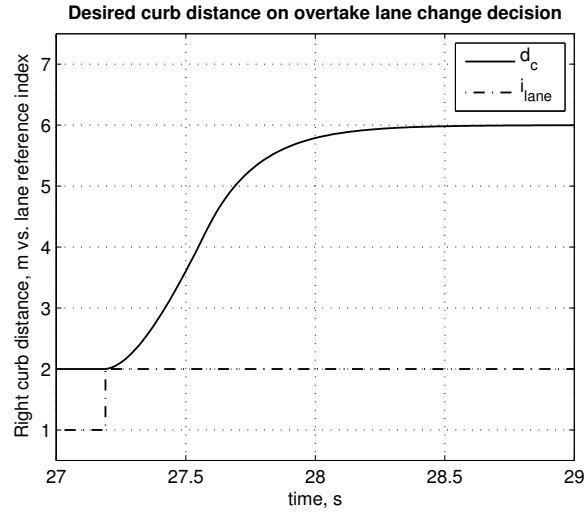


Figure 4.5.2: The behaviour planner issues a reference index change from lane 1 (right lane) to lane 2 (left lane). A smoothing low pass system is used to generate a curb distance d_c . For this particular example, the lane width is 4 meters.

purposes mentioned. Using the geometry in Fig. 4.5.1, α_e is given by

$$\alpha_e = \arcsin \left(\frac{\mathbf{v}_{cls} \cdot \mathbf{v}_{ls,cw} \|\mathbf{v}_{ls,cw}\|_2^{-1} - d_c}{r_{la}} \right) - \arctan_2 \left(\frac{\mathbf{v}_c \cdot M \mathbf{v}_{ls}}{\mathbf{v}_c \cdot \mathbf{v}_{ls}} \right), \quad (4.5.3)$$

where M is a 90 degree clockwise rotation matrix. The first term in Eq. 4.5.3 is the angle to the desired navigation point, with respect to \mathbf{v}_{ls} . The second term compensates for the host car's heading, i.e. rotation in relation to \mathbf{v}_{ls} . The \arctan_2 -function is used to take the vector components' signs into account.

4.5.2 Trajectory-defined lateral motion

For more complex traffic contexts, where lane following is not feasible, a trajectory has to be calculated to navigate from a context entry to an exit. In this work, trajectories for roundabouts and intersections are assumed given a priori. It has therefore been of interest to follow such trajectories.

Similar to the lane-defined lateral motion, the trajectory following algorithm utilizes the notation of lookahead radius. In addition, a scan distance is introduced to enable exclusion of navigation trajectories that are not in the car's proximity. As a result, only points within a scan area A_s , illustrated in Fig. 4.5.3, are considered for the lateral control algorithm. The scan area is defined by

$$A_s = \left\{ \forall p(\gamma, r) : -\frac{\pi}{2} < \gamma < \frac{\pi}{2}, r_{la} < r < r_{scan} \right\}, \quad (4.5.4)$$

where p is a point defined in the polar coordinate frame (γ, r) in Fig. 4.5.3. In the case when no

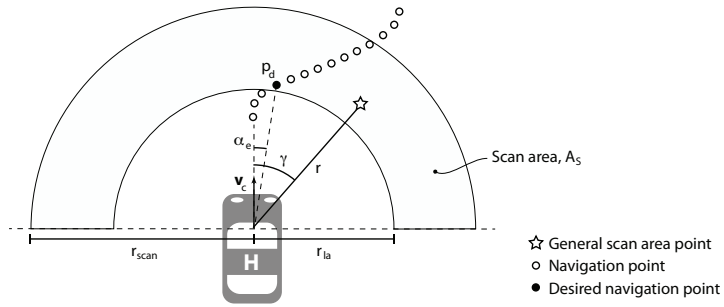


Figure 4.5.3: The algorithm scans an area for navigation points coming from trajectories. If points are found within the scan area, the point closest to the car is selected. An heading error angle α_e between the host car's heading and the selected navigation point is calculated.

trajectory points exist in the scan area, the host car finds itself in its lane defined mode. In the other case, the point closest to the host car, within A_s , is selected. A heading error angle α_e to the desired navigation point, depicted in Fig. 4.5.3, is calculated and given to the execution layer.

4.5.3 Longitudinal motion

Longitudinal motion goals, issued by the behaviour planner, either tell the motion planner to proceed or yield. The motion planner conforms to the longitudinal goals by making a *desired target selection*. That is, sending range- and range rate information, to the lower level execution layer (described in Section 4.6), of an appropriate point. If the desired behaviour is to proceed, a point in the far horizon is selected, hence the host car desires to follow its set speed. In contrast, e.g. if the system should stop at a yield sign, the sign is selected as a desired target.

