D3.5

Concepts and algorithms incl. V&V results from 2nd cycle

Project Number:	690705
Classification	Public
Deliverable No.:	D3.5
Work Package(s):	WP3
Document Version:	Vs. 0.13
Issue Date:	31.12.2017
Document Timescale:	Project Start Date: September 1, 2016
Start of the Document:	Month 15
Final version due:	Month 16
Deliverable Overview:	**Main document:** D3.7
Compiled by:	David Käthner
Authors:	Andrea Castellano (REL) Mark Eilers (HMT) Maximilian Graf (ULM) Elisa Landini (REL) Paulin Pekezou Fouopi (DLR) David Käthner (DLR) Mohamed-Cherif Rahal (VED) Stefan Suck (OFF)
Technical Approval:	Fabio Tango, CRF
Issue Authorisation:	Andreas Lüdtke, OFF

DISTRIBUTION LIST		
Copy type[1]	Company and Location	Recipient
T	AutoMate Consortium	all AutoMate Partners

RECORD OF REVISION		
Date	Status Description	Author
08.11.2017	Deliverable structure	David Käthner (DLR)
15.11.2017	Added separate sections for ULM and VED demonstrators	David Käthner (DLR)
01.12.2017	Changed structure to fit Elisa's' suggestions	David Käthner (DLR)
01.12.2017	Added OFF/HTM input	Mark Eilers (OFF)
01.12.2017	Added Elisa's input	Elisa Landini (REL)
01.12.2017	formatting & spelling	David Käthner (DLR)
04.12.2017	Added Max' input	Max Graf (ULM)

[1] Copy types: E=Email, C=Controlled copy (paper), D=electronic copy on Disk or other medium, T=Team site (Sharepoint)

05.12.2017	Added Stefan's input	Stefan Suck (OFF)
06.12.2017	Added Max' validation input	Max Graf (ULM)
06.12.2017	Added Paulin's input	Paulin Pekezou Fouopi (DLR)
06.12.2017	Added Mohamed's input	Mohamed-Cherif Rahal (VED)
07.12.2017	Added VED's input on validation	Mohamed-Cherif Rahal (VED)
08.12.2017	Changed wording in introduction, added conclusion	David Käthner (DLR)
11.12.2017	Added validation results for online risk assessment	Mark Eilers (HMT)
18.12.2017	Integrated various added input + feedback from reviewers	David Käthner (DLR)
22.12.2017	Integrated various added input + feedback from reviewers and compiled the final version	Paulin Pekezou Fouopi (DLR)

Table of Contents

1 Introduction

The activities in the Automate project have been organized in 3 cycles to guarantee that the maturity of the technologies developed in the project is iteratively increased while assessing that the progresses are consistent with the needs of the demonstrators and, in turn, with the overall concept and objectives of the project.

As shown in Figure 1, the first 2 cycles are focused on the development and technical validation of the components (i.e. the enablers) performed in WP2, WP3 and WP4. The experience acquired in the 1st cycle (lesson learnt) has been used at the beginning of the 2nd cycle to review the requirements and metrics for the design and development of the enablers and, as a consequence, to improve them.

At the end of the 2nd cycle, the enablers are planned to be integrated into the demonstrators in WP5, and the performances of the 1st version of the demonstrators are evaluated against their baseline in WP6.

In the 3rd cycle, WP2, WP3 and WP4 are fed with the results of this evaluation process to deliver the final version of the enablers. The 3rd cycle ends with the evaluation of the final version of the demonstrators.

This deliverable describes the current state of the enablers developed in WP3 in the first half of the 2nd cycle, as well as the experiments conducted and proposed to technically validate them according to the validation plan and the requirements and metrics defined in D3.4.

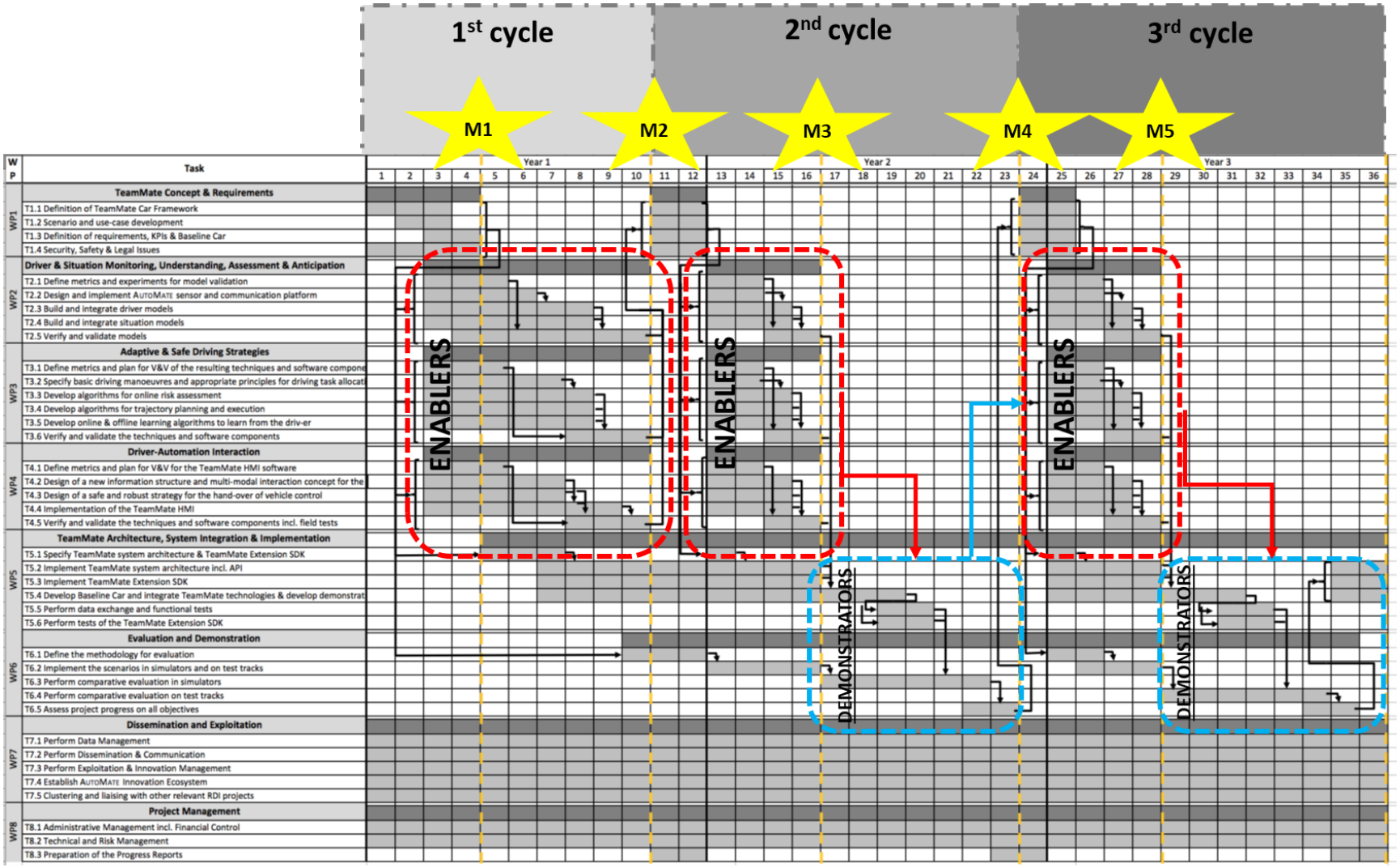


Figure 1: Project cycles, milestones and link between enablers (WP2, WP3 and WP4) and demonstrators (WP5 and WP6)

The development of all enablers follows the same process for WP2, WP3 and WP4. Therefore, the deliverable D2.4, D3.5 and D4.4 that describe the status of the development and validation of the enablers have been structured with the same chapters to reflect the common (parallel) process followed in WP2, WP3 and WP4 to deliver all enablers in time to be integrated into the demonstrators.

This deliverable first describes how the WP3 enablers contribute to the implementation of the TeamMate car concept. Next, in order to detail the background for the technical developments, the use cases and user stories have been described. This is followed by a report of the current status of the WP3 enablers. Finally, the validation of the WP3 enablers has been detailed.

2 How the WP3 enablers contribute to the implementation of the concept of the project

The top-level objective of AutoMate is to develop, evaluate and demonstrate the “TeamMate Car” concept as a major enabler of highly automated vehicles. This concept consists of considering the driver and the automation as members of one team that understand and support each other in pursuing cooperatively the goal of driving safely, efficiently and comfortably from A to B.

As a consequence, in order to show how the enablers contribute to the implementation of this concept, it is important to briefly explain why the cooperation is needed, and how the human and the automation can support each other to create a safe, efficient and comfortable driving experience.

As shown in Figure 2, both the human and the automation have **limits** that can negatively affect the safety as well as the efficiency, the comfort, the trust and the acceptance of the autonomous driving.

For the human, the limits are often related to their driving performance: they are likely to affect the safety, and cause accidents. For the automation, the limits, mostly at perception and decision level, may affect the efficiency and the comfort of the trip, and then, in turn, the acceptance of the automation.

The AutoMate approach is based on the mutual complementarity between the driver and the automation: this support is achieved through the cooperation, between the team members.

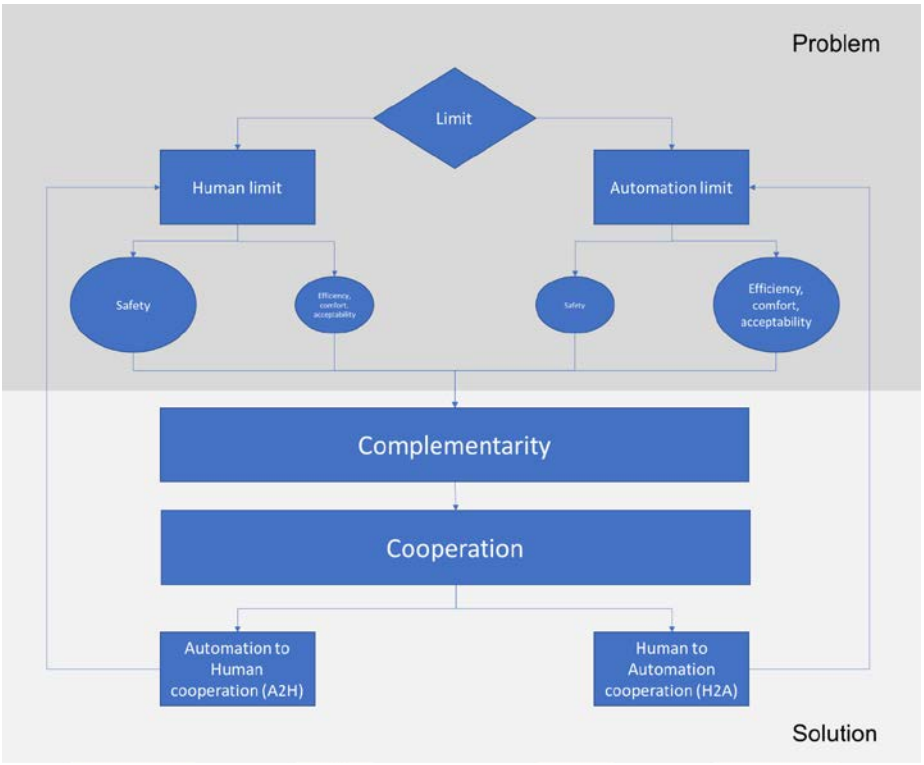


Figure 2: Schematic representation of the overall concept of the project.

