MAVEN

Managing Automated Vehicles Enhances Network

Deliverable 3.3:
Cooperative manoeuvre and trajectory planning algorithms

Version	Date	Release	Approval
07	28.02.2019	Reza Dariani	Klaas Rozema & Julian Schindler

Document Log

Version	Date	Comments	Name and Organisation
01	04.12.2018	Initial version- document structure	Reza Dariani, DLR
02	19.02.2018	DLR input	Reza Dariani, DLR
03	25.02.2018	DLR intern review	Julian Schindler, DLR
04	27.02.2018	HMETC input	Michele Rondinone, Thomas Walter, Dominik Matheis. HMETC
05	28.02.2018	Dynniq Review	Robbin Blokpoel, Klaas Rozema, Dynniq
06	28.02.2018	Pre-Final version	Reza Dariani, DLR
07	02.03.2019	Corrections	Julian Schindler, DLR

Authors

Beneficiary nº (1 to 5)	Name and Organisation	Country
1	Dariani Reza	Germany
2	Julian Schindler	Germany
3	Michele Rondinone	Germany
4	Thomas Walter	Germany
5	Dominik Matheis	Germany

Table of Contents

List of Figures

Abbreviations and definitions

Abbreviation	Definition
AD_SW	Automated Driving Software
AGLOSA	Adaptive Green Light Optimized Speed Advisory
AV	Automated vehicle
C2X	Car to X (communication)
CAM	Cooperative Awareness Message
CPM	Collective Perception Message
DGPS	Differential Global Positioning System
DLR	German Aerospace Centre
DMM	Decision Making Module
ECU	Electronic Control Unit
FASCar	Name of DLR automated vehicle
GLOSA	Green Light Optimized Speed Advisory
GNC	Guidance, Navigation, Control
GPS	Global Positioning System
GRP	Global Route Planner
HAD	Highly Automated Driving
HMETC	Hyundai Motor Europe Technical Centre
I2V	Infrastructure to Vehicle (communication)
IMU	Inertial Measurement Unit
LAM	Lane Advice Message
LC	Lane Change
LF	Low Frequency
MP	Motion Planner
OCP	Optima Control Problem
ODE	Ordinary Differential Equation
OEM	Original Equipment Manufacturer
PP	Path Planner

PRM	Probabilistic Road Map
ROS	Robot Operating System
RRT	Rapidly-exploring Random Trees
RSU	Road Side Unit
SOA	Service Oriented Architecture
SPAT	Signal Phase and Time
SQP	Sequential Quadratic Programming
UDP	User Datagram Protocol
V2X	Vehicle to X (communication)
VC	Vehicle Control
VSE	Vehicle State Estimator
WME	World Model Events
WP	Work Package

Executive summary

This deliverable is the textual description of the cooperative manoeuvre and trajectory planning algorithm which have been designed and implemented by DLR and HMETC for MAVEN.
An overview of different modules of vehicle automation is given. The trajectory planning and tactical and strategical modules and their roles on cooperation are explained. Furthermore, cooperation with infrastructure in the form of *AGLOSA* and lane change and cooperation with other automated vehicle in the form of platooning is described. Functionality of the algorithms is approved by simulation results, close field tests and urban tests in close relation to WP4, WP6 and WP7.

1 Introduction

This document contains a detailed description of the cooperative manoeuvre and trajectory planning algorithm implemented in MAVEN's WP3. To validate the functionality of the implemented algorithms, simulation results, test field results, as well as, urban area results are presented in this document.
This document consists of the following chapters:
Chapter 2 deals with vehicle automation and describes the developed and implemented vehicle automation sub-modules.
Chapter 0 is about manoeuvre and trajectory planning. It describes the approach and the implementation of the trajectory planner and the tactical and strategical modules, as well as their role in cooperation.
Chapter 4 concludes this deliverable.
As HMETC and DLR use a different vehicle architecture, different sensors and a different software framework, the concepts design and implementation for DLR and HMETC are done mostly separated. Therefore, the specific implementations of both are described in different sections. Nevertheless, cooperation of both is possible as both developments are bound to a common communication protocol and common message definitions. On top, the Platoon Logic part developed by DLR is used in the HMETC car as well, assuring correct platooning behaviour, as described in D3.1 [1].

2 Vehicle automation

In MAVEN deliverable D3.1 [1] vehicle architecture on module level has been described for DLR and HMETC. All MAVEN automated vehicles follow a common basic software architecture ("*AD* software"), composed of interfaces to the sensors and to a high definition map, a sensor data fusion, and modules for trajectory planning and vehicle control, as shown in Figure 1. In addition, a common module "Platoon Logic" developed by DLR, which is handling all platoon related procedures and information, e.g. the process of platoon forming and of platoon breaking in case other vehicles need to change lane to the lane of the platoon, is attached to the *AD* software. While the *AD* software differs in implementation between DLR and Hyundai vehicles, the Platoon Logic has been implemented as a common library for both.

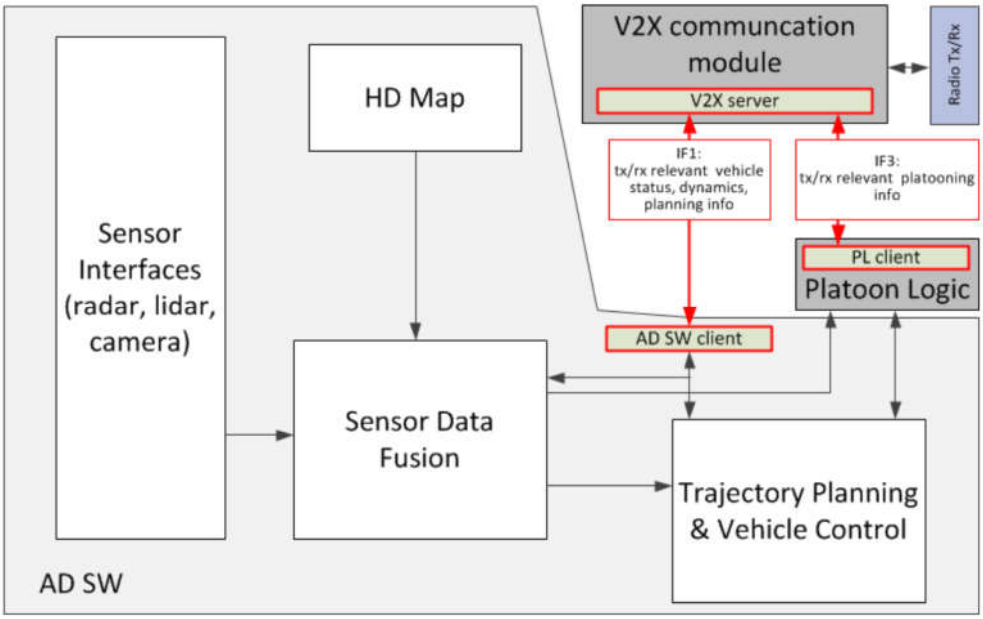

Figure 1: General vehicle automation architecture

2.1 DLR

DLR vehicle automation architecture is illustrated in Figure 2. As it is shown, it has three modules, tactical decision, trajectory planner and vehicle control.

- **Tactical decision:** In an optimal control based approach, the characteristic of a planned trajectory is an objective function. On another hand, defining a generic objective function which satisfies all driving situations and conditions is not feasible. Therefore, based on the forehead situation such as road geometry, obstacle information etc., driving strategy can be defined. Based on the defined strategy, an Optimal Control Problem *OCP* can be reformulated. As an example, maximum velocity is an important parameter which can be taken from the road from traffic signs or can be defined based on the driving area (Urban, Highway). But traffic network dynamics can also impact the driving speed. Therefore, a safe and feasible speed based on the current situation must be defined. Further consideration is not limited only to driving velocity but to reformulation of *OCP* such as the dynamic definition of the feasible region in *OCP*. Other examples are a far traffic light showing a red

phase which is not inside optimization horizon, or an advised velocity from the *AGLOSA* system, or any action needed for the platooning scenario, e.g. braking or accelerating vehicles and desired gaps.

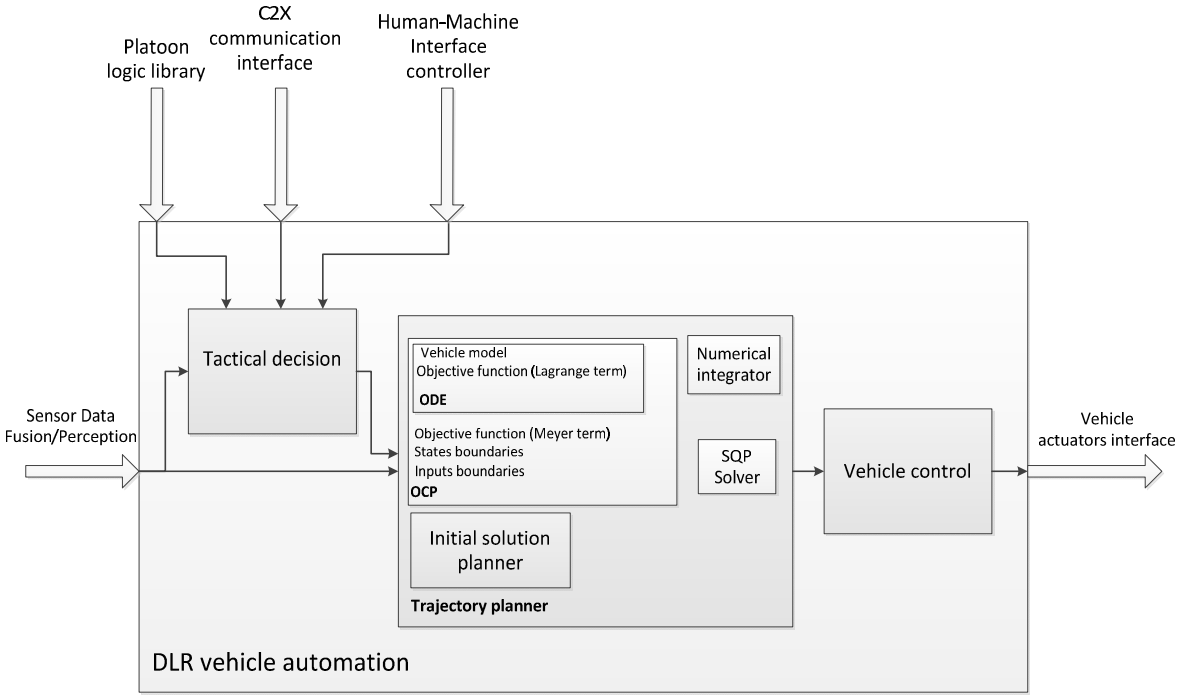

Figure 2: DLR vehicle automation

- **Trajectory Planner:**
  - *OCP*, *ODE* and *SQP* solver: As already mentioned [1], an optimal control based approach is used to plan a trajectory. A Sequential Quadratic Programing *SQP* method solves the optimization problem as minimizing an objective function f under nonlinear equality and inequality constraints as

$$x \in \mathcal{R}^n: \quad \begin{aligned} &\min f(x) \\ &g_j(x) = 0 \qquad j = 1, \dots, m_e \\ &g_j(x) \geq 0 \qquad j = m_e + 1, \dots, m \\ &x_l \leq x \leq x_u \end{aligned} \qquad (1)$$