4.6 Execution of motion goal

The execution layer generates the control signals needed to govern the host vehicle as planned. This section presents its lateral- and longitudinal objectives.

4.6.1 Lateral control

The lateral control objective is to adjust the host car's SWA α_{swa} to navigate against the current reference point (described in Section 4.5). The point is specified by a heading error angle α_e , relative to the car's heading direction. A simple proportional control-law is used to generate the SWA

$$\alpha_{swa} = K_p \alpha_e, \quad (4.6.1)$$

where K_p is a static gain.

4.6.2 Longitudinal control

An Adaptive Cruise Control (ACC) component is used for longitudinal control. It is a system, widely established in industry, for velocity control (Volvo Car Corporation, 2013a; BMW, 2013). Its objective is to follow a set speed. However, if a vehicle is found close enough in front, the ACC component adapts the host car's velocity to it. The process of selecting a specific vehicle, for velocity adaptation, is denoted *target selection*. In this work, a desired target selection is issued by the motion planner, in order to follow a path that fits a desired behaviour. However, if a target vehicle is found within the path, the ACC component uses its own target selection, to avoid collision. This is exemplified in Fig 4.6.1. A host car approaches an intersection from the

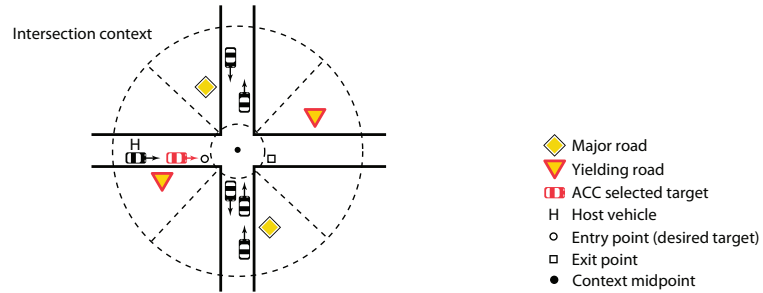


Figure 4.6.1: A host car (H) approaches an intersection from west. A major road with a dense traffic flow crosses the host car's road. The behaviour planner decides that the host car should yield at the upcoming entry point. It is hence selected it as a desired target. However, an intermediate car (red) is in front of the host vehicle, in the desired navigation path. It is therefore selected by the ACC component instead. Thus resulting in the host car stopping behind it before it reaches the entry point to the intersection.

west. A dense traffic flow, on a major road crossing from south and north, results in a decision to yield. Hence, the motion planner of the host vehicle selects the entry point of the intersection as a desired target. However, if the ACC component executes the desired target selection, the host car will probably collide with an intermediate target vehicle in front. The target is said to be in the desired navigation path. In such cases, the ACC selects the *target in path*, instead of the desired target selection issued by the motion planner. Consequently, the host vehicle adjusts its velocity to the preceding vehicle and stops behind it, avoiding collision.

4.7 Results of the automated drive system

This section presents results, from the automated drive system developed in this work. A host vehicle is equipped with the system, as it drives through a traffic environment (described in Section 3.1) developed using PreScan. Similar results can be obtained using CarMaker as modelling tool. The content of this section is ideally visualized as a movie; however, since that is not possible in this format, the results are described sequentially, as the host car navigates through the different traffic situations. The host car's movements over time, overlaid on the road model, is shown in Fig. 4.7.1. Interaction with three target vehicles (A, B, and C) occurs throughout the drive.

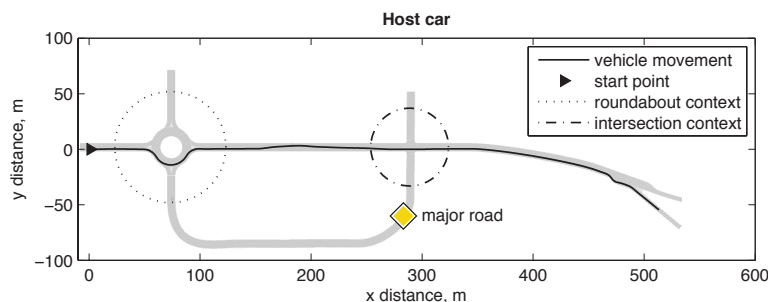


Figure 4.7.1: The automated drive control system successfully governs the host vehicle through a traffic environment. Its movements, throughout the simulation, are depicted above. It yields to a target vehicle in both the roundabout and the intersection. A second target vehicle drives slow in the right lane of the highway between the roundabout and the intersection. It is overtaken by the host vehicle at around $x \approx 180$ m. However, the overtake was initially delayed by a target vehicle that approached from behind.