While the Automation to Human Cooperation (A2H) is used to complement the human limits, the Human to Automation Cooperation (H2A) is implemented to allow the driver to support the automation to overcome its limits.

The complementarity between the driver and the automation is the conceptual solution to compensate the reciprocal limitations. While the cooperation is how the complementarity is implemented. Figure 3 shows how both the A2H and the H2A cooperation can be implemented in perception (state A and B) and in action (state C and D).

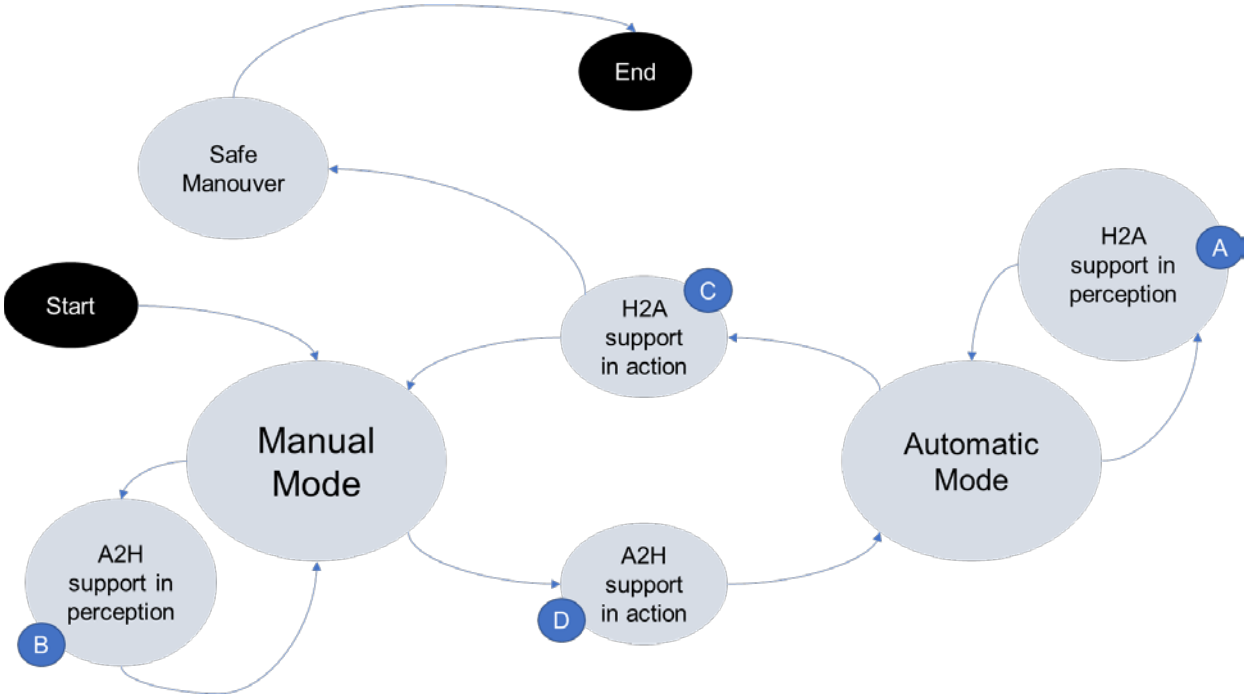


Figure 3: State machine that shows how the cooperation is implemented

The scenarios and use cases selected to demonstrate the relevance of each enabler are therefore representative and consistent with the direction of cooperation implemented by that enabler, as well as the modality of support (i.e. either in action or perception). Since the cooperation is implemented through the enablers developed in the project, Table 1 shows the role and relevance of each enabler in the cooperation.

WP	ID	Enabler	Enabler Owner	Aim of the enabler	Direction of the support	
					Automation to Human	**Human to Automation**
WP3						
	E4.1	Planning and execution of safe manoeuvre	ULM VED	It plans the possible manoeuvres and selects the most effective, efficient and comfortable.	Enabler E4.1 is needed to implement a **support in action** to complement the ability of the driver to intervene in case of risk	
	E4.2	Learning of intention from the driver	OFF HMT	It learns the driver's intention to predict the expected behaviour	Enabler E4.2 is needed to implement a **support in perception**	

					to complement the ability of the driver to assess the risk in case of risky behaviour	
	E5.1	Online risk assessment	OFF DLR HMT	It defines a safety zone where the vehicle is not likely to collide either with obstacles or other vehicles (according to the prediction of their future position).	Enabler E5.1 is needed to implement a **support in perception** to complement the ability of the driver to assess	

					the risk	

Table 1: Role and relevance of the WP3 enablers for the cooperation

3 Status of WP3 enablers in cycle 2 (enabler owners)

3.1 E4.1 - Planning and execution of safe manoeuvre (ULM)

The purpose of the planning and execution of safe manoeuvre in AutoMate is to plan all possible safe manoeuvres and select the most effective, efficient and comfortable.

3.1.1 Scenario and uses case where E4.1 is relevant

As shown in Table 1, Enabler E4.1 is needed to implement a support in action from the automation to the human (A2H) to complement the ability of the driver to intervene in case of risk.

One of the use cases of PETER scenario has been revised to highlight and clarify the role of E4.1 to implement this cooperation.

Peter is driving in a narrow rural road in Manual Mode. He approaches a tractor, that causes limited visibility or the road, but he is in a hurry, so he decides to perform the overtake. The TeamMate car detects a car approaching from the opposite lane. A collision is likely to occur. In order to avoid it, the TeamMate car takes the control of the vehicle and safely plans and executes a safe manoeuvre to drive the vehicle back to the original lane. When the situation is safe, the automation hands over the control to the driver (back to Manual Mode).

3.1.2 Concept

The trajectory is approximated by 2D points $x_k = x(t_k),\ k = 1 \dots n$ and linearly interpolated between these points. Velocities, accelerations etc. are determined by differential quotients. The variables x_k serve as variables of an optimisation problem that minimizes the following cost function:

$$L = \int_0^T j_{offs} + j_{vel} + j_{acc} + j_{jerk} + j_{yawr}\, dt.$$

j_{offs} is a cost term which contributes to drive the vehicle in the middle between the lane's boundary lines. The functions d_{left}, d_{right} are signed distance functions.

$$j_{offs}(x(t)) = w_{offs} \left| \frac{1}{2} \left(d_{left}(x(t)) + d_{right}(x(t)) \right) \right|^2.$$

The driving corridor consists of two boundary lines (modelled as polygonal lines) and the functions d_i are positive for all points on the left of the corresponding line i and negative for all points on the right.

The term

$$j_{vel}(x(t)) = w_{vel} \left| \boldsymbol{v}_{des}(x(t)) - \dot{\boldsymbol{x}}(t) \right|^2$$

is used to reach the current top speed. The vector v_{des} corresponds to the actual top speed. For more details refer to [1].

The acceleration term contributes to minimise the acceleration

$$j_{acc}(x(t)) = w_{acc} \, |\ddot{x}(t)|^2$$

while the jerk term contributes to minimise the jerk

$$j_{jerk}(x(t)) = w_{jerk} | \dddot{\boldsymbol{x}}(t)|^2$$

.Finally,$j_{yawr}$is the term for the yaw rate

$$j_{yawr} = w_{yawr} \ \dot{\psi}(t)^2.$$

The variables w_i are weighting factors.

To avoid collisions with other vehicles, these vehicles' positions are predicted for time intervals. Subsequently, obstacle polygons are defined to approximate their shape for time intervals (see Figure 4).

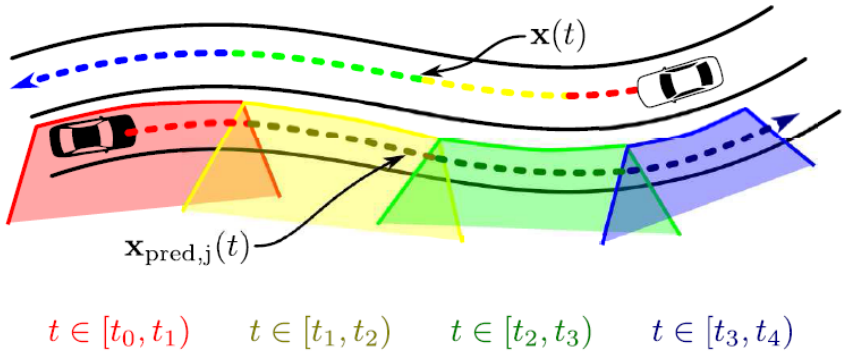


Figure 4: Polygons for collision avoidance with other vehicles (Ziegler J. et al. 2014).

In the next step, the shape of the ego vehicle has to be approximated. For this, circles are used, as shown in Figure 5

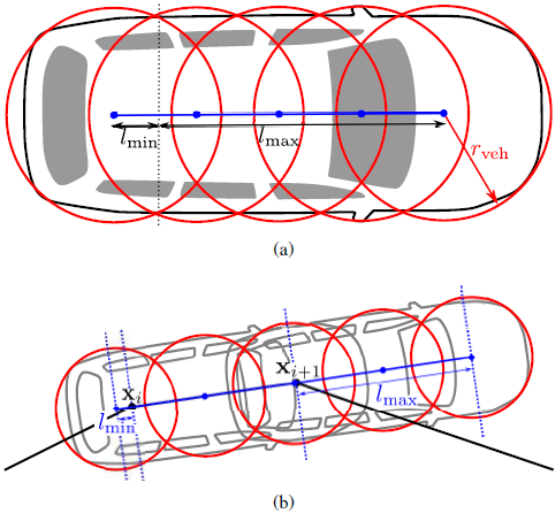


Figure 5: Circles to approximate the vehicles shape (Ziegler J. et al. 2014).

A number of k circles are chosen to approximate the vehicle shape in the interval of $[x_i, x_{i+1}]$. In the case of n discrete points and thus $n-1$ intervals,

$k(n-1)$ circles are obtained. In the case of o obstacles, $ko(n-1)$ constraints are obtained. In order to avoid collisions, it is required that the distance between the centre points of the circles and obstacles corresponds at least to the radius of the circles. Accordingly, the constraints for collision avoidance are as follows:

$$d_i \geq r_{veh}, i = 1 \dots ko(n-1).$$

In addition, there are the following constraints for curvature and acceleration:

$$\kappa(t) \leq \kappa_{max}, i = 1 \dots n$$

$$\kappa(t) \geq \kappa_{min}, i = 1 \dots n.$$

$$\|\ddot{x}\|^2 \leq a_{max},\ i = 1 \dots n.$$

Accordingly, there are overall $3n + ko(n-1)$ constraints. The resulting optimisation problem may e.g. be solved through a least squares solver.

3.1.2 Tailoring of the enabler for the ULM vehicle

Since the trajectory planning has been integrated and tested at first on the Ulm vehicle, the concept has been adapted for the environment resulting from the perception modules used with this demonstrator. The spatial cost term, which was based on the boundary lines, has been replaced by

$$j_{offs}(x(t)) = |d_{center}(x)|_2^2.$$

As a consequence, the orthogonal distance from the centreline is penalized. This change is made because the digital map used for the Ulm vehicle has no boundary lines but a reference line (i.e. the lane's centreline) instead.