$f_{(x)}$ is the objective function with the vector x , and function $g_j(x)$ represents the equalities and inequalities and x_l and x_u represent lower and upper boundaries respectively. The above description of an optimal control problem can be written in the following form. Reformulation is described in detail in [2] and [3].

$$\min_{\underline{x},\underline{u}} J(\underline{x}(t_f), \underline{u}(t_f)) \qquad (2)$$

with dynamic system and nonlinear constraints as

$$\dot{\underline{x}} = f(\underline{x}, \underline{u}) \quad (3)$$

$$g_l \leq g(\underline{x}, \underline{u}) \leq g_u \quad (4)$$

as well as states and input restrictions as

$$\underline{X_l} \leq \underline{x} \leq \underline{X_u} \quad (5)$$

$$\underline{U_l} \leq \underline{u} \leq \underline{U_u} \quad (6)$$

J refers to the objective function, $\underline{x}$ and $\underline{u}$ refer to states and inputs respectively, g to nonlinear constraints and index l and u to lower and upper value respectively. In equation 2, the Meyer term, considers states and inputs at final time t_f. To deal with the objective function which considers the complete optimization horizon, the objective function of equation 2, can be written as equation 7 in which the integral part, also called Lagrange term, is defined as an extra state inside the Ordinary Differential Equation *(ODE)* of the dynamic system.

$$J(\underline{x}, \underline{u}) = J\left(\underline{x}(t_f), \underline{u}(t_f)\right) + \int_{t_0}^{t_f} f_J\,(\underline{x}(t), \underline{u}(t))dt \quad (7)$$

- Numerical integrator: To solve the ordinary differential equation of the optimal control problem, an explicit Runge-Kutta method is used. Figure 3 illustrates a planning horizon τ with equidistance time horizon Δt. At each iteration, after planning a trajectory for a given time horizon τ, a portion of it, ξ, is sent to the vehicle controller. Then the *OCP*, based on the new sensor information, is updated and planning process is repeated for the new time horizon.

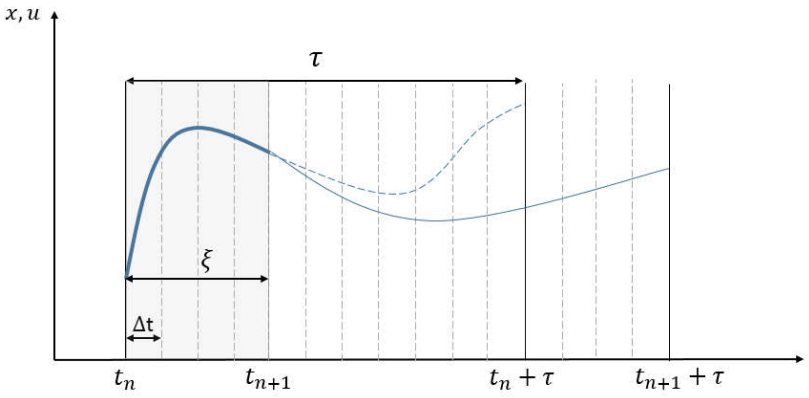

Figure 3: Moving-horizon approach

- Initial solution planner: As the optimization method used in the solver belongs to the class of quasi-newton methods, it requires a starting point or initial solution. This sub-module is explained in more detail in the next chapter.

- **Vehicle control:** The vehicle model used inside the trajectory planner is simplified (vehicle *ODE*) and it cannot reflect the complete behaviour of the real vehicle. On the other hand it

is not possible to consider all external disturbances and effects inside the trajectory planner due to their complexity and calculation time expenses. Hence, the driven trajectory after applying the actuator values, found by the trajectory planner, does not match the planned trajectory. Therefore, a closed loop controller is used to minimize the error and difference between calculated trajectory and the driven one.

2.2 HMETC

Hyundai's test vehicle framework running the MAVEN functions is based on the commonly used *ROS* (short for Robot Operating System), which is taking care of the communication between several so called nodes (sending and/or receiving endpoints/control units/functions), scheduling and maintenance tasks. On top of this base framework, control logic and specific MAVEN functions are integrated to fulfil the required MAVEN use cases like manoeuvring control upon road infrastructure advisories, handling of collective perception information or platooning.
As described in Deliverable D3.2 [4] and highlighted in Figure 4 and Figure 5 the sensor fusion module collects inputs from the individual sensors (including the *V2X* communication module) and provides a consolidated representation of the environment to the Guidance, Navigation and Control module (*GNC*).
The *GNC* module is devoted to compute the planned trajectory and derive motion objectives to be converted into vehicle control signals. In this block, the Decision Making Module (*DMM*) can support a threat assessment based on the vehicle route and the obstacles detected in the drivable region. As such, it is used to drive deceleration/stopping, as well as lane change decisions. For this purpose, the decision making module takes as inputs the list of detected and tracked objects as well as the list of lanes information from the sensor fusion module. This information is crossed with the ego vehicle state (heading, speed, position, etc.) from the Vehicle State Estimator (*VSE*) and with the intended route as received from the global route planner module. Moreover, the *DMM* can receive triggers to adapt the vehicle speed or change the lane based on *V2X* receptions from the infrastructure and has to also consider the presence of traffic signs, speed limits, etc. from *HAD* maps. Based on all these inputs, the *DMM* generates two outputs: a so called "feasible manoeuvre" and an "object threat list". A feasible manoeuvre such as lane change, go straight, stop, keep distance and possible associated speed and or distance values are computed based on the priority of the various inputs like intended route, object list, *C2X* speed and lane change advice, traffic signs from *HAD* map data, etc.

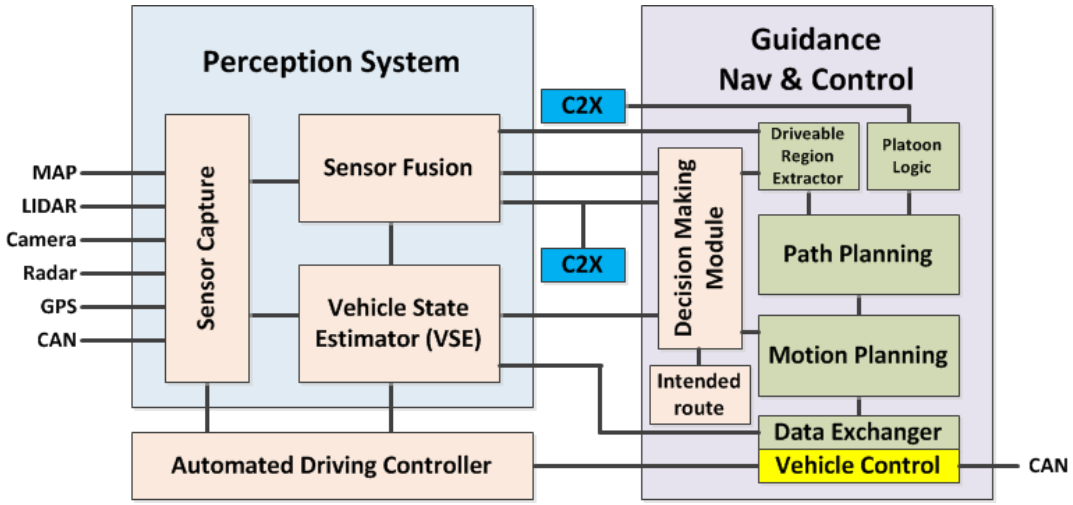

Figure 4: HMETC AD_SW building blocks (Perception and Guidance, Navigation and Control)

The object threat list is a set of objects whose position and dynamics currently constitute a threat (e.g. risk of collision) when compared with position and dynamics of the ego vehicle. Feasible manoeuvre and object threat list are given as inputs to the path and motion planner module. Based on them, the path planner module continuously computes a reference goal point and, by exploring all the possible paths to reach it, selects the most suitable one consisting of a set of intermediate waypoints. Finally, the motion planning module takes the waypoints as input and translates them into an objective position, speed and heading. The output of this motion planning is the input for the vehicle automation function (actuator control) enabling the vehicle to reach the target in a safe a comfortable way.
The platooning logic is also interfaced with the path planner and sensor fusion module in order to collect information needed to be exchanged via *V2X* with other cooperative automated vehicles, and accordingly enable the calculations of the platooning state machine.

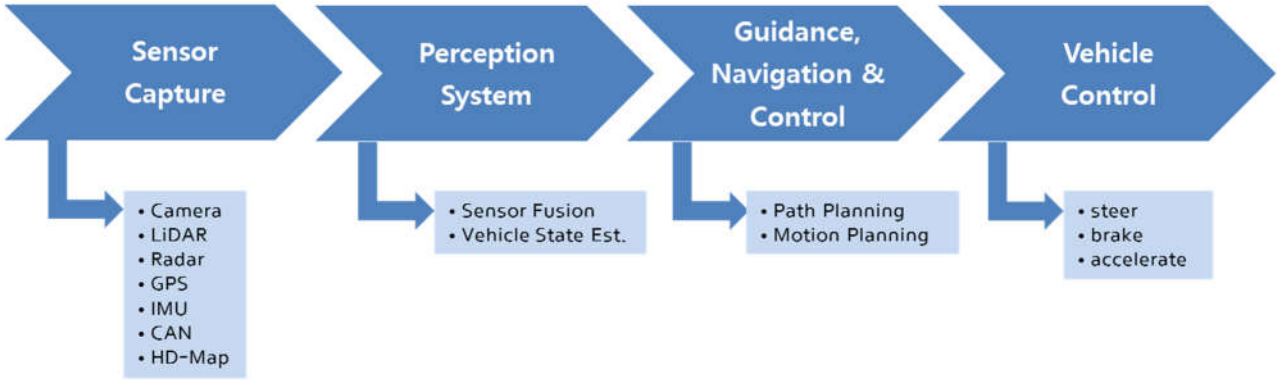

Figure 5: HMETC data processing approach "sense, act & control"

3 Manoeuvre and trajectory planner

In this chapter the manoeuvre and trajectory planner and its cooperation with other cooperative vehicles and infrastructure is explained.

3.1 DLR

As mentioned in the previous chapter, an optimal control based approach is used to plan a trajectory in DLR vehicles. The optimization method used in the solver belongs to the class of quasi-newton methods, hence requires an initial solution. While 3.1.1 describes the initial solution planner, 3.1.2 explains the trajectory optimization. Finally, 3.1.3 gives an overview on the Tactical decision and cooperation of the vehicle automation.