Yield target (A)

A target vehicle causes the host car to yield at two occurrences, in a roundabout and at an intersection. It drives through the inner zone of the roundabout, as the host car enters the outer zone, and takes the south exit. The target continues, and turns from south to east as it reaches the intersection.

Overtake target (B)

A target vehicle drives relatively slow in the right lane of a highway strip between the roundabout and -intersection. It is overtaken by the host vehicle.

Interference target (C)

The overtake of target B is affected by a target that delays the procedure, by approaching fast from behind when the overtake is desired.

The host vehicle starts from a highway; its current traffic context, throughout the simulation, is depicted in Fig. 4.7.2. It enters a roundabout, at $t \approx 2$ s, and discovers that there are other vehicles operating there; target A drives through the roundabout's inner zone, as the host car enters its outer zone. Consequently, a reflecting precedence list is generated, depicted in Fig. 4.7.3. It tells that the host vehicle does not have the authority to drive through the roundabout. Hence, the resulting behaviour is to yield at the entry point of the roundabout's inner zone, shown in Fig. 4.7.4. Figure 4.7.5 illustrates the host vehicle's set speed and velocity. The decision to yield makes the host car's velocity go to zero at $t \approx 8$ s. At roughly the same time, the host vehicle gets precedence (seen in Fig. 4.7.3) to navigate through the roundabout with a set speed of 15 km/h, given by the perception layer's road model. The host vehicle follows a trajectory, given a priori, through the roundabout and takes the second exit.

A highway strip is reached at $t \approx 22$ s and the host vehicle accelerates towards its new set speed of 70 km/h. By then, target B drives relatively slow in the right lane in front. As a consequence, the host vehicle identifies that an overtake manoeuvre is required at $t \approx 25$ s, depicted in Fig. 4.7.6. Nevertheless, the host vehicle does not have precedence at $t \approx 25$ s nor

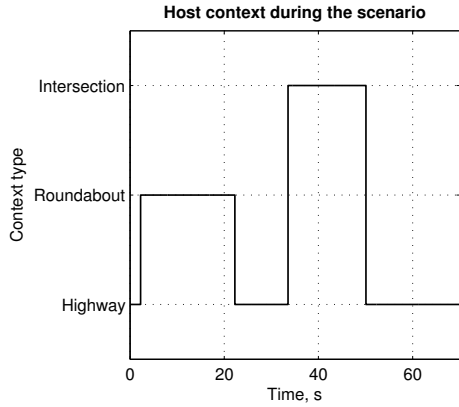


Figure 4.7.2: The host car navigates through three different traffic contexts during simulation, a roundabout, a highway and an intersection.

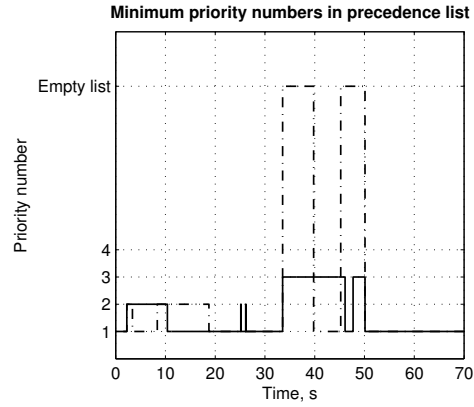


Figure 4.7.3: The host car's priority number (solid) is compared to the minimum priority number of all target vehicles (dashed) in the context. A lower number is a higher priority. The list is said to be empty when there are no target vehicles in the current context. Please note that the priority spikes at $t \approx 25$ s and $t \approx 26$ s solely occurs for the host vehicle (solid line).

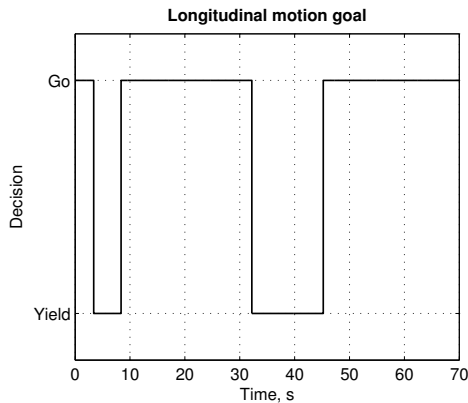


Figure 4.7.4: The behaviour planner decides that the host vehicle should yield at two occurrences, before entering the roundabout $t \approx 4$ s to $t \approx 9$ s and before entering the intersection $t \approx 32$ s to $t \approx 45$ s. The first occurrence is due to rights of way, the section is due to both rights of way and a stop sign.