Further, the nonlinear yaw rate term is moved away to penalize the rate of rotation around the gravity centre of the car, because it can cause several undesired affects, such as slower convergence etc.

Because of similar reasons, the constraints for the curvature are also moved away, to avoid such “very nonlinear” terms in the cost function.

Another adaption is that the optimisation procedure to solve the problem has changed to a self-written optimisation routine, instead of an open source software. Currently, last thoughts are made about which optimisation software will be used.

3.1.3 Implementation

In the section “Concept” there was stated that the cost function has the form

$$L = \int_0^T j_{offs} + j_{vel} + j_{acc} + j_{jerk} + j_{yawr}\, dt.$$

But since it is difficult (or too computationally expensive) to evaluate the integral, one approximates the integral by a sum

$$L = \sum_{i=0}^{N} \mathrm{j}_{offs,i} + j_{vel,i} + j_{acc.i} + j_{jerk,i} + j_{yawr,i}\ .$$

Like also mentioned in the Concept section, the yaw rate term is not included yet.

The algorithm was already implemented inside the software framework of Ulm. The mainly used inputs to the trajectory planner are the pre-stored digital map, the vehicle tracks, to provide positions velocities acceleration

etc. of other traffic participants and the state of the ego vehicle provided via GPS and ego motion services. Figure 6 present the module as a function block.

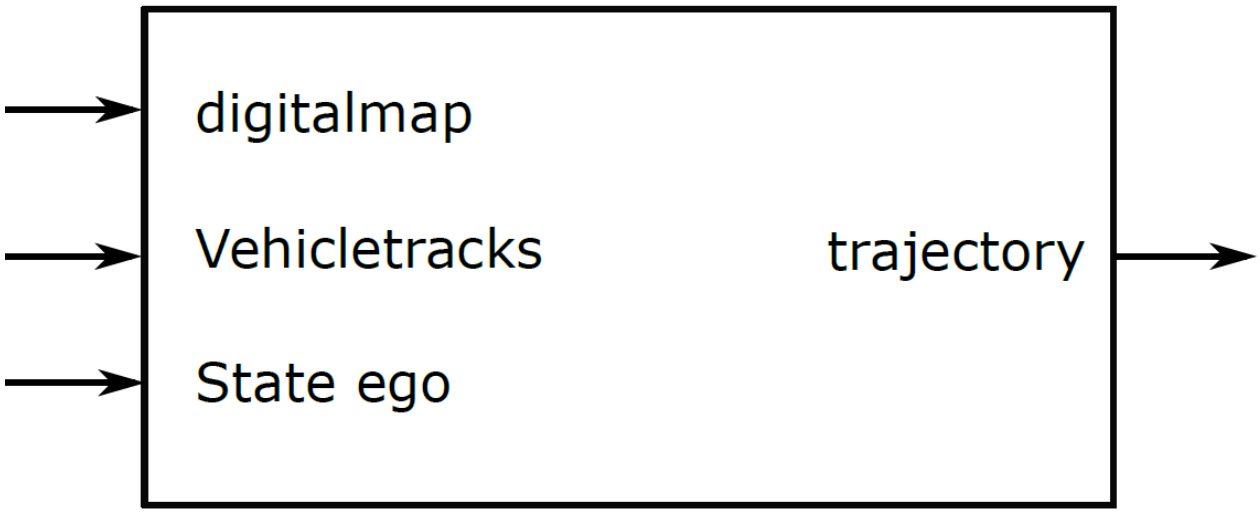


Figure 6: function block for the trajectory planning module

The implementation was done with the programming language C++, to guarantee real time-capability of the algorithms. So far, the optimisation routine (SQP-method, see previous section) was implemented. Although parts of the problem have been implemented (cost function, acceleration constraints but no spatial constraints yet). Since many matrix operations must be done in the optimisation routine, one needs to handle matrices. To do so, one could either implement own matrix classes, or use an already existing library for this. For the trajectory planner, the library C++ linear algebra library “Eigen” is used, to deal efficient with matrices. For reasons of runtime, one has the opportunity to parallelise code, this means if the

processor has more than one core, one can split the computation into multiple threads, and let them process in parallel by the processor. But the generation of multiple threads can be expensive, so it should be worth it. So far, there are no regions of code parallelised by the trajectory planner.

Another important point is that the time needed for computation must be considered. Let's assume, that a trajectory is computed in x seconds. Then the vehicle can react to its environment just every xth second. Although the planning module should not be triggered again while not having finished one computation. Therefor an upper bound for the computation time x_upper seconds>= x seconds must be identified. Then the planner is triggered every x_upper'th second. Such a functionality is included within the "Automotive data time-triggered framework" (ADTF) used by Ulm.

Last but not least it is important to mention on which system the trajectory planning module has been implemented, see the table below.

Processor	Intel i7
Read-access-memory	32GB
Operating System	Ubuntu 16.04 LTS

These are just a few specifications important for runtime performance.

3.2 E4.1 - Planning and execution of safe manoeuvre (VED)

3.2.1 Concept

Since ULM and VED have evaluated that they could face technical issues in tailoring and integrating the E4.1 enabler (developed by ULM) into the VED vehicle, VED has decided to start implementing another module for planning and execution of safe manoeuvre as part of a mitigation strategy for the integration phase.

The algorithm developed by VED is inspired by the works detailed in [2] which deals with urban and rural situations and will be extended to highway situations. This algorithm proposes a novel approach for developing a new architecture for the planning and control stages. Specifically related to the planning stage, this work proposes a two-stage local planning system (as shown Figure 7 consisting of

- a first stage (pre-planning) that searches the best trajectory configuration based on the characteristics associated to the infrastructure and physical limitations of the vehicles;
- a second stage (real-time planning) where vehicle dynamics are introduced to tune the generated trajectory.

Then, a control stage will be implemented to smoothly follow the automated vehicle trajectory generated.

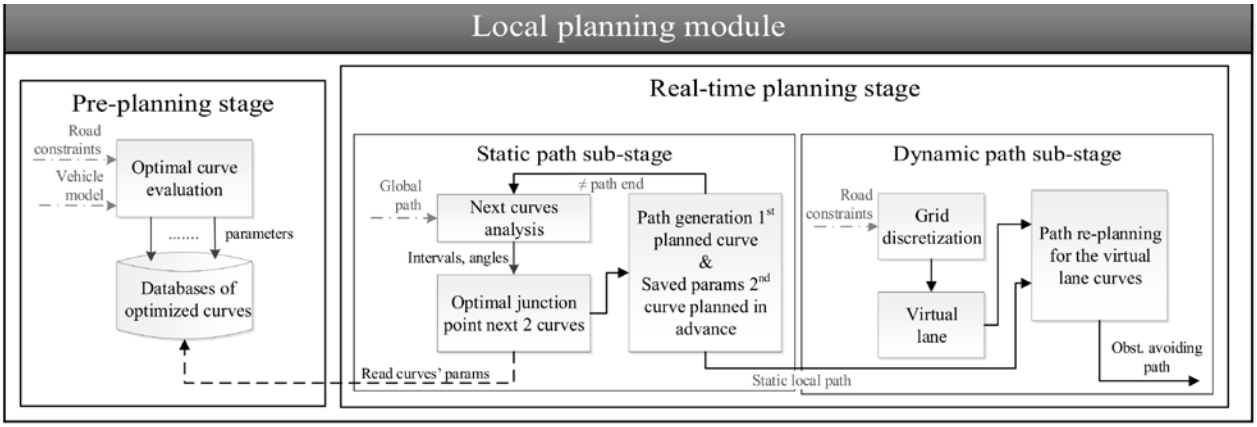


Figure 7: A detailed view of the path planning.

3.2.2 Implementation

This phase of the implementation includes the pre-planning stage.

This is the first stage of the proposed local planner. As it is not possible, in all the situations, to search for the best trajectory in real time due to the computational cost to generate all the possible trajectories, the goal of this module is to search the best possible trajectory for each kind of curve, taking into account parameters that are independent from the vehicle dynamics but dependent on a series of constraints that will be detailed later on. Therefore, this stage searches for the best trajectory achievable by the vehicle considering infrastructure limitation and vehicle constraints, generating the best curve for a given scenario.

At first, some intersection scenarios have been considered because it is easy to compare with the previous works in [2] and [3]. Some other urban scenarios, highway exists, entries and overtaking, obstacle avoidance or safe stops or safe strategies and manoeuvres will be considered throughout the development.

Trajectories are generated using Bézier curves. These are polynomial curves that use information of checkpoints, called control points, to be generated. Its simplicity, together with its low computational load allowed to implement and use them instead of other path generation techniques such as splines or clothoids [4], allowing a fast trajectory computation. This is a key feature in order to send the trajectory in real time to the control stage.

The goal of the Bézier curves is to build a continuous smooth and safe path to be tracked by the automated vehicle.

The following equation presents the mathematical definition for a n degree Bezier curve generation:

$$B(t) = \sum_{k=0}^{n} \binom{n}{k} (1-t)^{n-1} t^i P_i, \qquad t \in [0,1](1)$$

where *n* is the degree of the polynomial equation and *t* are the control points that define the curve. These kind of curves is very relevant for trajectory planning, in [4] and [5] a detailed description and a motivation of the use of such kind of curves to solve planning problems is provided.

Once the election of Bezier curves has been explained, the algorithm developed on the pre-planning stage is introduced. This algorithm seeks for the curve that better fits on a pre-defined intersection. A pseudo-code description of such algorithm is included in Figure 8

```
1:  Start
2:  Set configuration parameters of the urban scenario
3:  Calculate road limits and road constraints
4:  for lateral displacement = 0 to roadWidth/2-carWidth/2 by 0.1
5:     for d1=dist-0.1; d1>=0.2; d1-=0.1
6:        for d2=d1-0.1; d1>=0.1; d1-=0.1
7:           if(order==5) for d3=d2-0.1; d3>=0.1; d3-=0.1
8:              Calculate control points location / Make Bézier control points segmentation
9:              Check if the curve is feasible by the vehicle
10:             Check if the curvature constraints are fulfilled
11:             Check if the car doesn't invade the sidewalk
12:             while it is not complete the generation of the curve
13:                Generate the curve points
14:                Calculate the curvature and the derivative of the curvature of each point
15:             end while
16:             Calculate fitness with the cost function and check if it is the best solution
17: End
```

Figure 8: Algorithm 1: Path generation and selection algorithm.

Figure 9 is used to explain the algorithm. It shows an intersection scenario that is described geometrically with three points and the road width. Moreover, it is necessary to set some road constraints to keep the vehicle on the road while it is tracking the path.