3.1.1 *Initial solution planner*

In order to generate an initial solution, a path from current position to destination is needed. There are different methods such as Rapidly-exploring Random Trees *RRTs* [5] or Probabilistic Roadmap *PRM* which are efficient methods to find a path in unconstructed environment. But to plan a path in a structured environment, based on high precision digital map, a set of discrete points (x_p, y_p) from road center line, as shortest path, can be used instead of using graphs [3].
A simple kinematic vehicle model, as shown in Figure 6, is used to convert these discrete points to the initial solution, here steering angle and driving force.

$$\underline{u_{init}} = [\delta_{init}, F_{D\,init}]^T \quad (8)$$

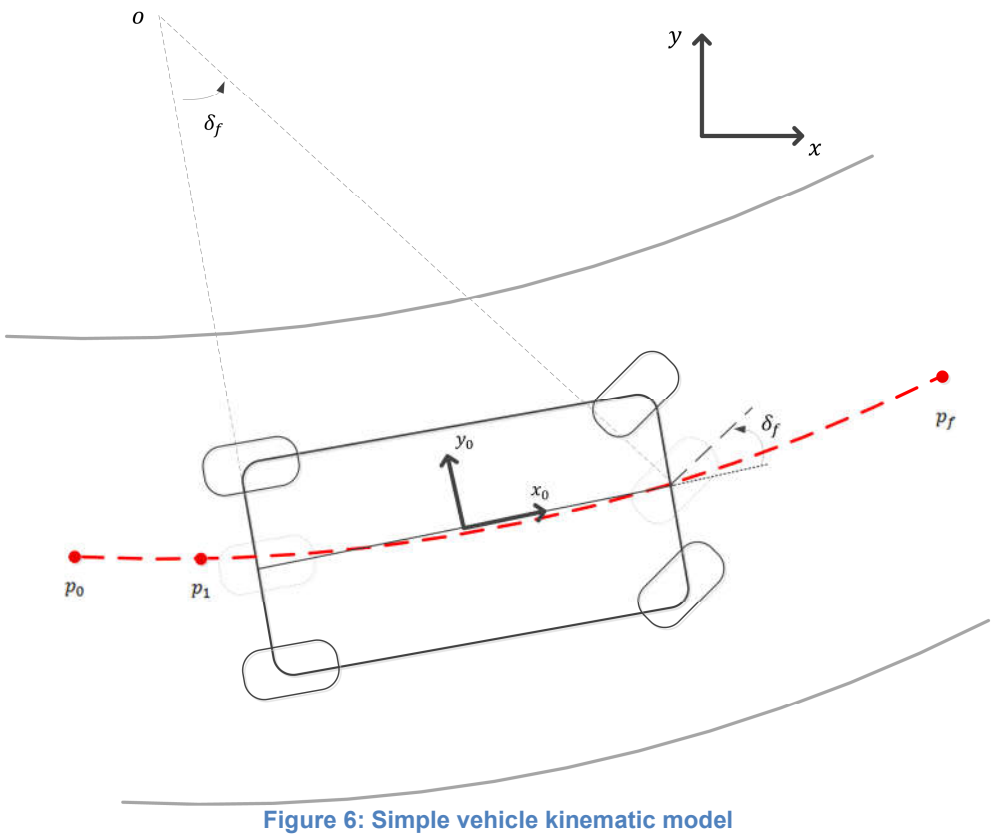

Figure 6: Simple vehicle kinematic model

3.1.2 *Trajectory optimization*

3.1.2.1 *Vehicle model*

To describe the vehicle's motion, a single track bicycle model is used, as shown in Figure 7, in which steering angle and driving force are input of the system, see equation 9. The vehicle is regarded as a rigid body moving in xy-plane. To simplify the model, front and rear wheels are summarized to one single wheel each and roll and pitch angels as well as tire characteristic are neglected. The system states, shown in equation 10, are coordinated at the centre of gravity x and y in a vehicle global coordinate system. The yaw angle ψ describes vehicle orientation. v is the vehicle velocity and β the vehicle side slip angle. s is the travelled distance as integral of velocity. This parameter is used to read the relevant data from digital map based on the current vehicle position.

$$\underline{u} = [\delta, F_D]^T \quad (9)$$

$$\underline{x} = [x, y, \psi, v, s]^T \quad (10)$$

Equation 11 is set of equations of motion of a single track model used in Ordinary Differential Equation *ODE* of trajectory optimization.

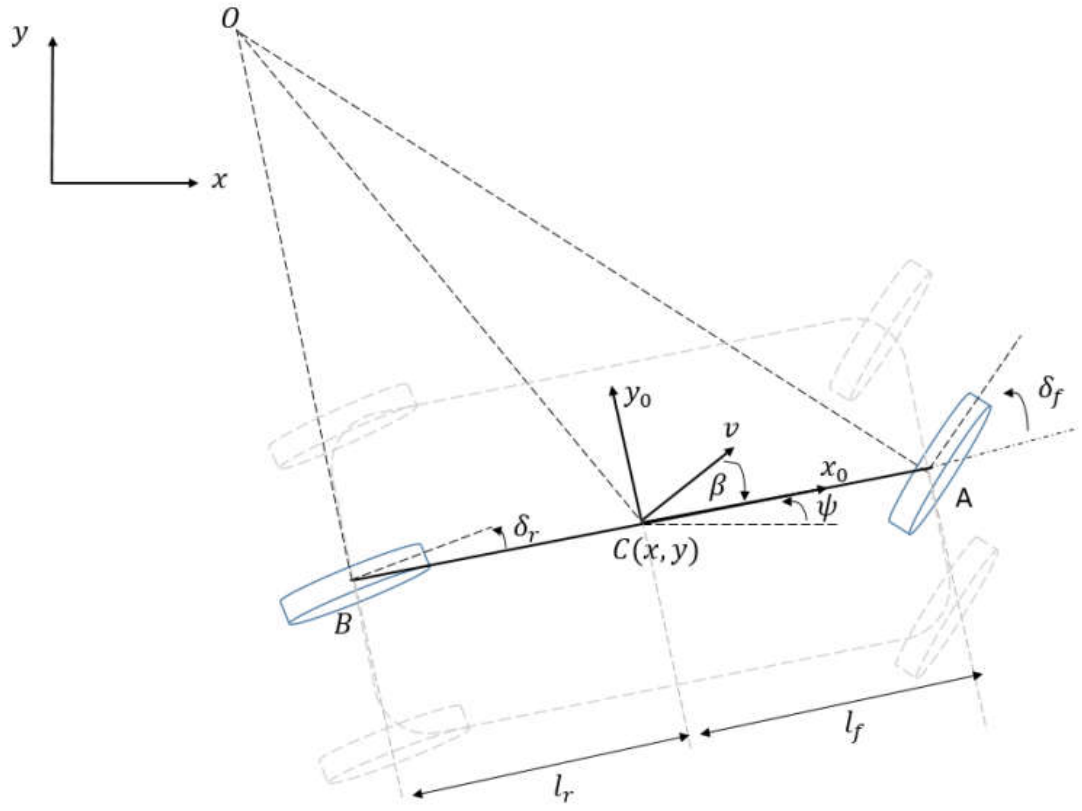

Figure 7: Vehicle single track model

$$\dot{x} = v.\cos(\psi - \beta)$$
$$\dot{y} = v.\sin(\psi - \beta)$$
$$\dot{\psi} = \frac{v.\cos(\beta)\tan(\delta)}{l_f + l_r} \quad (11)$$
$$\dot{v} = \frac{1}{m}(F_D - F_L)$$
$$\dot{s} = v$$

3.1.2.2 *Objective function definition*

As equation 5 and 6 show, inequalities and equalities can be considered inside the optimal control problem. This form of consideration has the advantage of being checked at each iteration, in order to find the control values which satisfy the inequalities and equalities. On another hand the considered parameters must be exact at each iteration. As the equations show, the validity is only guaranteed, when the inequalities and equalities are satisfied, otherwise the solution is invalid and a new solution must be found. This kind of definition is called “hard constraints”. Another way to define equalities and inequalities is to define them as penalty function inside the objective function. Advantage is that the optimal solution of the objective function also satisfies the inequality and equalities. Another advantage is that the border of validity-invalidity is not sharp as hard constraints and a smooth margin (two times differentiable) can be defined. Figure 8 illustrates hard constraints and soft constraints. Driving at the centre line of the road, not exceeding the road boundaries, driving with a desired velocity etc. can be formulated in the form of a penalty function as soft constraints in the Lagrange term.

Equation 12 describes the road boundaries penalty function which has two parts. The first part is a penalty function which results in driving at the centre line and the second part keeps the vehicle within the road boundaries. α is the lateral distance between vehicle and lane center line which is a function of the vehicle position and road centre line and rb_u and rb_l are road upper and lower boundaries respectively. $\mathcal{W}_{cl}$ and $\mathcal{W}_{offs}$ are weight coefficients.

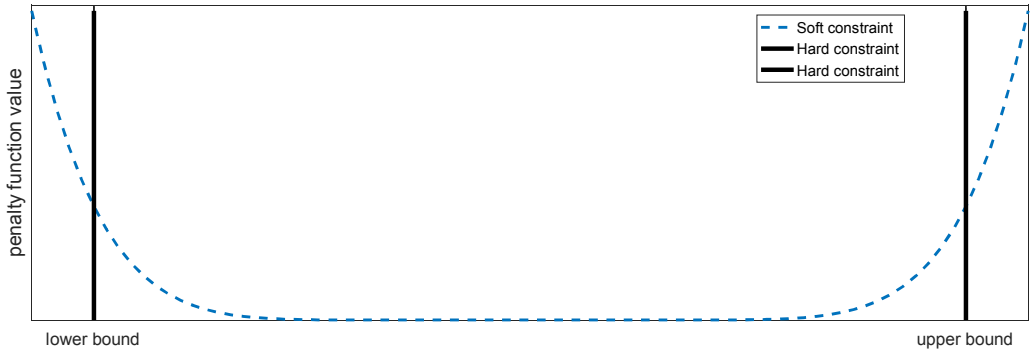

Figure 8: Hard constraints vs. soft constraints

$$J_{rb}(\underline{x},\underline{u}) = \mathcal{W}_{cl}\int_{t_n}^{t_n+\tau}\alpha(\underline{x},s)^2 + \mathcal{W}_{offs}\int_{t_n}^{t_n+\tau} d\left(\alpha(\underline{x},s), rb_u(s), rb_l(s)\right)^2 \quad 12$$

Equation 13 is the penalty function with results in driving with desired speed. Desired speed is calculated by the tactical decision module in each iteration.

$$J_v(\underline{x},\underline{u}) = \mathcal{W}_v\int_{t_n}^{t_n+\tau}(v - v_{des})^2 \quad 13$$

Equation 14 is the Meyer term of the objective function with is to maximize the travelled distance within a horizon. This term is added to avoid stand still behaviour, as stand still in some cases satisfies most of the objective function terms.

$$J_s(\underline{x},\underline{u}) = -\mathcal{W}_s\big(s(t_n+\tau) - s(t_n)\big) \quad 14$$

The objective function as explained in equation 7 can be written as the sum of the Lagrange and Meyer terms explained above.

$$J\left(\underline{x}(t_f),\underline{u}(t_f)\right) = J_s(\underline{x},\underline{u}) + J_v(\underline{x},\underline{u}) + J_{rb}(\underline{x},\underline{u}) \quad 15$$

3.1.3 *Tactical decision and cooperation*

Infrastructure plays a crucial rule in MAVEN. It orchestrates the traffic network especially at intersections by sending Lane Advice Messages (LAM) and AGLOSA messages to the automated vehicles. Therefore vehicle automation must react properly to the messages received from the infrastructure. On another hand an automated vehicle cooperates with other automated vehicles, specific in MAVEN are the platoon use cases [1]. This latter is done in the tactical decision module. Another important point to be mentioned, as Figure 3 illustrates, longer horizon, and more frequent OCP updates are required in complex scenarios but due to the complexity of the approach the choice of horizon and update frequency is limited [3]. But the strategical decision can be taken independent from the *OCP* horizon. Figure 9 as an example illustrates two situations: one is a pedestrian crossing the street and the second, a turn left situation but both out of the planning horizon. In both situations, tactical decisions by using data from sensor-data fusion and digital map/navigation can reformulate the *OCP* by suggesting a safe velocity.