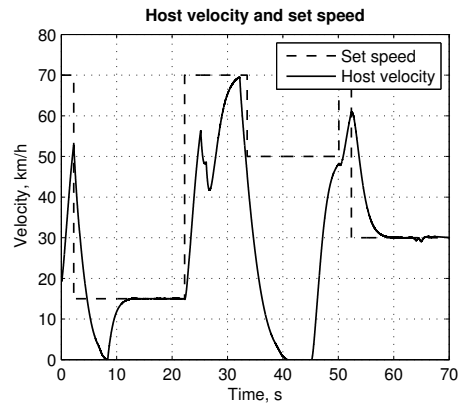


Figure 4.7.5: The automated drive system follows a set speed and adjusts the velocity to desired targets. At $t \approx 9$ s an yield point is selected as desired target, hence the host car stops. Similarly, at $t \approx 40$ s the host car stops at a stop sign next to an intersection. At $t \approx 25$ s, a the host intends to overtake a target vehicle. However, an interfering car approaches from behind and consequently the host adjusts it speed, to the slower driving vehicle in front, until the target car has passed.

at $t \approx 26$ s. This since the desired overtake cannot occur without interfering with target C that approaches from behind. Consequently, the host car stays in the right lane and the ACC component selects target B in front. This can be seen on the host car's velocity, that decreases significantly at the same time. It can also be seen in Fig. 4.7.7 that indicates the presence of a target in the navigation path. However, target C eventually passes the host vehicle and a new overtake request is issued at $t \approx 27$ s. Since an intended lane change does not interfere with

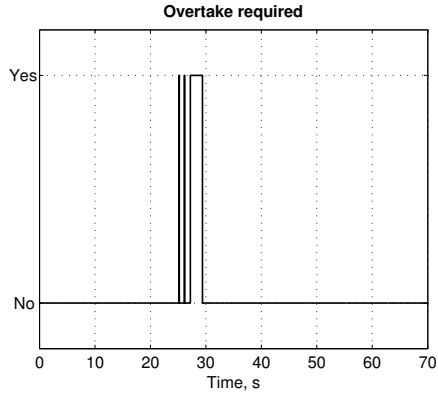


Figure 4.7.6: *It is desired to overtake a target vehicle during highway drive. When the system, at first, answers negatively on the desired overtake, it adjusts the velocity by breaking. The ‘overtake required’ signal is triggered by time-to-collision, hence the time-to-collision increases and for that cause two impulses appear at $t \approx 25$ s and $t \approx 26$ s respectively.*

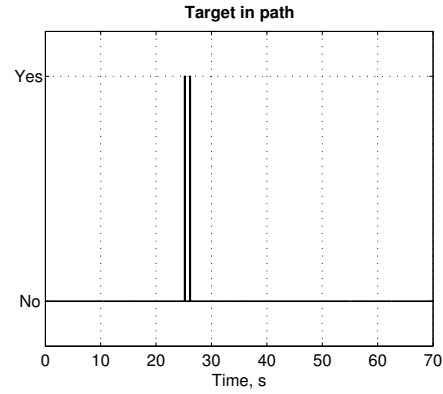


Figure 4.7.7: *The ACC component, within the execution layer, has a built in target selection process. It intervenes if there is a target in the desired navigation path, e.g. when there is a slower driving car in the same lane, in front of the host car.*

other traffic, a motion goal to change lane is issued by the behaviour planner, shown in Fig. 4.7.8. Thence, the host car changes to the left lane, passes target B and changes back to the right lane. A series of images, shown in Fig. 4.7.9, illustrates the process step by step.