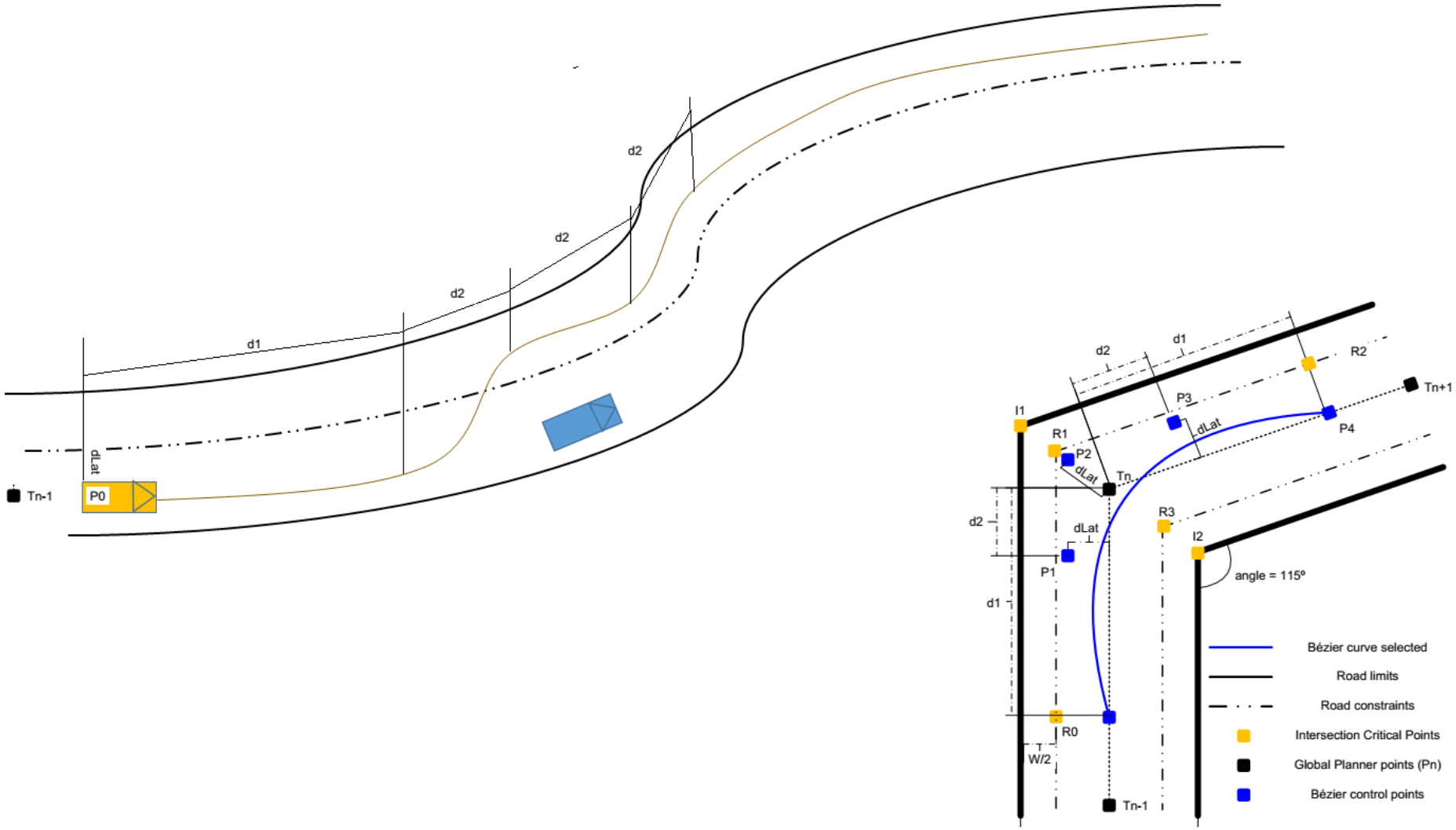


Figure 9: Visual representation of the algorithm described in Figure 8.

Firstly (as defined in Figure 8 in line 2), it is necessary to define the parameters of the urban scenario. These are as follows:

- the geometric information about the road (curvature, angles, etc.),
- location of the first (Tn-1), mid (Tn) and last (Tn+1) points (the space horizon),
- distances from the midpoint of the intersection to the control point pairs: d1 for P0 and P4, d2 for P1 and P3,
- lateral displacement for the internal control points (P1, P2 and P3).

Secondly (line 3), the road limits are calculated according to the road width and the intersection angle. In addition, road constraint limits are generated, considering an internal separation of half the car width with respect to the road limits. The control points are placed in the convex hull formed with these points, as indicated in the 4th Bézier property, in order to ensure the vehicle does not invade the sidewalk.

Thirdly (lines 4-7), with these nested loops it is possible to generate the different curves changing the parameters that define them:

- Lateral displacement, dLat, is changed from 0 to the road constraints edge each 0.2 meters in urban areas and each 0.5m in rural and highway areas.
- The distance to the external control points, d1, is changed from the distance between the first and mid intersections points to 0, i.e., the external control points (P0 and P4) are placed every 0.1 meters.
- The distance to the internal control points, d2, is changed from d1 to 0, i.e., the internal control points (P1 and P3) are placed every 0.1 meters from the location of the external control point to the midpoint of the intersection (Tn).
- If we want to generate 5th order Bézier curves (line 7), we need to change in the same way the distance d3 related to the pair of

innermost control points, in order to move the control points from the previous control points to the intersection midpoint.

Moving the control points both longitudinally and laterally over the segments that define the intersection, the Bézier curves are generated and later analysed to find the optimal one.

Fourthly (line 8), the algorithm sets the control points P1 to Pn-1 into the convex hull polygon formed by the points R0, R1 and R2 that describe the external band of the road (convex hull) [10], while the control points P0 and P1 are placed over the last and first straights before and after the intersection respectively (based on the first property of Bézier curves).

Fifthly (line 9), it is necessary to check if the curve is realizable by the vehicle, that is, the maximum curvature in all Bezier curve must be smaller or equal to the maximum curvature. We are assuming that the maximum curvature feasible is when the angle of the wheel is maximum (Equation of Ackerman), and this occurs at the middle of the curve.

Sixthly (line 10), in order to get a smooth trajectory and a good approach for a consecutive curves scenario, a constraint in the curvature at the beginning and at the end of the curve is defined. We consider that the curvature at these points should be almost equal to zero, that is, the switching from a curve stretch to a straight stretch should be imperceptible for car's passengers.

Seventhly (line 11), it is verified that the car doesn't invade the sidewalk. To that end, since the midpoint of the curve is the one of maximum curvature, it must be ensured a minimum distance of half of the vehicle width (W/2)

between the midpoint of the curve and the joint of the segments that define the road limits (I2), i.e., forcing the midpoint of the curve at the left side of the point I1 it is ensured to have the curve in the road constraints.

Eighthly (line 12), it begins the segmentation process, that is, the generation of the Bézier curve. Equation 1 is used to generate the n-degree Bézier curves (line 13), where n is the degree of the Bézier curve, until the segmentation process is completed according to the number of divisions defined for the curve.

Ninthly (line 16), to determine the optimality of a trajectory, it is necessary to define an intelligent function using a cost function to measure it. This cost function takes into account all the possible parameters independently of vehicle dynamics.

The first approach proposed takes into account the curvature, as it was considered in the previous works [2]. Specifically, it considers the measure of the curvature in the three most critical points: at the beginning, in the middle and at the end of the curve as follows:

$$fitness = curvature_{beginning} + curvature_{middle} + curvature_{end}$$

The goal, here, is to find the minimum possible curvature at the beginning and at the end of the curve, in order to get a smoother and more comfortable trajectory, achieving continuity between straight and curved segments [6]. Thus, we try to find the curve that minimizes the value returned by this cost function. This cost function maximises the fitness of the curvatures at the beginning, the middle and the end of the trajectory.

The analytical method is used during this work to calculate the curvature. This method calculates the curvature at each point of a curve whose coordinates (x, y) depend on this variable (see also [7] and [8]). Because of the simplicity of this equation, it makes its application in computational algorithms much faster, if the equation that defines the curve and if its derivative is previously loaded. Considering two-dimensional curves, the equation to obtain the curvature is defined as follows:

$$k(t) = \frac{\overrightarrow{Q'}(t) \times \overrightarrow{Q''}(t)}{\left\|\overrightarrow{Q'}(t)\right\|^3} = \frac{x'(t)\, y''(t) - y'(t)\, x''(t)}{\sqrt{\left(x'^2(t) + y'^2(t)\right)^3}}$$

Coming up next, as a second approach we use the derivative of the curvature in this cost function trying to minimize it at the three points mentioned above. The derivative penalizes sudden changes on the curvature, leading to a smoother and more comfortable path.

$$fitness = curvature'_{\,beginning} + curvature'_{\,middle} + curvature'_{\,end}$$

In addition, a real-time local planning will be developed. This module will be in charge of searching the optimal curve taking into account several parameters from the vehicle dynamics, such as speed, steering angle or frictional forces. To that end, this module will charge an already suitable trajectory from the pre-planning stage to tune it up for the given intersection scenario, taking into account vehicle dynamics parameters.

3.3 E4.2 - Learning of intention from driver (HMT)

In the context of AutoMate, the purpose of the algorithms to learn the driving intention from the driver is to enable the system to adapt its

automation strategies to the driver's preferences and guarantee a human expert-like and safe driving behaviour. In the following it is described how the TeamMate car learns from the driver during the second cycle of AutoMate.

3.3.1 Scenario and uses case where E4.2 is relevant

As shown in Table 1, Enabler E4.2 is needed to implement a support in perception from the automation to the human (A2H) to complement the ability of the driver to assess the impact of a risky behaviour.

One of the use cases of PETER scenario has been revised to highlight and clarify the role of E4.1 to implement this cooperation.

Peter is driving in a narrow rural road in Manual Mode. He approaches a tractor that causes limited visibility or the road. The TeamMate car detects a car approaching from the opposite lane. Since Peter is not aware of the car, he decides to overtake, and the TeamMate car detects his intention. In order to avoid an imminent collision, the TeamMate car informs Peter about the approaching vehicle and warns him about the risky manoeuvre. Peter suddenly becomes aware of the risk, and he does not perform the overtake until it is safe.

3.3.2 Concept

In the second cycle learning from the driver is understood as the learning of driving intentions e.g. "*lane change left*". The learning algorithms have been used to extend the *Driver Intention Recognition* (DIR) model from WP2. The DIR is basically a dynamic Bayesian Network which creates estimations about the intentions and future behaviour of human driver.

Since the intentions of the driver can't be observed directly, they are interpreted as a hidden process, which "emits" observable effects on the traffic situations, e.g., position of the ego vehicle, physical relations to other traffic participants, control actions of the driver, etc. The dependence of intentions and action variables on the past perception is described by a traditional sensor model.

The DIR model models the joint density distributions over actions, intentions and observations over an arbitrary length $T \geq 1$ as:

$$p(\boldsymbol{A}^{1:T}, \boldsymbol{I}^{1:T}, \boldsymbol{O}^{1:T}) = p(\boldsymbol{O}^1|\boldsymbol{A}^1, \boldsymbol{I}^1)p(\boldsymbol{A}^1, \boldsymbol{I}^1)\prod_{t=2}^{T} p(\boldsymbol{O}^t|\boldsymbol{A}^t, \boldsymbol{I}^t)p(\boldsymbol{A}^t, \boldsymbol{I}^t|\boldsymbol{A}^{t-1}, \boldsymbol{I}^{t-1}).$$

Where $\boldsymbol{O} = \{O_1, \dots, O_{n_O}\}$ denotes a set of continuous and/or discrete random variables that represent the observations of the current traffic situation, $\boldsymbol{A} = \{A_1, \dots, A_{n_A}\}$ denotes a set of continuous and/or discrete random variables that represent the actions of the driver, and $\boldsymbol{I} = \{I_1, \dots, I_{n_I}\}$ denotes a set of discrete variables that represent different intentions of the driver, e.g., "*lane following*", "*lane change left*".