3.1.3.1 *Lane change*

As already mentioned, lane changes can be triggered from the infrastructure by sending a Lane Advice Message. In processing this, the tactical decision module is analysing possible gaps by using information about the objects on the requested lane. Based on the speed, distance and time headway of the following and preceding vehicle of each gap, a cost value is assigned to a gap. Then the gap with minimum cost is selected and lane change request is sent to *OCP*. Due to its dynamics the gap selection and its independency to ego planning, is not a convex problem, therefore the best gap selection is done not only by gap cost, but also by gap consistency. Figure 10 illustrates ego vehicle and different gaps on the left lane.

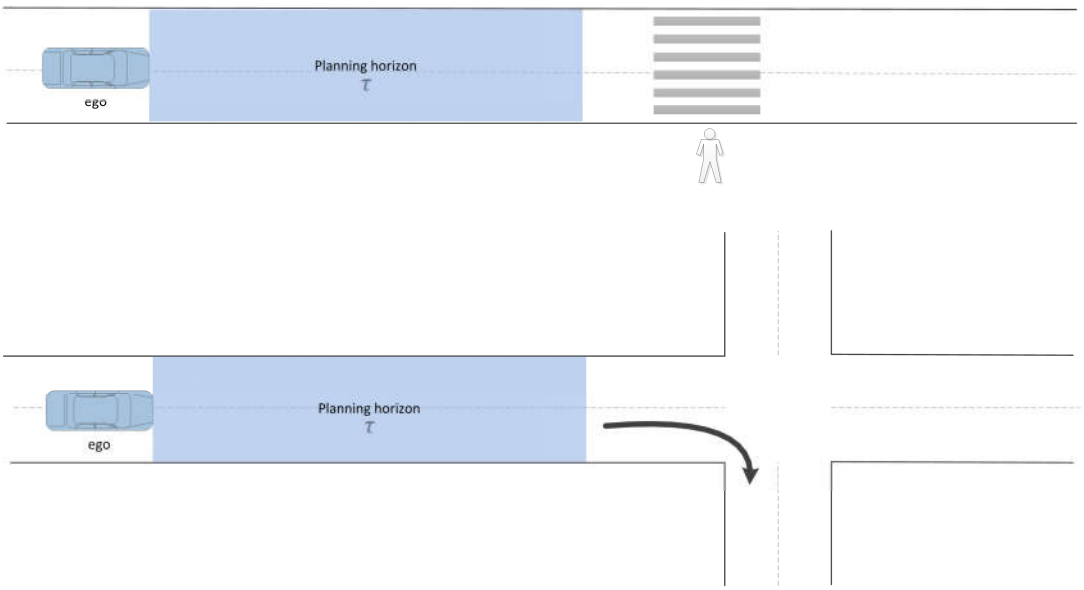

Figure 9: Two example of tactical decision use cases

Figure 11 illustrates the different simulation frames of a lane changing scenario. (I) is the starting point which shows ego at right lane and three other vehicles at left lane. At (II) a lane change request is received from infrastructure and tactical decision starts the gap analysis, Figure 10. Based on the above mentioned criteria, a cost is assigned to each gap and the second gap is selected as a suitable one. At (III) the lane change manoeuvre is started and at (IV) the ego is driving on the left lane.

3.1.3.2 *AGLOSA*

The vehicle automation via *V2I* communication, may receive *AGLOSA* messages at intersections. In these messages several dynamic zones and a velocities assigned to each of these zones are defined. Figure 12 left illustrates the dynamic zones, with different green colors from dark to bright, at Braunschweig Tostmannplatz. Same figure on the right illustrates a testing scenario with a mobile traffic light which sends *AGLOSA* messages to the *FASCarE* vehicle at the DLR grounds.

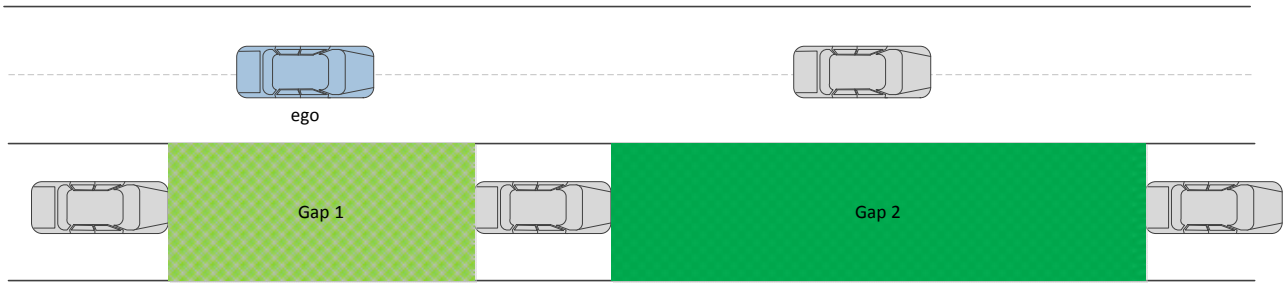

Figure 10: Gap analysis

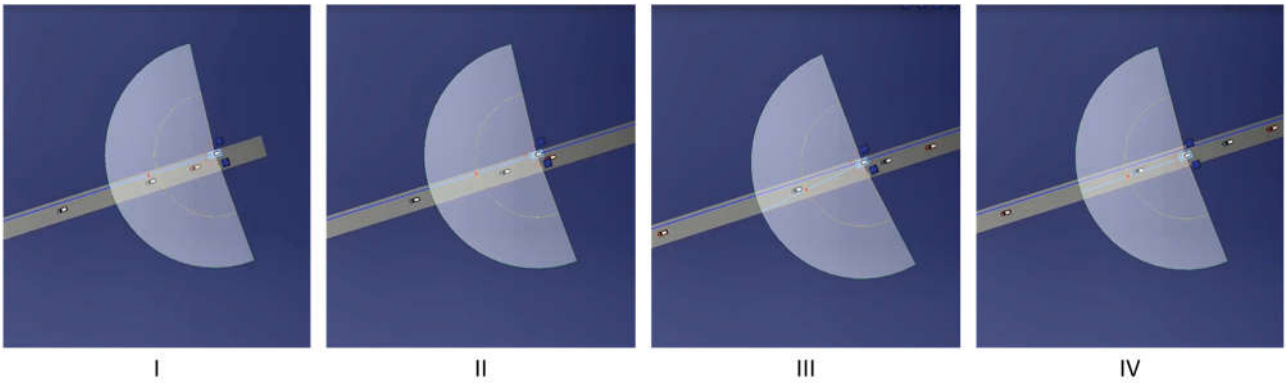

Figure 11: Simulation frames of lane changing

The tactical decision module finds out at which dynamic zone the vehicle is driving and sends the velocity of that zone as desired velocity v_{des} to *OCP* in each iteration.
Figure 13 illustrates the vehicle driven velocity for an *AGLOSA* test at DLR. In the figure, at (1) the vehicle accelerates to reach the maximum permitted velocity which is also defined as desired velocity. At (2) the vehicle receives an *AGLOSA* messages and reduces its velocity in order to drive with the velocity assigned to the current zone. At (3) the vehicle is in another zone, hence another velocity is considered as desired velocity. At (4) the vehicle arrives at the intersection that has green phase, it crosses the intersection and accelerates to maximum permitted velocity. At (5) the vehicle decelerates as it is approaching the end of the test track. Red in the figure is the velocity sent to the vehicle controller and blue is the driven velocity.

Figure 12: Left: dynamic AGLOSA zones at Tostmannplatz. Right: FASCarE following Hyundai Ioniq at the same place

Other speeds calculated by tactical decision at each iteration are:

- **Front man speed:** time headway based collision free speed calculated at each iteration to keep safe distance with front man.
- **Lateral comfort:** Driving with unsuitable velocity in curve results in high lateral acceleration or discomfort for vehicle passenger. This parameter can be considered as penalty function inside the objective function, in the form of minimizing lateral acceleration or jerk, but for

simplicity, can be also calculated based on the road curvature and maximal accepted lateral acceleration, here $2\ ^{m}/_{s^2}$.

- **Merging speed:** In order to change the lane smoothly, vehicle speed must be adapted based on the gap's following vehicle.
- **Platoon speed:** a desirable speed in order to drive, close and break a platoon must be calculated which is explained in next section.

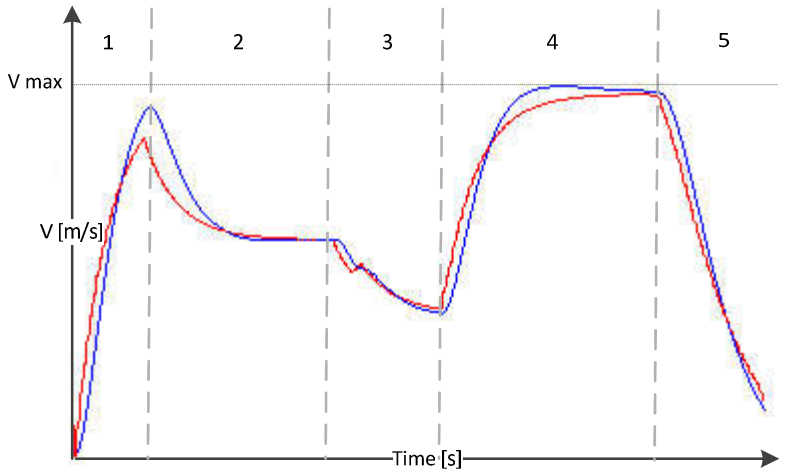

Figure 13: AGLOSA test velocity profile

3.1.3.3 *Platooning*

MAVEN automated vehicles are able to build and drive in platoons. The platoon logic is designed and developed as a unique module used in DLR and HMETC vehicles. The logic is explained in detail in D3.1 [1].
The top right of Figure 14 illustrates the four platoon state machines. At tactical decision based on the vehicle capabilities and vehicle destination, set the platoon logic inputs, see D3.1 [1]. The vehicle automation receives trajectory information of the leader and based on this information plans a manoeuvre such as lane change and receives the state of the distance state machine, such as close distance, normal distance and gap distance.