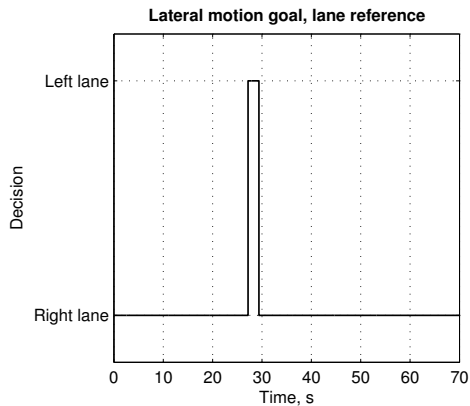


Figure 4.7.8: *The system responds positively on the desired behaviour, to overtake a target vehicle, when it has precedence to change lane.*

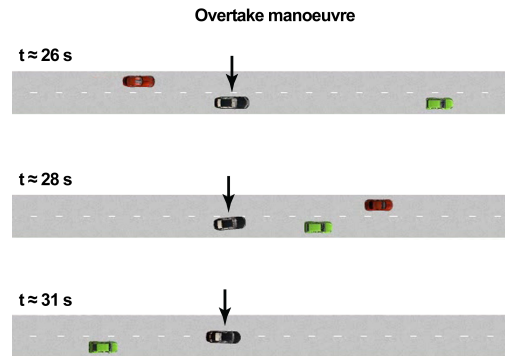


Figure 4.7.9: *The host vehicle (arrow) wish to overtake a target in front. However, a second target, approaching from behind, makes the host vehicle respond negatively on its desired behaviour. When the target passes, the host gets precedence to change lane and overtake the target in front.*

The host vehicle enters an intersection context at $t \approx 33.5$ s, seen in Fig. 4.7.2. A stop sign is detected at approximately the same time, thus an unconditional stop is required, illustrated in Fig. 4.7.10. The system therefore issues a motion goal to yield, depicted in Fig. 4.7.4. The host car’s velocity goes to zero, and it stops in front of the sign. After three seconds, in a stand still pose, the stop requirement is revoked at $t \approx 43$ s. However, target A drives through the intersection from $t \approx 40$ s to $t \approx 45$ s, seen from the precedence list. It enters the intersection from the south, which is a major road, unlike the host car’s entry that connects from the west.

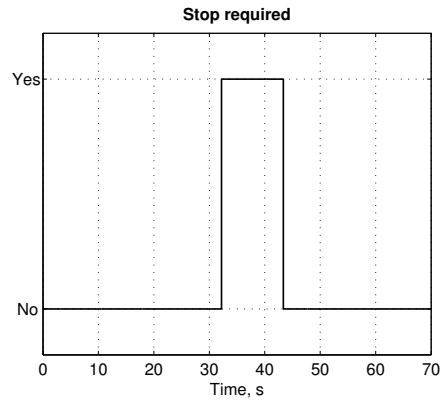


Figure 4.7.10: *The host car approaches a stop sign, at which point the mission planner requires an unconditional stop. As a result, the host vehicle has to stop and stand still for at least three seconds.*

As a result, the host vehicle retains its motion goal to yield, until target A exits the context. The host vehicle regains precedence at $t \approx 45$ s, and drives through the intersection.

The host car reaches a highway context, once more, and starts to accelerate to set speed. At $t \approx 53$ s, a speed sign is detected by the system. Hence a new set speed of 30 km/s is sent to the ACC component, seen in Fig. 4.7.5. Consequently the host car's velocity is adapted to the new speed limit. The highway strip is later left, as seen in Fig. 4.7.1, by following the right curb to take a highway exit.

5 Conclusions

This work has five important results, to the automotive community, concerning automated drive:

1. Prominent strengths of two traffic environment modelling tools

Two traffic environment models have been developed using two capable modelling tools, PreScan and CarMaker. The resulting models have been connected to an automated drive control system developed in MATLAB/Simulink. The study shows the two tools' prominent strengths, concerning modelling and simulation capabilities. The results are beneficial to community members that search for tools to speed up virtual development of automated drive control systems. The different strengths stated, can help to indicate which software that fits specific needs. For instance, if modelling of sensors is desired, PreScan might be a good alternative. If it, on the other hand, is of interest to analyse vehicle dynamics, one might find CarMaker a good alternative. However the findings are restricted to the scope of this work. The result should hence be used with this in mind.

2. A modular framework for automated drive

An automated drive framework is implemented in this thesis. It has shown to be a very capable structure, that divides tactical planning from decision making and control. Hence development of subsystems can be performed independent of each other, which might be beneficial to large development projects.

3. Road model parametrization for context based planning

This work has developed a road model parametrization, to enable the planning layers to have access to the current traffic context and prevailing -regulations. In that way, the system can prepare-, make-, and execute decisions based on the current traffic situation.