The parameters of the DIR model can be learned offline via machine-learning methods from multivariate time-series of driving data, which are annotated by an expert in advance. Due to the annotation the intention is not hidden in training data and the model can be learned with *complete data*. The offline learned model should be able to recognize the driving intentions of the "average driver" of all human drivers contained in the training data.

The algorithms to learn from the driver performs an online learning to recalibrate the parameters of the DIR model during the driving process to fit

the model the driving behaviour and the driving preferences of the individual driver.

For online learning the data is not annotated by a human expert. Thus, the corresponding driving intentions remain hidden and the algorithm has to handle *incomplete data.* For each point in time t the online learning receives the current observation of the traffic situation $\boldsymbol{o}^t$.

To learn the driving intention, for example, for a lane change it has to be determined if and when a lane change happened. One option is to check the lane of the ego vehicle. If the ego vehicle actually changes its lane, it can be observed that $ego\_lane^{t-1} != ego\_lane^t$ and the intention I^t can be assumed as known. Since a lane change manoeuvre takes multiple time steps and an intention is usually formed even earlier, it cannot be assumed that whenever the lane is not actually changed that the intention is not *"lane change"*. Instead it should be estimated how many previous time steps *x*, which were observed before *t*, should also have the intention I^t.

The problem can be stated as the following smoothing problem:

$$p(\boldsymbol{I}^{t-x}|\boldsymbol{O}^{1:t}) \propto p(\boldsymbol{I}^{t-x}|\boldsymbol{O}^{1:t-x})p(\boldsymbol{O}^{t-x+1:t}|\boldsymbol{I}^{t-x})$$

Thus, except for the moment when the change of lane is actually observed the driver intention is a hidden variable and its value is missing for training. The learning of the model parameters can at best be considered as a semi-supervised learning problem. The Expectation Maximization (EM) algorithm is one suitable approach to unsupervised and semi-supervised learning of the model parameters. Assuming that for every observation there can be only one corresponding intention *hard EM* could be applied.

Considering a sequence of observations represented by $X = \{x_1, \dots, x_n\}$ representing, a sequence of hidden states represented by $= \{y_1, \dots, y_n\}$, and a corresponding model parameterized with Θ which defines probabilities $P_\theta(x_1, \dots, x_n, y_1, \dots, y_n)$ *hard EM* in general solves the following optimization problem:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \max_{y_1,\dots,y_n} P_\theta(x_1, \dots, x_n, y_1, \dots, y_n)$$

By alternately optimizing Θ and Y a local optimum for the problem can be found. The general optimization algorithm is:

1. Initialize Θ
2. Repeat until $P_\theta(x_1, \dots, x_n, y_1, \dots, y_n)$ converges.
    a. $(y_1, \dots, y_n) := \operatorname{argmax}_{y_1,\dots,y_n} P_\theta(x_1, \dots, x_n, y_1, \dots, y_n)$
    b. $\Theta := \operatorname{argmax}_\Theta P_\theta(x_1, \dots, x_n, y_1, \dots, y_n)$

Point *a* is called the E-step and is the evaluation of the expectation given the current parameter set Θ. For *hard EM,* the first E-step is to set the unknown values for the hidden intention to the expected intention.

Point *b* is the M-step, which modifies Θ in order to maximize the expectation that was computed during E-step [9].

For the essential part of the online learning the parameter update, the model parameters of the DIR are described with so called hyper parameters Θ.

Assuming X are the observations and Y the hidden states of a first order Markov process, similar to the DIR model, an observation at a certain time x^t depends only on the hidden state y^t and this state depends only on its previous state y^{t-1}. The model at a time slice can be described with the

hyper parameters $\Theta_{y|y_i}$ and $\Theta_{x|y_i}$. Where, for example, $\Theta_{y|y_i}$ expresses the probability that y^t takes a certain value if y^{t-1} had the state , $i = \{1, \dots, k\}$.

If the parameters are learned from a data set D the following conditional probability density is applicable:

$$p(\Theta_{y|y_i}, \Theta_{x|y_i}|D) = \prod_i p(\Theta_{y|y_i}|D)p(\Theta_{x|y_i}|D)$$

Since the parameters are independent from each other, $p(\Theta_{x|y_i}|D)$ and $p(\Theta_{y|y_i}|D)$ can be determined for each parameter separately.

For a Boolean variable the conditional posterior probability over the parameter after observing the learning data D would also be a Beta distribution.

$$p(\Theta_{y|y_0}|D) = Beta(\alpha_0 + \#_{y_0|y_0}, \alpha_1 + \#_{y_1|y_0})$$

Where α_0 and α_1 are the so called priors, and $\#_{y_0|y_0}$ as well as $\#_{y_1|y_0}$ are the counts that y^t takes the corresponding state if y^{t-1} was in state 0. The priors parameterize the Beta distribution and would be derived from a previously offline learned model. For discrete non-Boolean variables a Dirichlet distribution can be used.

$$p(\Theta_{y|y_0}|D) = Dir(\alpha_0 + \#_{y_0|y_0}, \alpha_1 + \#_{y_1|y_0} \dots, \alpha_k + \#_{y_k|y_0})$$

Due to the usage of Hard EM the at this point D can be considered as fully observed, thus the counts can be obtained from the data set.

The actual probability for the inference the can then be computed via the Bayes-Estimate [10]:

$$p(y_1^t|y_0^{t-1}, D) = \frac{\alpha_1 + \#_{y_1|y_0}}{\sum \alpha + \sum \#}$$

3.3.3 Improvements

In the first cycle only discrete variables were considered for the updating of the parameters of the previously offline learned DIR model. Since in the driving context many variables are rather continuous than discrete and the DIR model in general is able to deal with this kind of variables, the online learning was extended to also be able to update distributions for continuous variables.

The DIR uses the Gaussian distributions for the representation of continuous variables. The parameters update of model nodes which contain continuous variables is achieved by using the normal-inverse-Wishart distribution, which is the conjugate prior of a (multivariate) Gaussian.

3.3.4 Implementation

In D5.1 “TeamMate System Architecture incl. open API for 2nd cycle” is described that the components in the TeamMate architecture may exchange information based on socket communication. An interface to the risk assessment instance might be established in that way if each observed traffic situation could be labelled with a corresponding risk value. By doing so, it could be ensured that manoeuvres which contain too risky traffic situations are not considered and learning of unsafe behaviour is avoided.

For the communication with the DIR, to signalize that an updated model is ready, to compile both modules into one component is a feasible option. Thus, no socket communication would be required.

The online learning requires basically the same services as the DIR, meaning data from the sensor and communication platform in terms of belief state about the current state of the traffic situation and map of the road network.

The component then works as follows:

- A provided initially offline learned model is loaded
- Each cycle the received data is read into a ring buffer which can hold several seconds of data:
    - For every received data point $ego\_lane^{t-1} != ego\_lane^{t}$ is evaluated to check if a lane change happened
        - If a lane change is detected, the last x second are fetched from the buffer, transformed into a datatype which can be used for several learning procedures
        - All data points in the container are marked as lane change data
        - For every distribution in the loaded model the corresponding learning/parameter updating procedure is started
        - The updated model is stored and the inference engine is informed about the update

3.4 E5.1 - Online risk assessment (OFF + DLR + HTM)

The purpose of online risk assessment in AutoMate is the calculation of *safety corridors* that quantify the safety of the current and near-future traffic situation according to a metric of risk. These safety corridors are used by the

TeamMate car to assess and plan safe and feasible trajectories, leading to a set of algorithms that allow identifying safe and reasonable arrangements of the driving process.

3.4.1 Scenario and uses case where E5.1 is relevant

As shown in Table 1, Enabler E5.1 is needed to implement a support in perception from the automation to the human (A2H) to complement the ability of the driver to assess the risk of a situation. For this enabler, the same PETER scenario already described for E4.2 has been considered to highlight and clarify its role to implement the A2H cooperation.

Peter is driving in a narrow rural road in Manual Mode. He approaches a tractor that causes limited visibility or the road. The TeamMate car detects a car approaching from the opposite lane. Since Peter is not aware of the car, he decides to overtake, and the TeamMate car detects his intention. In order to avoid an imminent collision, the TeamMate car informs Peter about the approaching vehicle and warns him about the risky manoeuvre. Peter suddenly becomes aware of the risk, and he does not perform the overtake until it is safe.

3.4.2 Concept

In the following, let Δ denote a temporal step width and η_{max} denote a maximal step width, resulting in a desired prediction horizon $\eta_{max}\Delta$, and $\boldsymbol{V} = \{v_1, \dots, v_{n_V}\}$ denote a set of n_V objects (usually traffic participants) detected by the sensor platform of the TeamMate vehicle at some time step t.

As described in D3.3 “Concepts and algorithms incl. V&V results from 1st cycle”, the output of the online risk assessment at each time step t is a set $\boldsymbol{c}^{t:t+\eta_{max}\Delta}$ of *safety corridors* $\boldsymbol{c}^{t:t+\eta_{max}\Delta} = \left(\boldsymbol{c}^{t:t+\Delta}, \boldsymbol{c}^{t+\Delta:t+2\Delta}, \dots, \boldsymbol{c}^{t+(\eta_{max}-1)\Delta:t+\eta_{max}\Delta}\right)$. For the sake of readability and as envisioned for online risk assessment, we will silently assume that $\Delta = 1s$ and omit the addition of Δ in the following.

Each safety corridor $\boldsymbol{c}^{i:i+1}, t \leq i < \eta_{max}$ defines a region over a temporal interval $[i, i+1]$ for which the probability of collision between the TeamMate vehicle and a *single* object $v \in \boldsymbol{V}$ *or* the road boundaries is upper-bounded by two of user-defined thresholds δ_R and δ_v.

Formally, each safety corridor $\boldsymbol{c}^{i:i+1}$ is defined as a set of *polygonal lines* $\boldsymbol{c}^{i:i+1} = \left\{L_R^{t:t+\eta_{max}}, L_1^{i:i+1}, \dots, L_{n_V}^{i:i+1}\right\}$, where a polygonal line L should be understood as a *closed broken line*, i.e. a *polygon*, composed of a finite number of line segments, specified by a sequence of points $L = (A_1, \dots, A_k)$, where each $A_j \in L$ is defined as a pair $A_j = \left(x_j, y_j\right)$ denoting the x- and y-coordinates in a Cartesian coordinate system.

For a safety corridor $\boldsymbol{c}^{i:i+1} = \left\{L_R^{t:t+\eta_{max}}, L_1^{i:i+1}, \dots, L_{n_V}^{i:i+1}\right\}$, $L_R^{t:t+\eta_{max}}$ denotes a polygonal line derived from the road boundaries that *encloses* a region in which the probability of collision with the road boundaries is below the threshold δ_R. Each $L_j^{i:i+1}, j = 1, \dots, n_V$ denotes a polygonal line that *excludes* a region for which the probability of collision with a corresponding object is below a threshold δ_V. As such, the joint set of the set of polylines $\left\{L_R^{t:t+\eta_{max}}, L_1^{i:i+1}, \dots, L_{n_V}^{i:i+1}\right\}$ implies a continuous “*safe area*” for the temporal interval $[i, i+1]$, in which the probability of collision with *any* object is upper-bounded by a probability

$$\delta = 1 - (1 - \delta_R)(1 - \delta_v)^{n_V}$$

that can be used by the path planning algorithm to plan current and future trajectories.