In normal distance state, the vehicle is driving normally and keeps a safe distance to the front man. In case of the close distance state a velocity profile to keep a desired time-headway with leader is generated. Close gap desired time headway is 1.5 [s] in an urban scenario, limited by the maximum braking capability of the FASCarE test vehicle of $3\ ^{m}/_{s^2}$. In close test field and simulation smaller values can be used. For gap distance the same approach is used, but in this case 3.0[s] as desired time headway is used. Figure 15 illustrates time headway-velocity profile for platooning.

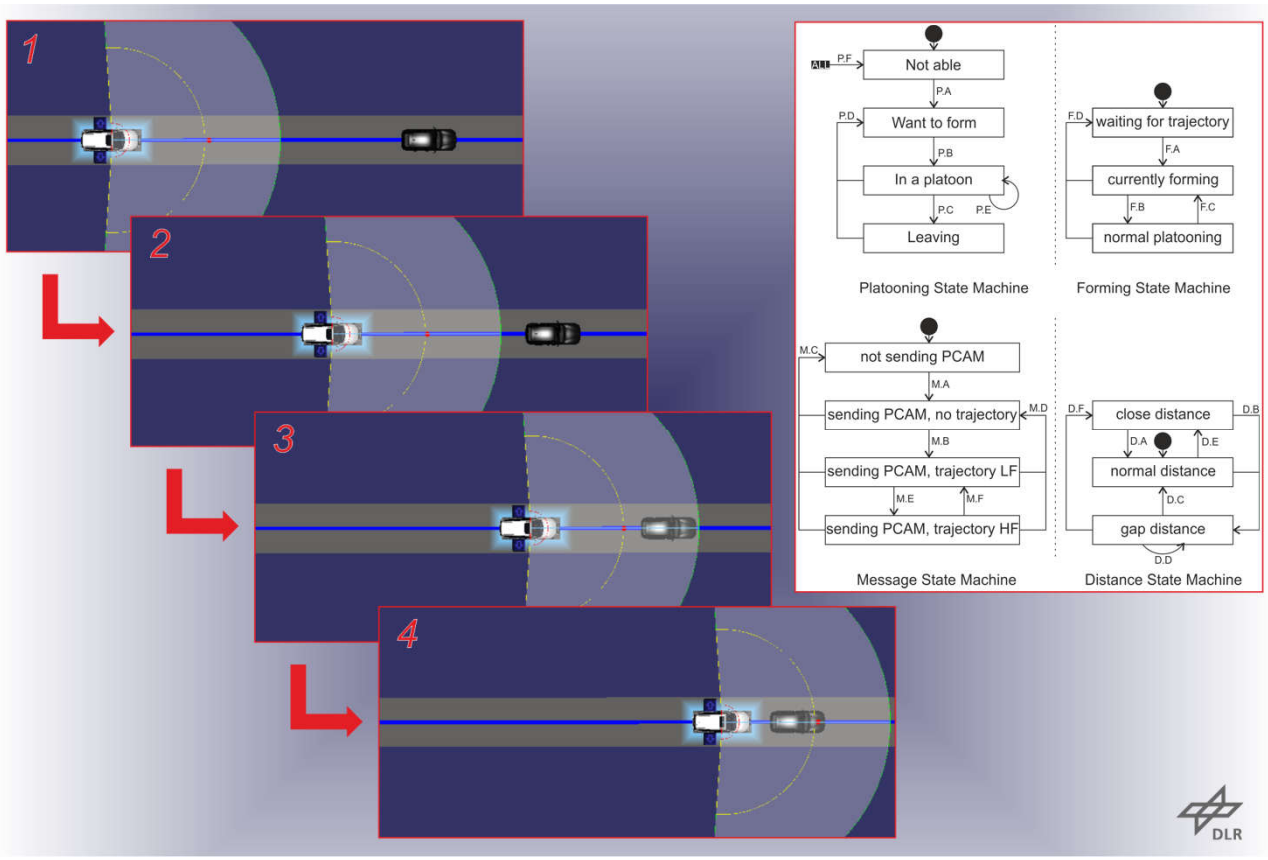

Figure 14: Simulation frames of platoon building

Figure 14 illustrates the simulation frames of building a platoon. At 1 the vehicle drives normally and is communicating its desire to form a platoon. The front man is also an automated cooperative vehicle communicating its desire to form a platoon. At 2, as all the criteria to build a platoon between two vehicles are matched, the states change to in platoon. Therefore the follower accelerates to reduce the gap. At 3 the follower is closing the gap and at 4 both vehicles are driving in platoon.

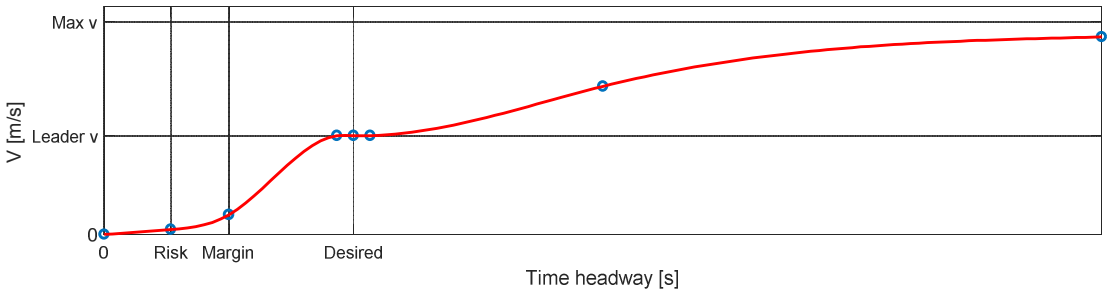

Figure 15: Platooning time headway-velocity profile

3.2 HMETC

3.2.1 *Cooperative decision making*

This section provides the general description and high level information related to the functionality of the *DMM* module used in HMETC *AD* software.
Autonomous vehicles are intended to drive in a dynamic environment and expected to react scrupulously with high level of intelligence. In an autonomous vehicle, the real-time information about the surroundings is provided by exteroceptive sensors (like RADAR, LiDAR, camera, etc.) and the information about the vehicle state is provided by the proprioceptive sensors (like *GPS*, *IMU*, etc.). The information about the environment around the autonomous vehicle is represented in terms of World Model Events. In the *AD* software, at the beginning of a journey the global route planner (*GRP*) provides the way points for the optimal route. In real-time, it is crucial to decide the most appropriate driving manoeuvre along this route depending on the sensor and *HD* map information. Road safety is considered as the prime factor while deciding a manoeuvre. In this context, the Decision Making Module (*DMM*) continuously processes the world model events and the driving directions to decide the feasible manoeuvre. Based on the environment (like highway, city, etc.) the manoeuvres are decided using a rule-based state machine. The manoeuvre decision is then utilized by the *GNC* modules of the *AD* software.

The *DMM* currently supports the following manoeuvres:

- **Lane keep** – the ego-AV drives in the lane by maintaining lane centre
- **Distance keep** – In the presence of a slower-moving obstacle at the same lane as the ego-AV, and when changing the lane is not possible, the ego- AV follows the obstacle by maintaining a safe distance
- **Stop and Go** – In the presence of a static obstacle in the same lane as the AV, and when changing the lane is not possible, the ego-AV decelerates and stops at a safe distance. Similarly, the ego-AV decelerates and stops when *V2X* messages from a cooperative traffic light indicate that the current phase is red and the ego-vehicle is reaching the stop line.
- **Left Lane Change** - In the presence of a slower-moving or static obstacle in the same lane as the ego-AV, or when a *V2X* LAM advice is received from the cooperative infrastructure, the ego-AV executes a lane change manoeuvre to the left lane
- **Right Lane Change -** In the presence of a slower-moving or static obstacle in the same lane as the ego-AV, or when a *V2X LAM* advice is received from the cooperative infrastructure, the ego-AV executes a lane change manoeuvre to the left lane
- **Destination** – When the ego-AV approaches the destination it decelerates and stops

The operations performed by the *DMM* to alternate among the different possible states are described in the following by making reference to the following figure:

- **Data Extraction -** The Decision Making Module requires various information about the environment. Relevant aspects about the lanes, road, obstacle and route need to be extracted from the large information available. Additionally, the information from *V2X* messages received from other cars of the Infrastructure or car can also be extracted. For example, information related to lane marking type, driving direction of the road etc. will be extracted from Road Network (*HD* map) information.