4. Automated drive using an Adaptive Cruise Control component

An Adaptive Cruise Control (ACC) component was used to execute a desired navigation path. However, the ACC component also intervenes, by adjusting the host car's velocity, if target vehicles are found within the path. Hence collisions are avoided.

5. Efficient overtaking by evaluating decision costs

This work develops an algorithm for cost based overtake, of n target vehicles driving in the right lane, with respect to an upcoming traffic context and merging possibilities. The algorithm values acceleration- and velocity profiles of different possible decisions.

Simulation results agree with the desired behaviour. The system drives the host vehicle through a variety of traffic contexts; such as a roundabout, an intersection and a highway, in a safe and efficient manner, while interacting with other vehicles operating in the traffic environment. It consequently yields to target vehicles that have precedence, and overtakes target vehicles when considered efficient. In addition, the system handles different types of traffic signs, such as stop- and speed signs, and rights of ways. It has been assumed that the control system has access to all relevant target vehicle information, at all times. The implications of this assumption are however considered small, or at least manageable. In fact, the automated drive system only considers targets that are in the proximity of the host vehicle. If such targets are obscured, e.g. by a tree in an intersection, the framework could be extended to drive conservatively and assume that there are targets behind the tree. In that way, the resulting behaviour might be to yield until sufficient environment information is given. If it is desired to implement the system in a real vehicle, I advice future work to concern generation of trajectories, in complex traffic contexts, that conform to motion goals issued. In addition, generation of the precedence list can be extended to include more traffic situations and -regulations. The road model, held by the perception module, can be adjusted to fit a relevant map provider.

References

- BMW (2013). *BMW Technology Guide. Active Cruise Control*. [Online; accessed 5-mar-2013]. URL: http://www.bmw.com/com/en/insights/technology/technology_guide/articles/active_cruise_control.html.
- DARPA (2013). *Overview. What is the Urban Challenge*. [Online; accessed 5-mar-2013]. URL: <http://archive.darpa.mil/grandchallenge/overview.asp>.
- Gehrig, S.K. and F.J. Stein (1998). “A trajectory-based approach for the lateral control of car following systems”. In: *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*. Vol. 4. IEEE, pp. 3596–3601.
- Gietelink, O.J. (2007). *Design and validation of advanced driver assistance systems*.
- HAVEit (2012a). *About HAVEit*. [Online; accessed 7-nov-2012]. URL: <http://www.haveit-eu.org>.
- (2012b). *Highly Automated Driving*. [Online; accessed 13-mar-2013]. URL: <http://www.haveit-eu.org/displayITM1.asp?ITMID=5&LANG=EN>.
- interactIVe (2012). *Accident avoidance by active intervention for Intelligent Vehicles*. [Online; accessed 7-nov-2012]. URL: <http://www.interactive-ip.eu>.
- IPG Automotive (2013). *CarMaker. Shift the test drive into simulation!* [Online; accessed 5-mar-2013]. URL: <http://www.ipg.de/index.php?id=609>.
- Kammel, S. et al. (2008). “Team AnnieWAY’s autonomous system for the 2007 DARPA Urban Challenge”. In: *Journal of Field Robotics* 25.9, pp. 615–639.
- MathWorks (2013). *Simulink. Simulation and Model-Based Design*. [Online; accessed 5-mar-2013]. URL: <http://www.mathworks.com/products/simulink>.
- Rajamani, Rajesh (2012). “Adaptive Cruise Control”. In: *Vehicle Dynamics and Control*, pp. 141–170.
- Tass (2013). *PreScan product information*. [Online; accessed 5-mar-2013]. URL: <http://www.tass-safe.com/en/products/prescan>.
- Thrun, S. et al. (2007). “Stanley: The robot that won the DARPA Grand Challenge”. In: *The 2005 DARPA Grand Challenge*, pp. 1–43.
- Urmson, C. et al. (2008). “Autonomous driving in urban environments: Boss and the urban challenge”. In: *Journal of Field Robotics* 25.8, pp. 425–466.
- Urmson, Chris et al. (2009). “Autonomous driving in traffic: Boss and the urban challenge”. In: *AI Magazine* 30.2, p. 17.
- Volvo Car Corporation (2013a). *Safety Technologies. Adaptive cruise control down to standstill*. [Online; accessed 5-mar-2013]. URL: <http://yourfleet.volvocars.com/preview/safety-technologies.aspx>.
- (2013b). *Vision 2020*. [Online; accessed 7-nov-2012]. URL: <http://www.volvocars.com/intl/top/corporate/volvo-sustainability/safety/pages/vision-2020.aspx>.