A visual example of a safety corridor is provided in Figure 10. In this example, the safety corridor is composed of a polyline $\boldsymbol{L_R^{t:t+\eta_{max}}}$ associated with the lane boundaries and two polylines $\boldsymbol{L_1^{t+1:t+2}}$ (blue) and $\boldsymbol{L_2^{t+1:t+2}}$ (green) associated with two traffic participants. The grey hachured area represents the area of collision-free travel. We note that a safety corridor abstracts from the dimension of the TeamMate vehicle itself, which should instead be taken into account by the path planning algorithms.

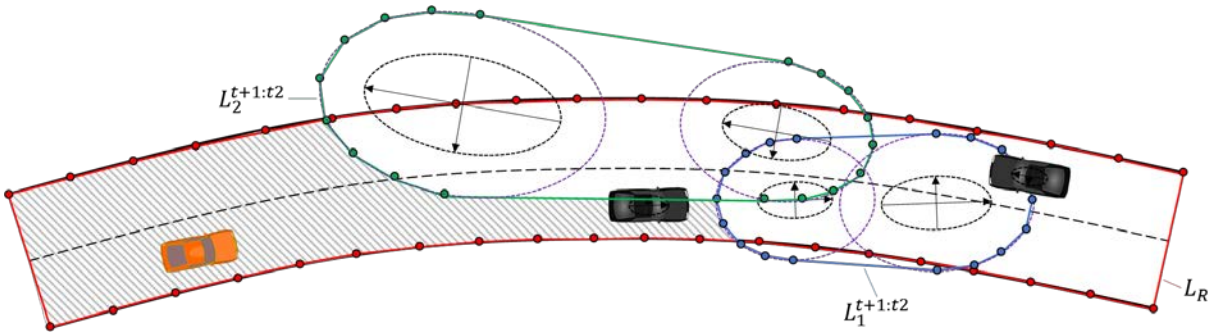


Figure 10: Exemplary visualization of a safety corridor for a temporal interval $[t+1, t+2]$.

To derive the safety corridors, the online risk assessment relies on a prediction of the temporal and spatial evolution of the traffic situation, provided by the Vehicle and Situation Modelling Module (for a detailed description, please refer to D2.4 “Sensor Platform and Models incl. V&V results from 2nd cycle”).

For this, we assume that at each point in time t, the sensor platform provides a belief state $p(\boldsymbol{X}_v^t|\boldsymbol{o}^{1:t})$ for each $v \in \boldsymbol{V}$, estimated from the sensor observations received up to the current point in time $\boldsymbol{o}^{1:t}$, where $\boldsymbol{X}_v^t = \{X_v^t, Y_v^t, \Theta_v^t, V_v^t, A_v^t, W_v^t, S_{L_v}^t, S_{W_v}^t, E_v^t, C_v^t, L_v^t\}$, as described in Table 2.

Table 2: Description of variables for the representation of an object $v \in V$ in the vicinity of the TeamMate vehicle considered for the first cycle.

Variable	Type	Unit	Description
X_v	Continuous	[m]	X-coordinate of the centre of the object $v \in \boldsymbol{V}$ in a two-dimensional spatial coordinate system relative to the position of the TeamMate vehicle
Y_v	Continuous	[m]	Y-coordinate of the centre of the object $v \in \boldsymbol{V}$ in a two-dimensional spatial coordinate system relative to the position of the TeamMate vehicle
Θ_v	Continuous	[rad]	Yaw-angle relative to a reference axis
V_v	Continuous	[m/s]	Longitudinal velocity along the objects heading
A_v	Continuous	[m/s²]	Longitudinal acceleration
W_v	Continuous	[rad/s]	Yaw-rate

S_{L_v}	Continuous	[m]	Length (along the objects x-axis)
S_{W_v}	Continuous	[m]	Width (along the objects y-axis)
E_v	Binary	{true,false}	Binary flag, whether the object $v \in \boldsymbol{V}$ exists in the current traffic scene.
C_v	Discrete	$\{0, \dots, \lfloor C_v \rfloor\}$	Classification of the object $v \in \boldsymbol{V}$, e.g. PKW, LKW, VRU, etc.
L_v	Discrete	$\{0, \dots, \lfloor L_v \rfloor\}$	The lane, the object $v \in \boldsymbol{V}$ is currently located in, e.g. fast or slow lane on a two-lane road

For the actual prediction, let $\boldsymbol{S}_v = \{X_v, Y_v, \Theta_v, V_v, A_v, W_v\}$ denote a six-dimensional state for any $v \in \boldsymbol{V}$. At each point in time t, the Vehicle and Situation model is used to infer a sequence of future states $p\left(\boldsymbol{S}_v^{t+i} | E_v = true, \boldsymbol{o}^{1:t}\right), i = 1, \dots, \eta_{max}$.

The online risk assessment then uses these predictions to derive a region that encompasses the probable future location of the object v, in respect to its position, dimension, and orientation, with a probability of $(1 - \delta_v)$, i.e., we aim that the probability that an object $v \in \boldsymbol{V}$ is located outside of the predicted region is upper-bounded by δ_v.

3.4.3 Improvements

3.4.3.1 Safety Corridor Dynamic Objects

Let $p(\boldsymbol{S}_v^{t+i}|E_v = true, \boldsymbol{o}^{1:t})$ denote a predicted state for some object $v \in \boldsymbol{V}$, representing a six-dimensional multivariate Gaussian distribution. For simplicity of notation, we omit denoting the indices and conditions, such that in the following we simply use that $p(\boldsymbol{S}) = p(X, Y, \Theta, V, A, W)$ instead of $p(\boldsymbol{S}_v^{t+i}|E_v = true, \boldsymbol{o}^{1:t})$.

To derive a safety region that encompasses the bounding box of the object, we need to consider the size of the object. In the first cycle, we abstracted from our beliefs concerning the yaw angle Θ of the object and extended our beliefs about the location of the vehicle by the radius of a circle, resulting from a rotation of the vehicle (as shown in Figure 11 (a)).

Unfortunately, and especially for large objects like e.g. LKWs, this resulted in excessive safety regions, therefore artificially restricting the area of safe travel.

To overcome this limitation, we improved our approach by taking our beliefs concerning the yaw angle into account. For this, we use the belief state $p(\boldsymbol{S})$ to first derive a three-dimensional belief state over the location and pose:

$$p(X, Y, \Theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(X, Y, \Theta, v, a, w)\, dv\, da\, dw.$$

The marginalized belief state represents our beliefs about the location of the centre of the bounding box of the object and the yaw angle. Knowing the length l and width w of the object, we can use this belief state to derive

belief states about the locations of the *corners* of a bounding box, covering the true location, orientation and dimension of the object.

Let l denote the length of a vehicle and w denote its width, and let $i = 0, \dots, 3$ denote the index of one of the four corners, given a yaw angle θ and assuming the centre of the bounding box is the origin, the position of the corners is given by:

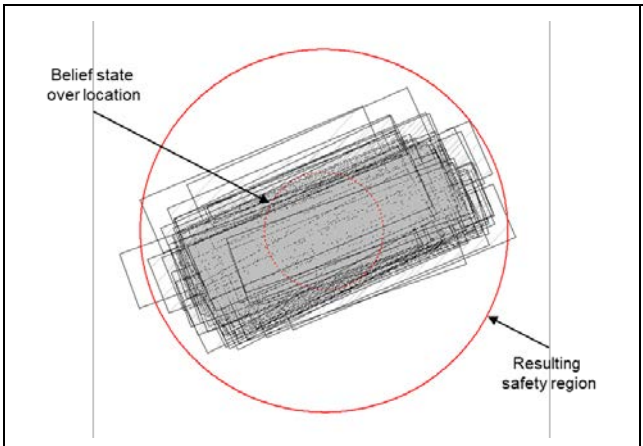
$$x_i = (-1)^i \cdot \frac{l}{2} \cdot \cos\theta - (-1)^{(i+1)} \cdot \frac{w}{2} \cdot \sin\theta,$$

$$y_i = (-1)^{\left\lfloor \frac{i}{2} \right\rfloor} \cdot \frac{l}{2} \cdot \sin\theta + (-1)^{\left(\left\lfloor \frac{i}{2} \right\rfloor + 1\right)} \cdot \frac{w}{2} \cdot \cos\theta.$$

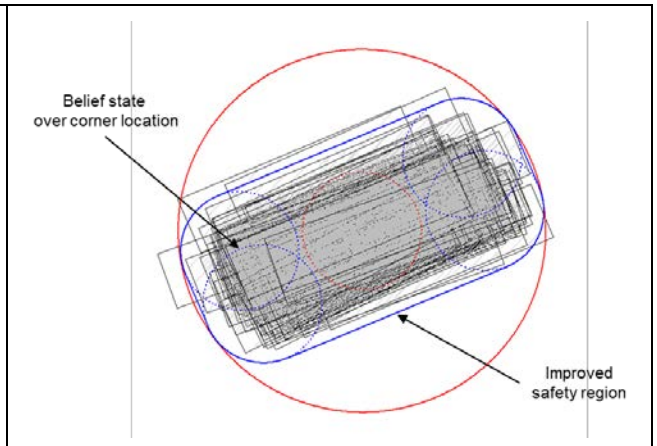
We use the technique of unscented transformation (as described in D2.2 "Sensor Platform and Models including V&V results from 1st cycle") to approximate a set of belief states $p_i(X_i, Y_i), i = 0, \dots, 3$ representing our beliefs concerning the location of the objects corners. As previously described in D3.3 "Concepts and algorithms incl. V&V results from 1st cycle", we then perform an Eigen-decomposition for each $p_i(X_i, Y_i)$ separately to derive the Eigen-values and -vectors and construct a polygonal approximation L_j of the coverage. The convex hull L of these polygons then provides a reasonable estimate of the coverage of the vehicle itself.

As we have that, each bounding box is the convex hull of the four corners, the convex hull of the coverages of the four corners encloses all vehicle bounding boxes. A comparison of both approaches is shown in Figure 11. Rectangles denote the bounding boxes of an LKW resulting from sampling potential centre-locations and orientations. The approach in cycle 1 results in

an overestimation of the safety area (a), the improved approach provides a much more reasonable result (b).



(a)



(b)

Figure 11: Comparison between approaches in cycle 1 and cycle 2 for deriving the safety area from a predicted belief state.

The error induced by approximating the area of the ellipse by a polygon can easily be obtained from the reference circle. Here we have that the area of reference circle with a radius $r = 1$ is given by πr^2. The area of a polygon with n equidistant vertices along the arc is given by:

$$A = \frac{1}{2} n r^2 \sin\frac{360°}{n}$$

We opt to use a number of 100 vertices, which results in an error $< 0.1\%$.

3.4.3.2 Safety Corridor Between Road Boundaries

A module for extracting the safety corridor from road boundaries was presented in D3.3 in the first project cycle.

The system first extracted map data from a map file in OpenDrive format. In the second step, the ego-vehicle pose was matched into the map to get the ego-vehicle lane. Depending on the lane marking type, road boundaries polylines were sampled from the ego-vehicle lane marking.

One limitation of this approach was that the ego-vehicle pose uncertainty was not taken into account.

In the second project cycle, we address this issue by using the ego-vehicle pose uncertainty as the uncertainty of the ego-lane centre. After that we sampled 2 lanes by shifting the ego-lane centre with quantiles z_{δ_V} and $z_{1-\delta_V}$of the given collision probability threshold δ_V and $1-\delta_V$ (see blue and red polylines in Figure 12).