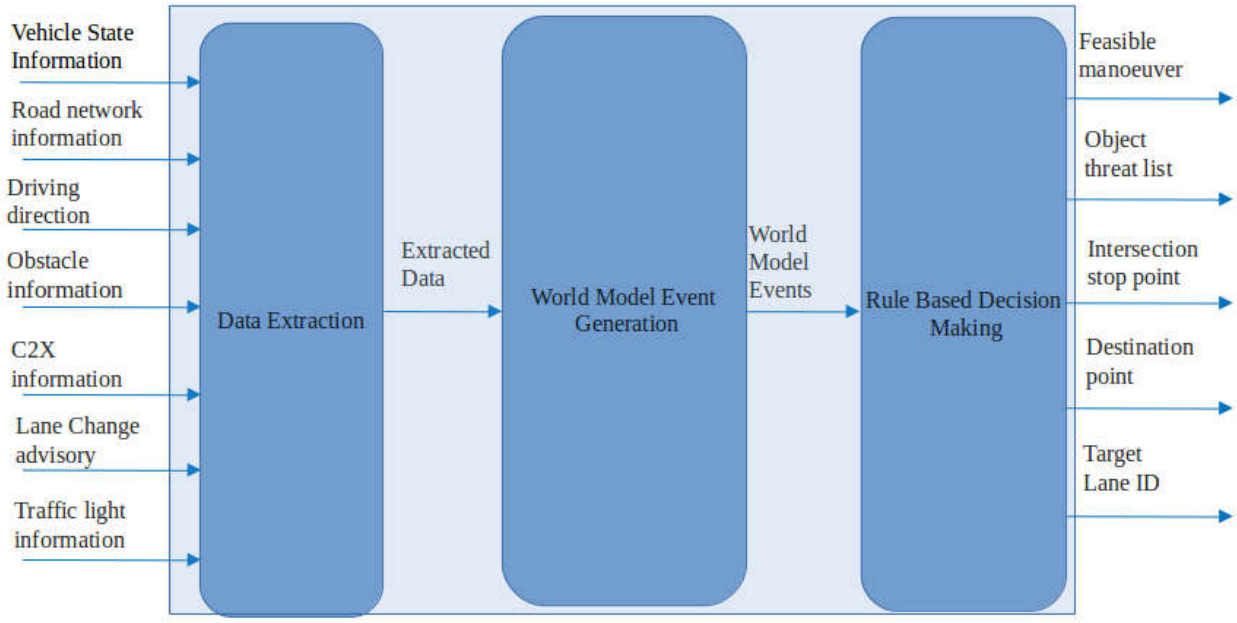

Figure 16: schematic representation of the operations performed by the DMM

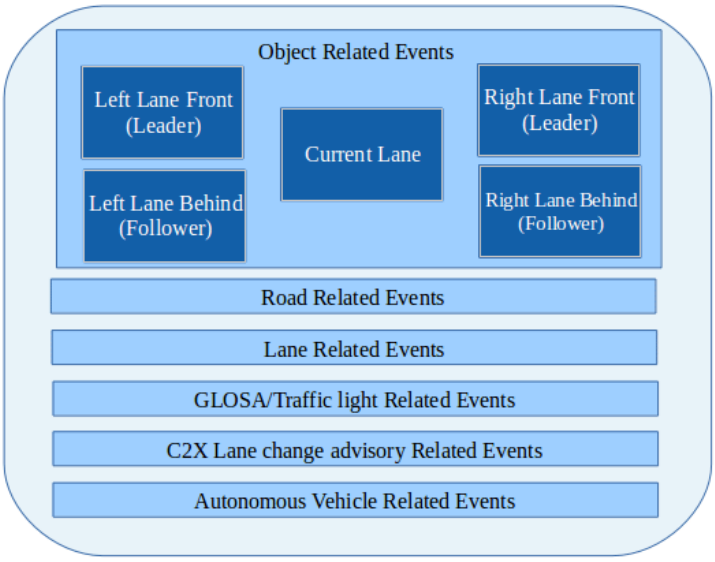

Figure 17: DMM WM events classification

- **World Model Event Generation** - The environmental data extracted from the Data extraction component, is processed and represented as World Model Events (*WME*) by the World Model Event generation component. For example, based on the lane marking type (DASHED, SOLID etc.), the “Lane-related” World Model Event "Lane Boundary Crossable" will be SET or RESET. Multiple WM events are continuously generated based on the classification given in the Figure 17.

- **Rule Based Decision Making** - The transition from one state to the other will be controlled by the transition conditions. Transition conditions are rules implemented by

using different combinations of the world model events. The decision making process is continuously performed and instantaneous in nature. Depending on the occurrence of world model events, transitions will happen from one state to another state. One of the states will be the finally decided manoeuvre as output of the state machine. A pictorial representation of the rule-based decision making is as shown below:

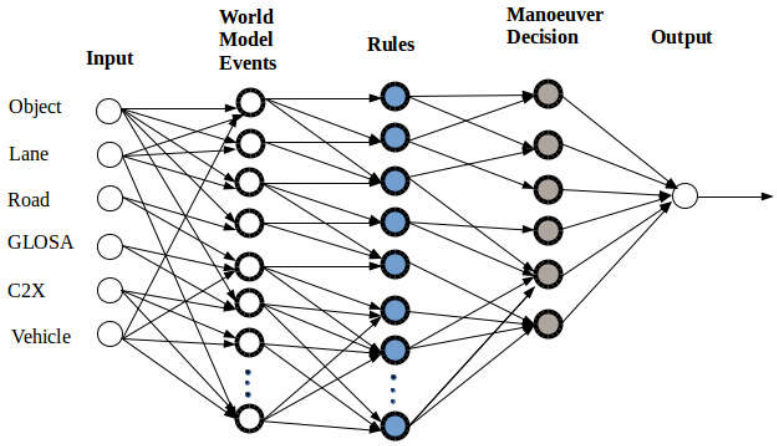

Figure 18: DMM Rule-based decision making

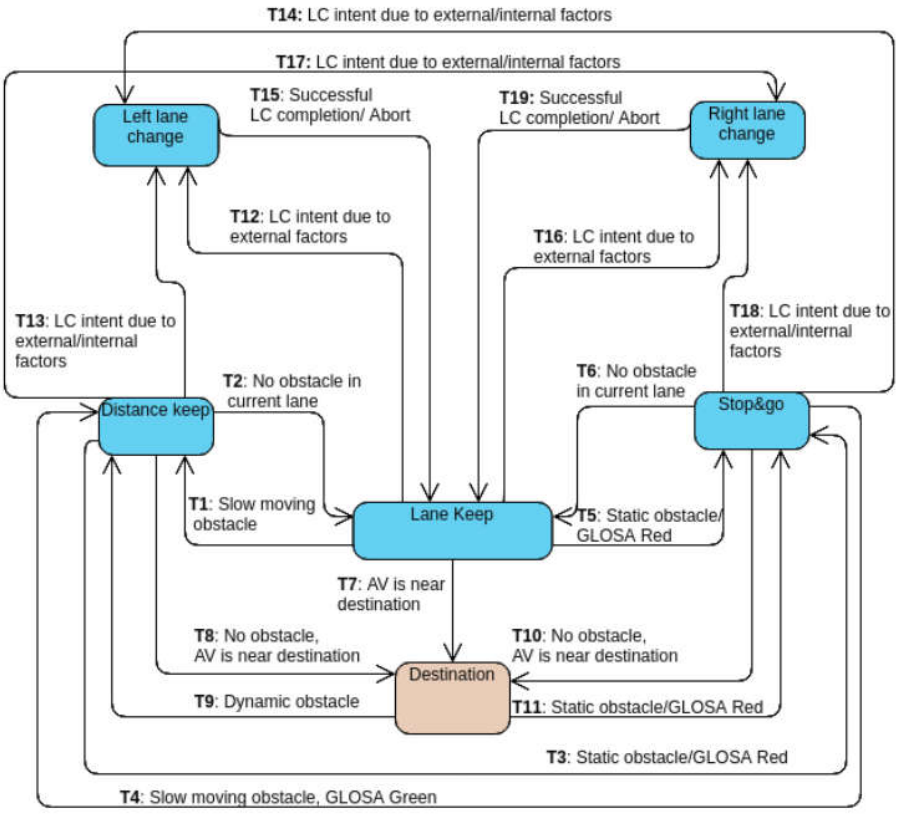

Figure 19: DMM state diagram

The state diagram regulating the transitions among the *DMM* states is represented in the Figure 19.

The transitions conditions adopted by the state machine are as follow:

T1: LANE KEEP TO DISTANCE KEEP:
1. In case there is an obstacle moving at lower speed than the ego vehicle in the same lane. In this case, a lane change is not feasible and the obstacle kept within a distance keep threshold distance that depends on the ego-vehicle speed.

T2: DISTANCE KEEP TO LANE KEEP:
1. When the obstacle in the ego vehicle lane is moving at a higher speed than the ego vehicle.
2. When the obstacle in the ego lane is moving at a lower speed, but it is beyond the distance keep threshold distance.
3. When there is no obstacle in the current lane.

T3: DISTANCE KEEP TO STOP AND GO:
1. When the slower obstacle in front stops in the ego lane before a traffic light–signalized stop line.
2. When slower obstacle in front in the ego lane crosses a traffic light–signalized stop line and the current traffic light signal (received via *V2X*) is RED.
3. When the ego-vehicle is approaching a traffic light–signalized stop line and the current traffic light signal (received via *V2X*) is RED.

T4: STOP AND GO TO DISTANCE KEEP:
1. In case there is a slower-moving obstacle in the ego lane and traffic light signal status (received via *V2X*) turns GREEN.

T5: LANE KEEP TO STOP AND GO:
1. When the obstacle in front in the ego lane is static and lane change is not feasible.
2. When the ego vehicle is approaching an intersection and the traffic light signal status (received via *V2X*) is RED.

T6: STOP AND GO TO LANE KEEP:
1. When there is no obstacle in front in the ego lane.
2. When the front obstacle in the ego lane moves at speed higher than the ego vehicle speed.
3. When the traffic light signal status (received via *V2X*) turns GREEN.

T7: LANE KEEP TO DESTINATION:
1. When the ego vehicle approaches the destination and the distance between ego vehicle and the destination point is less than a threshold distance.

T8: DISTANCE KEEP TO DESTINATION:
1. When the ego vehicle approaches the destination and the distance between ego vehicle and the destination point is less than a threshold distance .

T9: DESTINATION TO DISTANCE KEEP:
1. When the destination point is after a slower-moving obstacle in the same lane as the ego-vehicle.

T10: STOP AND GO TO DESTINATION:

1. When the ego vehicle approaches the destination and the distance between ego vehicle and the destination point is less than a threshold distance.

T11: DESTINATION TO STOP AND GO:
1. When the destination point is after the static obstacle.

T12: LANE KEEP TO LEFT LANE CHANGE:
1. When there is a *V2X* request for *LC* from the infrastructure and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.
2. When there is a slow moving or static obstacle in the ego lane and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.

T13: DISTANCE KEEP TO LEFT LANE CHANGE:
1. When there is a *V2X* request for *LC* from the infrastructure and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.
2. When there is a slow moving or static obstacle in the ego lane and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.

T14: STOP AND GO TO LEFT LANE CHANGE:
1. When there is a *V2X* request for *LC* from the infrastructure and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.
2. When there is a static obstacle in the ego lane and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.

T15: LEFT LANE CHANGE TO LANE KEEP:
1. When the lane change is completed.
2. When the lane change is aborted due to environmental conditions (Ex: drastic reduction of LV velocity, Figure 20.

T16: LANE KEEP TO RIGHT LANE CHANGE:
1. When there is a *V2X* request for *LC* from the infrastructure and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV , Figure 20.
2. When there is a slow moving or static obstacle in the ego lane and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV , Figure 20.

T17: DISTANCE KEEP TO RIGHT LANE CHANGE:
1. When there is a *V2X* request for *LC* from the infrastructure and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.
2. When there is a slow moving or static obstacle in the ego lane and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.

T18: STOP AND GO TO RIGHT LANE CHANGE:
1. When there is a *V2X* request for LC from the infrastructure and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.
2. When there is a static obstacle in the ego lane and a suitable gap distance is available between the ego-vehicle and the OLV, LV & FV, Figure 20.

T19: RIGHT LANE CHANGE TO LANE KEEP:
1. When the lane change is completed.
2. When the lane change is aborted due to environmental conditions (Ex: drastic reduction of LV velocity, Figure 20.

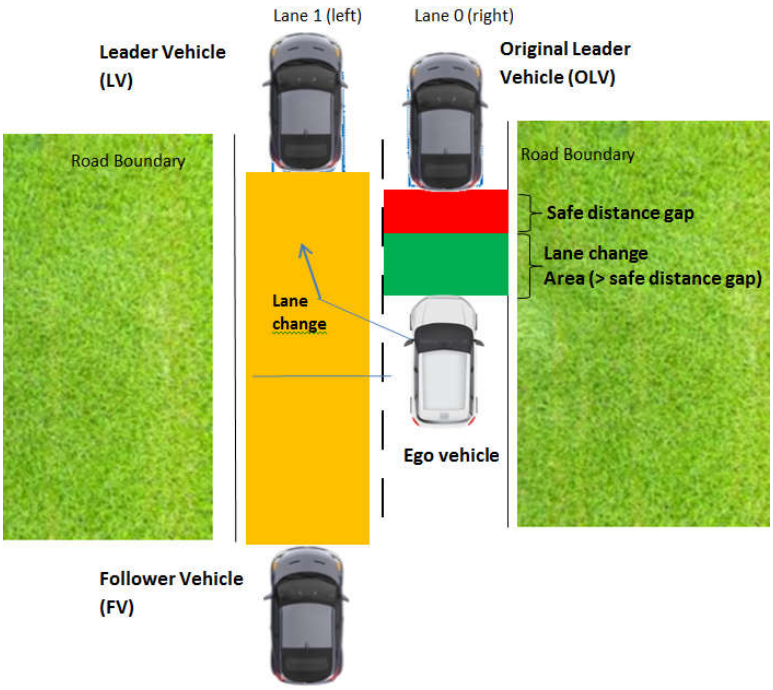

Figure 20: naming scheme for obstacle vehicles considered by DMM for lane change decisions

3.2.2 *Cooperative aspect*

In section 2.2 it has been highlighted that a dedicated *V2X* interface node is in charge, on the reception side, to extract data of received *V2X* messages from the *V2X* communication module. The interface node then distributes this data to the different *AD_SW* modules that reuse it. For the communication between the *V2X* communication unit and the *V2X* interface node, *UDP* sockets are used. When the *V2X* modules receives *V2X* messages from the infrastructure (*GLOSA* and *LAM* messages) *UDP* sockets are used to provide data structures to the *GNC DMM* module that reuse it (IF_1 in Figure 1).
Important to mention that the HMETC *AD* framework includes an emulation approach used to emulate *V2X* transmissions and receptions in absence of other cooperative automated vehicles or infrastructure (see Figure 21). The *AD_SW* emulation module can be used to record data extracted from *V2X* messages received from real infrastructure (e.g. speed or lane change advices recorded at a given traffic light). These recordings are converted into *ROS* bag files and stored in the format as they would be received over the above mentioned *UDP* interfaces. The *ROS* bag files can be then “replayed” within the *AD_SW* logic of the ego-vehicle when performing simulations or road tests on the test track. This replaying of the bag file is emulating the receptions of *V2X* messages from the transmitting infrastructure (virtual cooperative stations in this case).

3.2.2.1 *Cooperative planning in reaction to GLOSA and LAM advices*

This section describes the functionalities performed by the *GNC* to process the *V2X* data relative to *GLOSA* and *LAM* advices. A complete and detailed description of the data structures passed as inputs to the GNC for this purpose is given in the deliverable D5.1 [6].

When *GLOSA* data is received by the *GNC*, the following operations are performed:

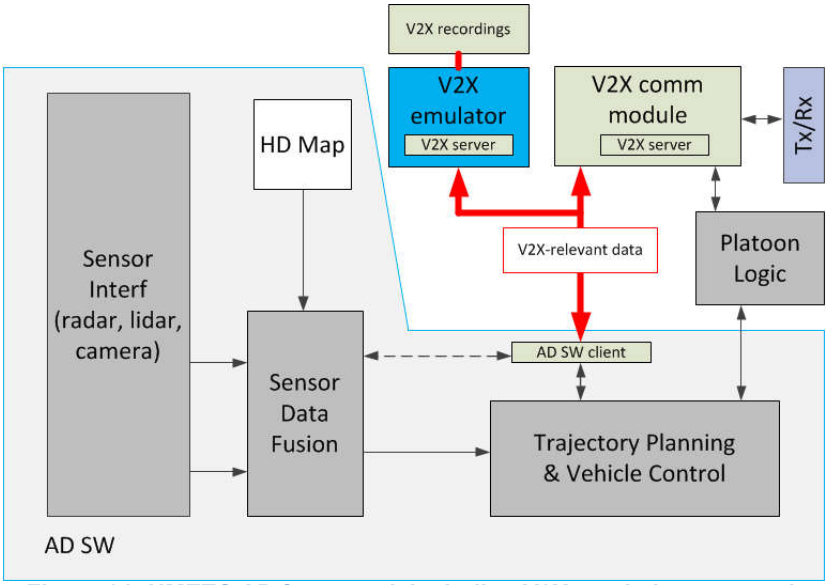

Figure 21: HMETC AD framework including V2X emulation approach

- The received *GLOSA* information provides speed advices applicable to particular combinations of ingressing and egressing lanes at the intersection. As the ego vehicle already knows the route it would be traversing on, the ingressing and egressing lanes relevant for the ego vehicle's route are extracted from the *HD* map and considered for *GLOSA* adaptation. The same applies for the position of the stop line over the relevant ingressing lane, which identifies the position where the car needs to stop in case of red.
- The distance of ego vehicle from its current position to that corresponding stop point in the relevant ingressing lane is continuously calculated.
- The traffic light/infra provides speed advices for distinct distance-zones along each ingressing lane. As a consequence, the *HD* map lane ID associated to the *V2X* ingressing lane ID where the vehicle is currently driving is extracted. Then, the speed advices and signal phase corresponding to that particular lane are continuously read.
- Using the currently calculated distance to stop line, the ego vehicle localizes into the distance-zones associated to the speed advices. The speed advisory corresponding to the zone where the vehicle is currently is sent out to the motion planner for speed adaptation. The speed adaptation request can be accepted only in case it is not conflicting to the objective of maintaining a safe distance to any obstacle in the ego-lane. If a conflict is detected, the *GLOSA* speed adaptation is rejected.

The above mentioned approach has been verified in simulation by running either a synthetic set of *GLOSA* speed advices (compliant in format to those transmitted in the Helmond and Braunschweig test sites), as well as recordings of real-road *V2X GLOSA* collected directly at Helmond and Braunschweig. The following figure shows the simulation outputs of the *GLOSA* adaptation, which verifies the functionality and prepares its implementation in the *AD* vehicle prototype.

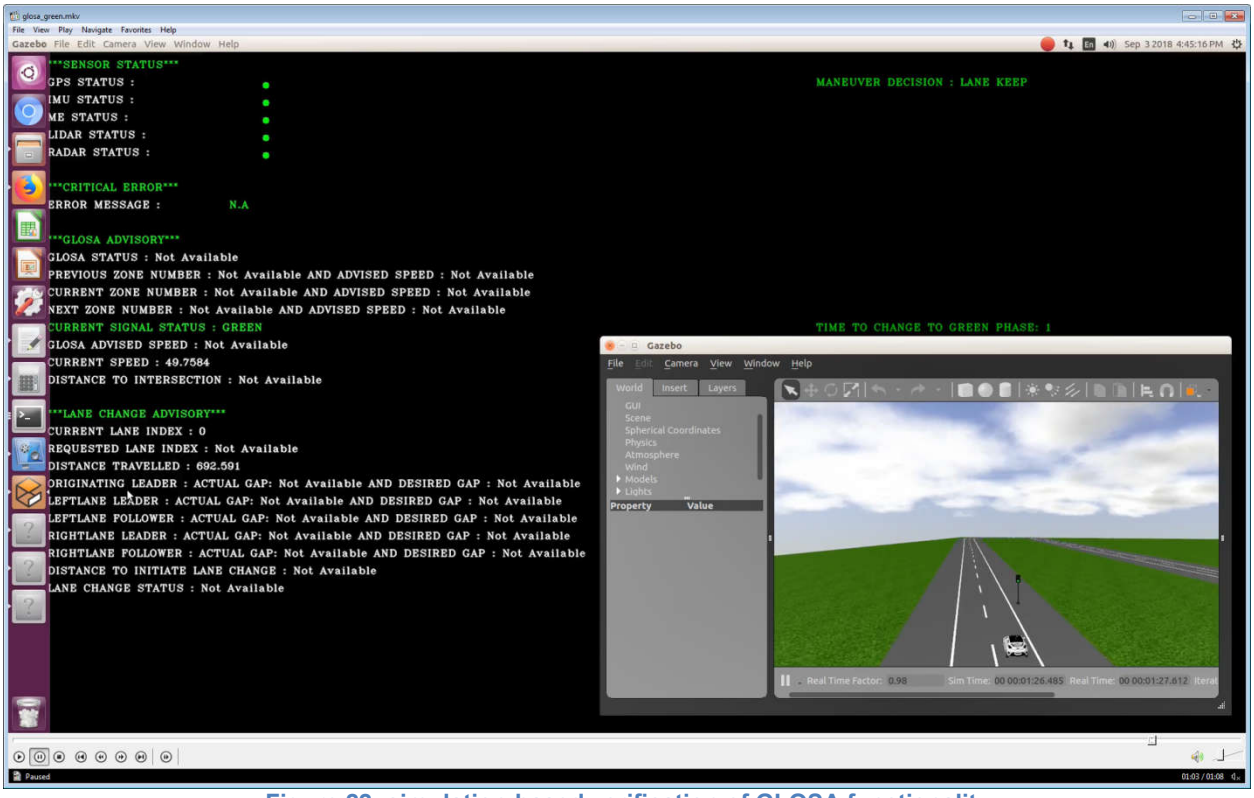

Figure 22: simulation-based verification of GLOSA functionality

On this basis, HMETC performed tests on the Griesheim test track using the HMETC automated vehicle stimulating the *AD_SW* planning and control modules to adapt the ego speed to the one dynamically suggested by the *AGLOSA* traffic light controller running at the Tostmannplatz signalized intersection. For this purpose, a preliminary task was the collection/recording of *GLOSA* information transmitted by the Tostmannplatz *RSU*. Figure 23 shows a picture of the Hyundai car during these recording sessions.

Figure 23: Recording of GLOSA messages on the Tostmannplatz

These recordings represent the real-world dynamic evolution of the traffic light phases and speed advices over a given time window and can be replayed by the *AD_SW* as an additional emulated input for the vehicle automation (see figure above).
The speed adaptation of the automation system has been successfully verified on the Griesheim test track for different combinations of vehicle distances from the stop line and *GLOSA* recordings. Figure 24 shows one example of driving automated on the Griesheim track while replaying the

above mentioned *GLOSA* recordings collected at the Tostmannplatz. As we can see, the automated car's speed adapts dynamically to the *GLOSA* speed while approaching the stop line (placed at approximately 850m driven distance). Please notice that the *GLOSA* speed equal to zero means that no speed advice is present in the recordings at that moment. By following the current value of the *GLOSA*, the vehicle drives at a lower speed than the maximum allowed (50 kph) which allows crossing the stop line without stopping after the green phase has started.