At the end, we generated the safety corridor from road boundaries by sampling respectively the closest left (left blue polyline in Figure 12) and right (right red polyline in Figure 12) lane markings from the 2 lanes generated above. Figure 12 illustrates an example of generated safety corridor assuming that the ego-pose uncertainty is $\pm 1m$.

The lane markings from the map do not match the one in the camera image because the map's representation is in a 2 dimensional space. The missing height information into the map lead to projection error into the image plane. For simplification, we used the height of the camera above the road as the height of the map data.

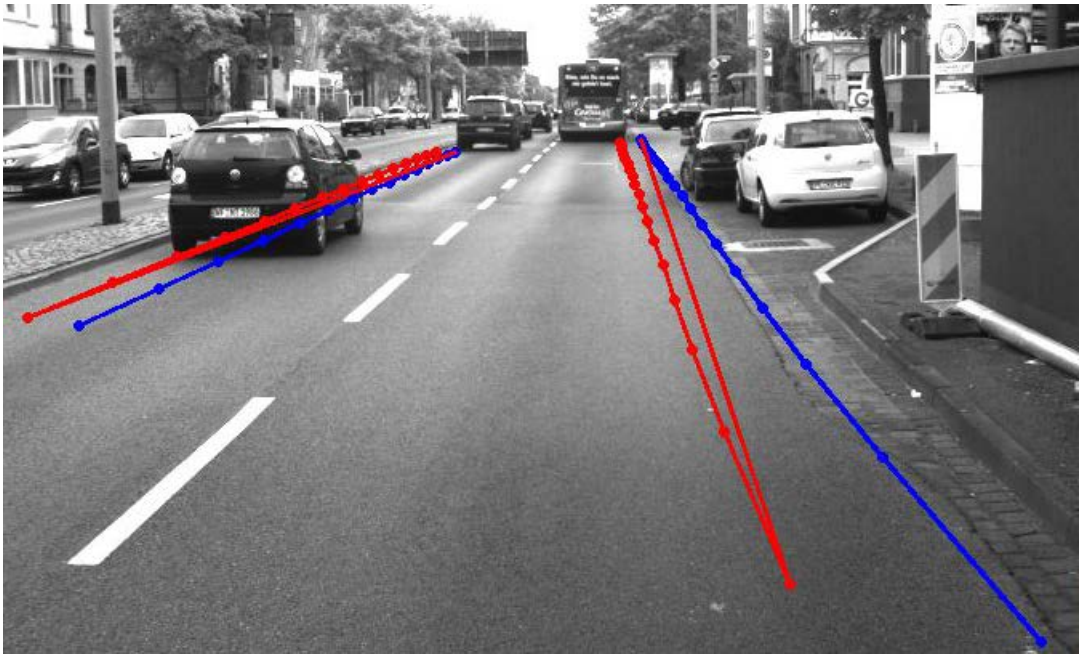


Figure 12: Example of a generated safety corridor assuming that the ego-pose uncertainty is ±1m.

3.4.4 Implementation

As described in D5.1 “TeamMate System Architecture incl. open API for 2nd cycle”, integration of components in the TeamMate architecture is planned on a client-server model. Each component may provide services to and require services from other components, realized by the exchange of information based on socket communication. Due to the tight coupling between the prediction of the spatial and temporal evolution of the traffic scene and online risk assessment, we currently opt to integrate both functionalities in a single component for online risk assessment.

The online risk assessment component requires services from the sensor and communication platform in terms of belief state about the current state of

the traffic situation. Furthermore, the component requires a high precision map of the road network, allowing inferring the future behaviour of traffic participants.

Provided with the most actual sensor data, the component then works as follows:

- Each cycle (currently repeated every 50ms) begins with an initialization:
    - Assuming that the sensor and communication platform uniquely identifies each detected object $v \in \boldsymbol{V}$ with an ID, the prediction of the spatial and temporal evolution of the traffic scene maintains an individual Hidden Markov Model (HMM) for each object $v \in \boldsymbol{V}$ to infer the belief state over potential behaviour hypothesis of the object. The component then checks, whether any HMM can be discarded due the corresponding object no longer being present (e.g., due to an object leaving the sensor range) or whether any new HMM must be created (e.g., due to an object entering the sensor range).
    - For each detected object, the corresponding HMM is used to update its beliefs about the most probable future behaviour hypothesis, e.g., in the case of the Peter scenario, whether the object intends to stay on its own or change to an adjacent lane.
    - For each detected object $v \in \boldsymbol{V}$, having access to $p(\boldsymbol{S}_v^t)$, provided by the sensor and communication platform, the component is

used to derive the coverage polygon L_v^t from the marginalized belief state $p(X_v^t, Y_v^t, \Theta_v^t)$.

- After the initialization is complete, the algorithm attempts to sequentially provide the safety corridors $\boldsymbol{c}^{t+i\Delta:t+(i+1)\Delta}, i = 0, \dots, \eta_{max} - 1$. For this, the following steps are performed for each object $v \in \boldsymbol{V}$:
  - Let h_{max}^t denote the most probable behaviour hypothesis. Based on the belief state $p(\boldsymbol{S}_v^{t+i\Delta} | h_{max}^t)$, the Vehicle and Situation Modelling Module is used to predict the next future state $p\left(\boldsymbol{S}_v^{t+(i+1)\Delta} | h_{max}^t\right)$ and the online risk assessment is used to derive the corresponding coverage polygon $L_v^{t+(i+1)\Delta}$ from the marginalized belief state $p\left(X_v^{t+(i+1)\Delta}, Y_v^{t+(i+1)\Delta}, \Theta_v^{t+(i+1)\Delta} | h_{max}^t\right)$.
  - Using both coverage polygons $L_v^{t+i\Delta}$ and $L_v^{t+(i+1)\Delta}$, the online risk assessment derives their convex hull $L_v^{t+i\Delta:t+(i+1)\Delta}$, representing the assumed coverage over the temporal interval $[t + i\Delta: t + (i + 1)\Delta]$.
- The coverage polygons $L_v^{t+i\Delta:t+(i+1)\Delta}$ for each object $v \in \boldsymbol{V}$ are then combined with polygonal line derived from the road boundaries, $L_R^{t:t+\eta_{max}}$, to form the safety corridor $\boldsymbol{c}^{i:i+1} = \left\{L_R^{t:t+\eta_{max}\Delta}, L_1^{t+i\Delta:t+(i+1)\Delta}, \dots, L_{n_V}^{t+i\Delta:t+(i+1)\Delta}\right\}$.
- The process is repeated until either all safety corridors $\boldsymbol{c}^{t:t+\eta_{max}\Delta} = \left(\boldsymbol{c}^{t:t+\Delta}, \boldsymbol{c}^{t+\Delta:t+2\Delta}, \dots, \boldsymbol{c}^{t+(\eta_{max}-1)\Delta:t+\eta_{max}\Delta}\right)$ have been computed, or the available time is over and a new cycle will be started.

After the cycle is completed or has been aborted, the online risk assessment provides the most actual and complete safety corridor as a service to other components.

4 Validation of WP3 enablers

4.1 E4.1 - Planning and execution of safe manoeuvre (ULM)

To guide the vehicle safely through the environment, the stated optimisation problem is solved in every planning step. To validate the trajectory planner we will do simulation runs. Therefor the planner drives the simulated vehicle through a curvy street. We regard the development of the cost function, as well as the constraints violation and the computation time.

As mentioned in the section “Implementation” of the trajectory planning concept, the cost function is as follows

$$L = \sum_{i=0}^{N} \mathrm{j}_{offs,i} + j_{vel,i} + j_{acc.i} + j_{jerk,i}.$$

Here the term about the yaw rate is not inserted, since it is not implemented.

The evolution of the cost term is an indicator for how well the optimiser performs, as well as for the quality of the resulting trajectory.

4.1.1 Dataset for validation

To evaluate the performance of the trajectory planner, we will use the digital map of Ulm. This map is included into a simulation framework written in matlab and includes streets of different quarters in Ulm. The validation of the planner will take place on the lane through “Lehr” in Ulm. Therefor the reference line (center line) of this route was extracted from the map and the speed limit was set to 70km/h. There are no other traffic participants on the

street, since (like already mentioned) spatial constraints are not included yet.

It is important to mention, that just the left lane's center line was extracted from the digital map. The boundary lines as well as the right lanes center line are generated synthetically.

4.1.2 Results

For the results we regard a curved section of the map (roundabout section). On this section we show 4 different snapshots and optimisation information for the trajectories.

Snapshot 1

Iter	Obj	Con
0	2.187804e+01	0.000000e+00
1	2.179980e+01	0.000000e+00
2	2.175217e+01	0.000000e+00
3	2.170850e+01	0.000000e+00
4	2.167912e+01	0.000000e+00
5	2.166159e+01	0.000000e+00
6	2.165548e+01	0.000000e+00

7	2.165209e+01	0.000000e+00
8	2.165208e+01	0.000000e+00

Snapshot 2

Iter	Obj	Con
0	2.395762e+01	0.000000e+00
1	2.394100e+01	0.000000e+00
2	2.393275e+01	0.000000e+00
3	2.392869e+01	0.000000e+00
4	2.392577e+01	0.000000e+00
5	2.392568e+01	0.000000e+00
6	2.392561e+01	0.000000e+00
7	2.392561e+01	0.000000e+00

Snapshot 3

Iter	Obj	Con
0	2.041375e+01	0.000000e+00

1	2.041096e+01	0.000000e+00
2	2.041072e+01	0.000000e+00
3	2.041069e+01	0.000000e+00

Snapshot 4

Iter	Obj	Con
0	1.367673e+01	0.000000e+00
1	1.367673e+01	0.000000e+00
2	1.367673e+01	0.000000e+00

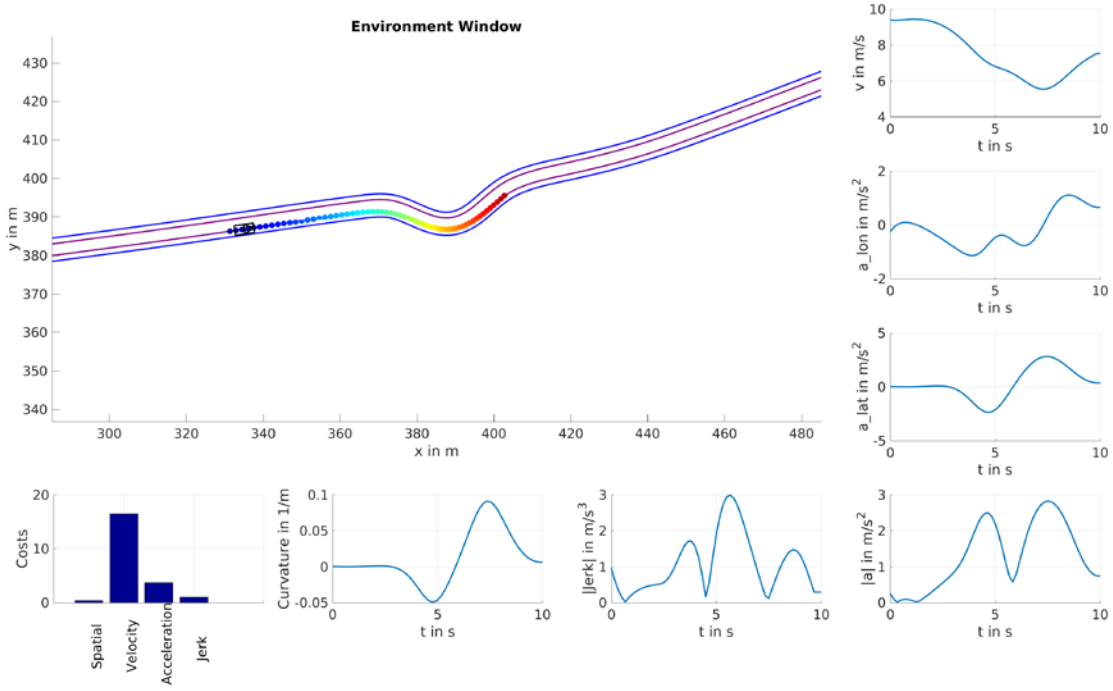


Figure 13: Simulation result (snapshot 1)