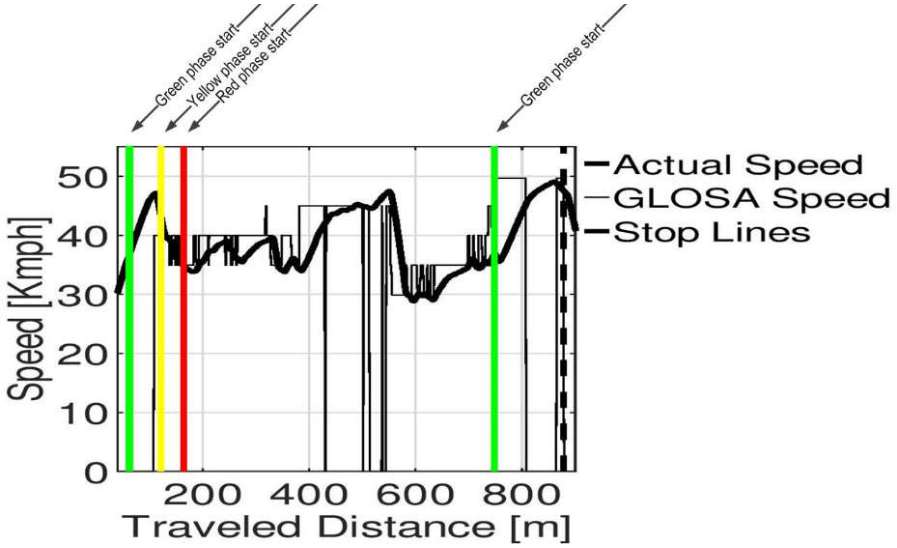

Figure 24: GLOSA adaptation verification on the AD vehicle reusing real-world GLOSA recording

When *LAM* data is received by the *GNC*, the following operations are performed:

- The received *LAM* information provides lane change advices applicable to particular target ingressing lanes at the intersection. As the ego vehicle already knows the route it would be traversing on, the ego and target ingressing lanes relevant for the ego vehicle's route are extracted from the *HD* map and considered for lane change. The same applies for the position of the stop line over the relevant ingressing lane, which identifies the position where the car needs to stop in case of red (the *GLOSA* information about the current traffic light phase is continuously considered in the background for possible speed adaptation).
- The distance of ego vehicle from its current position to that corresponding stop point in the relevant ingressing lane is continuously calculated.
- The traffic light/infra provides lane change advices in form of target lanes to change to and the distance to the stop line where the *LC* should be initiated. As a consequence, the *HD* map lane ID associated to the *V2X* ingressing lane ID where the vehicle is currently driving is extracted. Then with the extracted ingressing lane ID, the lane change advice is read to identify the ID of the target lane.
- Using the currently calculated distance to stop line, ego vehicle counts down from its current position to the target distance suggested for initiating the *LAM* advice. At the point of the target distance, a lane change request to the *DMM* is triggered. The request is accepted only if the transition conditions highlighted in Section 3.2.1 are met. In particular the execution of a *V2X*-triggered lane change request is subject to the availability of safe gaps with the surrounding vehicles. If such gaps are not available at the triggering moment, the *DMM* will try to execute the lane change at later instants till the vehicle will reach a threshold distance from the stop line. The *DMM* will continuously check the availability of safe gaps during this period and execute the *LC*

as soon as they are granted. If the threshold distance to the stop line is reached without getting safe gaps, the *LC* request is definitely aborted.

Similarly to *GLOSA*, the above mentioned *LC* functionality has been verified in simulation as well as in emulation using the HMETC *AD* vehicle prototype. In this case a synthetic *LAM* advice (compliant in format to those transmitted in the Helmond and Braunschweig test sites) is used. The following figure shows the simulation outputs of the *LC* adaptation (in combination with *GLOSA*), which verifies the functionality and prepares its implementation in the *AD* vehicle prototype.

LAM emulation tests using the HMETC automated vehicle for *LC* execution on the Griesheim test track have also been successfully performed. In this case, the objective was stimulating the *AD*_SW *DMM*, planning and control modules to adapt to the *LC* advices in different conditions of surrounding traffic, with real cars opening and closing the safe gaps considered for *LC* execution decisions. The following figures show some pictures of the Hyundai car during these testing sessions. Videos of the performed tests have been also shown during consortium meetings.

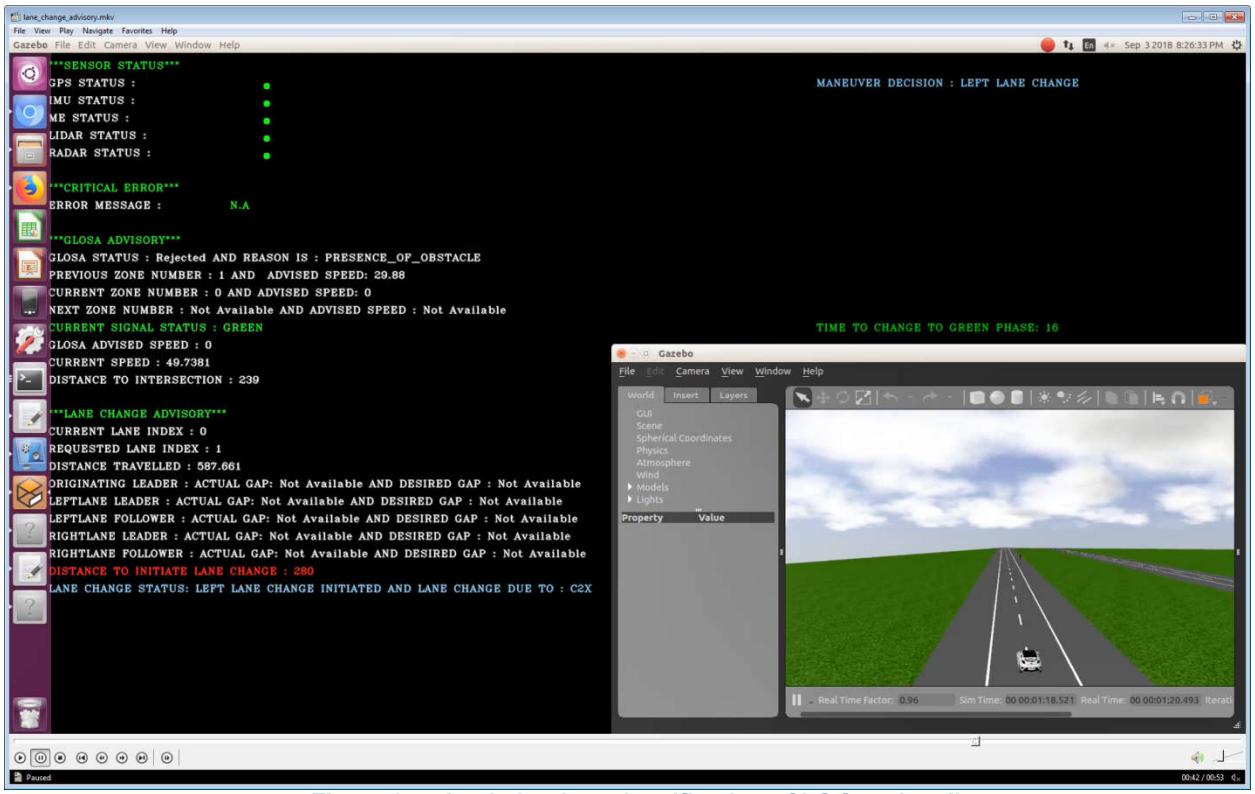

Figure 25: simulation-based verification of LC functionality

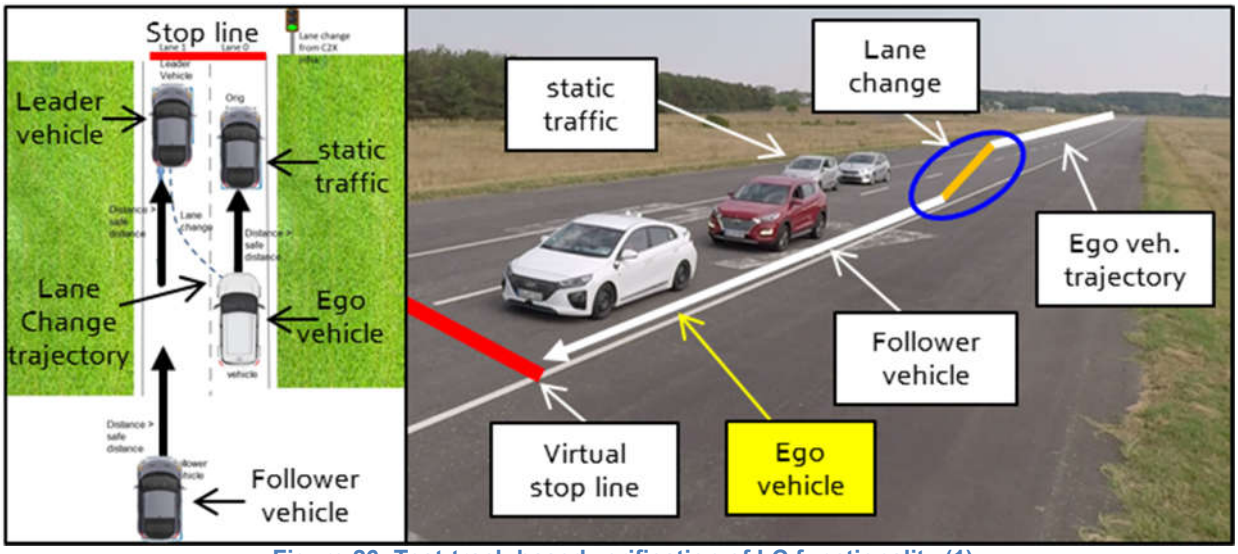

Figure 26: Test-track-based verification of LC functionality (1)

Figure 27: test-track-based verification of LC functionality (2)

4 Conclusion

In this deliverable the implementation and the results of cooperative manoeuvre and trajectory planning of DLR and HMETC has been explained.

As mentioned in this document, DLR and HMETC approach differs. DLR trajectory planner is optimal control based and the planned trajectory is a function of objective function. Mathematic formulation of the optimal control problem has been explained. It has been explained that due the problem complexity, the optimization horizon is limited and therefore tactical decision is required to reformulate the optimal control problem. The role of tactical decision in cooperation with infrastructure, *LAM* and *AGLOSA* and other cooperative automated vehicle in platooning use cases has been explained. HMETC trajectory planner is rule based. Decision Making Module receives information about vehicle surrounding, infrastructure and other cooperative vehicles and generates a set of feasible manoeuvres and object threat list. Path and motion planner uses feasible manoeuvres and object threat list as inputs and select the most suitable manoeuvre. Selected manoeuvre satisfy safety aspects as well as cooperation.

Despite the differences on the approach, DLR and HMETC automation architecture is similar in order to enable compatibility to perform MAVEN use cases.

To validate the functionality of approaches and algorithms the results of simulation test, close field tests and urban tests of vehicle automation and its cooperation has been illustrated and presented.

5 Bibliography

[1] MAVEN Deliverable D3.1. "Detailed Concepts For Cooperative Manoeuvre And Trajectory Planning And For In Vehicle Cooperative Environment Perception".

[2] S.Schmidt, “Ein Optimales Steurungs- und Reglungskonzept für autonome Elektrofahrzeuge,” Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2013.

[3] R. Dariani, “Hierarchical Concept of Optimization Based Path Planning for Autonomous Driving,” Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2016.

[4] MAVEN Deliverable D3.2. "Cooperative environment perception algorithms".

[5] LaValle and Stevem, “Rapidly-exploring Random Trees: A new tool for path planning,” Technical report, Computer science department, Lowa state University, October 1998.

[6] MAVEN Deliverable D5.1. "V2X communications for infrastructure-assisted automated driving".