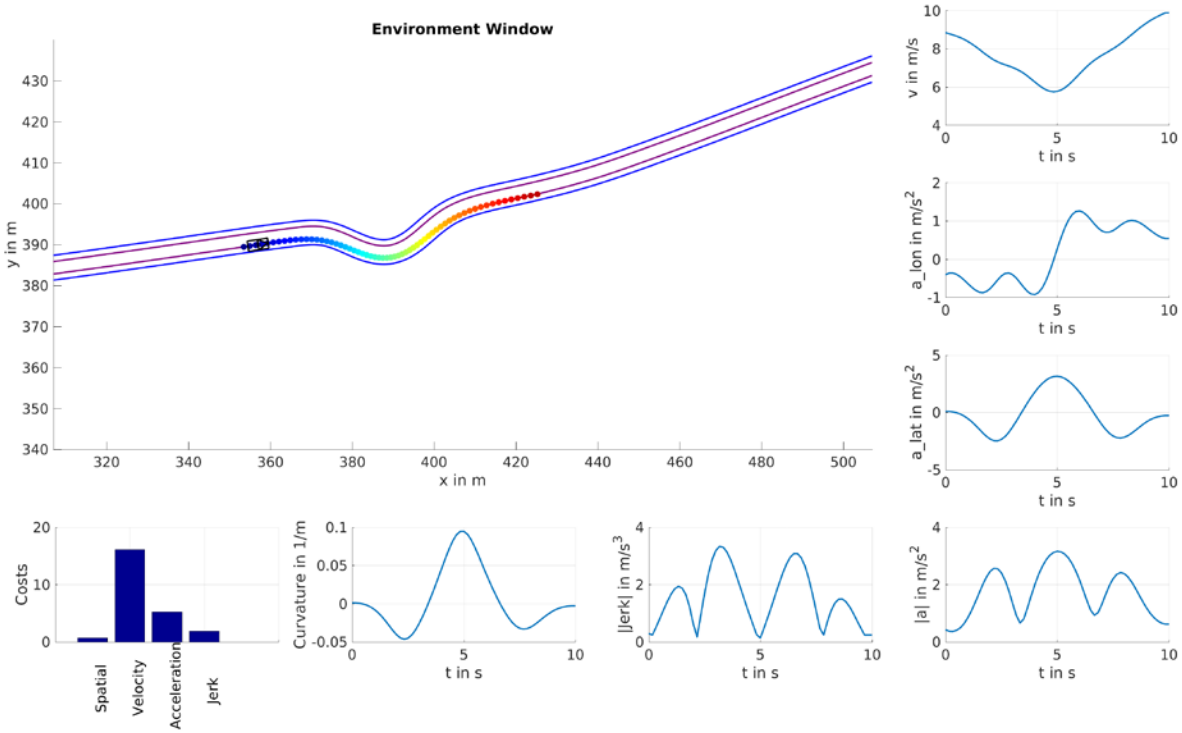


Figure 14: Simulation result (snapshot 2)

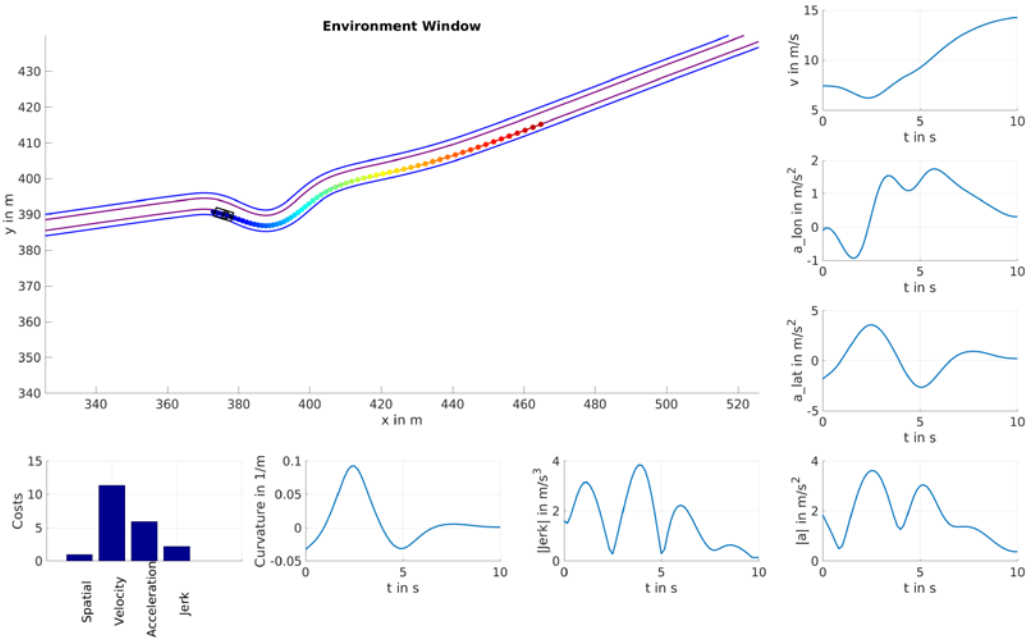


Figure 15: Simulation result (snapshot 3)

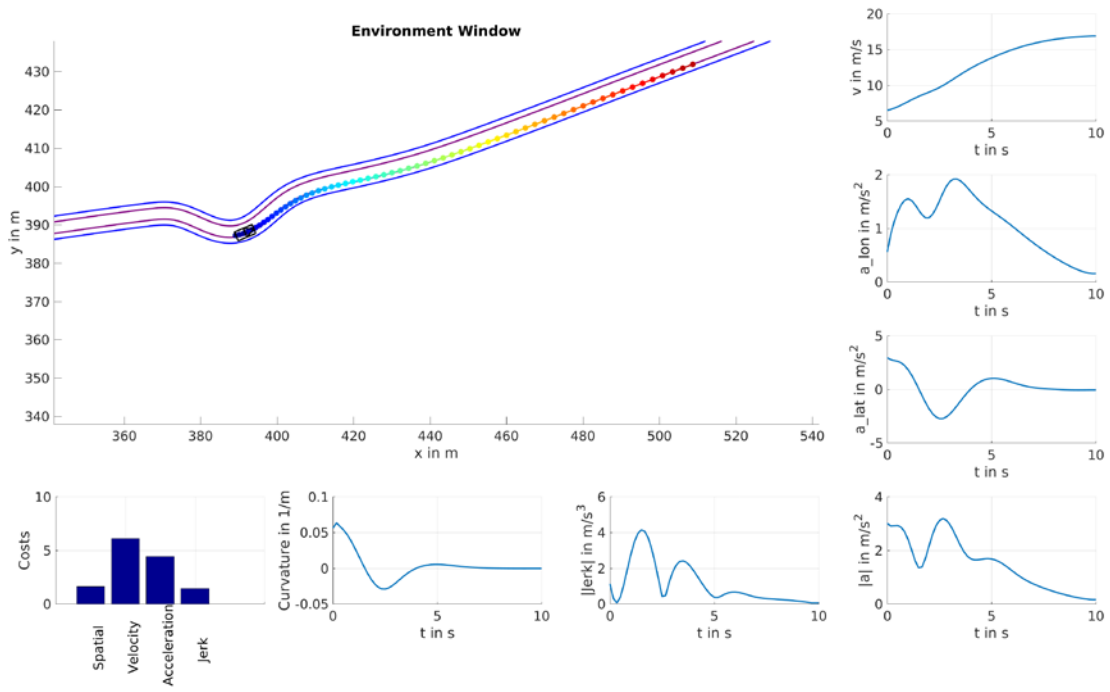


Figure 16: Simulation result (snapshot 4)

The big window (Environment window) shows the environmental data. The smaller windows show the (from top right to bottom right to bottom left) velocity, longitudinal acceleration, lateral acceleration, absolute acceleration, absolute jerk, curvature and costs of the trajectory. The results look satisfying. The color-coding of the trajectory is for temporal parametrisation. The trajectory contains points over time, at the end of the horizon the points are depicted as red and in the beginning as blue.

The iteration output shows that the costs are decreasing and the constraint violation (Con) is always zero, what means that the maximum acceleration (here chosen to 5m/s^2) is never exceeded. On the 4th snapshot the costs take always the same values, even at the last iteration. This must be fixed in

the future. Here it might be that, breaking criteria is not parametrized properly or has to be extended in such a way, that if there is no decrease in the costs, the optimiser breaks the optimisation procedure. Be aware, that the costs in the 0^{th} iteration are the costs after the first iteration step!

So far this is the current state of the trajectory planning; several extensions and bug fixes have to be made in the future.

4.2 Planning and execution of safe manoeuvre (VED)

In order to validate the proposed algorithm, a comparison among the different cost functions that can be used to determine what is the best trajectory.

Some experiments have been carried out on urban scenarios of multiple consecutive intersections, including ones with the same direction of rotation and others with the opposite Figure 13 represents the tested scenario. There, a lane road defined by squared intersections is assumed, where the lane width is 3 meters to fit with real scenarios. It is composed of two right turns and two left turns. Path planning considering pairs of curves was tested, taking into account a higher horizon of vision that considers the following curve and its respective turning angles and their direction of rotation. Upper part of Figure 13 shows both the planned and tracked path with the previous team's implementation [2, 3] and the proposed one. Whereas, in the lower part the curvature and its derivative are presented. In the paths sub-figure the red continuous line shows the path generated with the previous team's implementation, detailed in [2, 3], while the blue continuous line represents

the one generated with the proposed planning system. The dotted lines represent the paths tracked by the vehicle during the simulation. The planned paths have been generated with 4th degree Bezier curves. This is the minimum degree in order to generate continuous enough paths to be joined for several curves. Meanwhile, in the lower sub-figure, the red and blue continuous lines represent the curvatures for both planning algorithms, the dashed lines represent their curvature derivative respectively. Table II summarizes the results obtained. First, an improvement on the curvature is appreciated, being this clearly manifested in lower peak curvature values: 0.4295 m-1 is the maximum curvature with the previous team algorithm, whereas 0.2891 m-1 is the curvature with the proposed one. It means an improvement of a 32%. Furthermore, it can be also appreciated an improvement on the curvature derivative, being manifested in both peak and mean values. These improvements rely on the generation of smoother planned paths, allowing the vehicle to track more comfortable trajectories. This behavior can be noticed on Figure 13, where the path is better tracked by the vehicle in our approach based on the controller developed in [3].

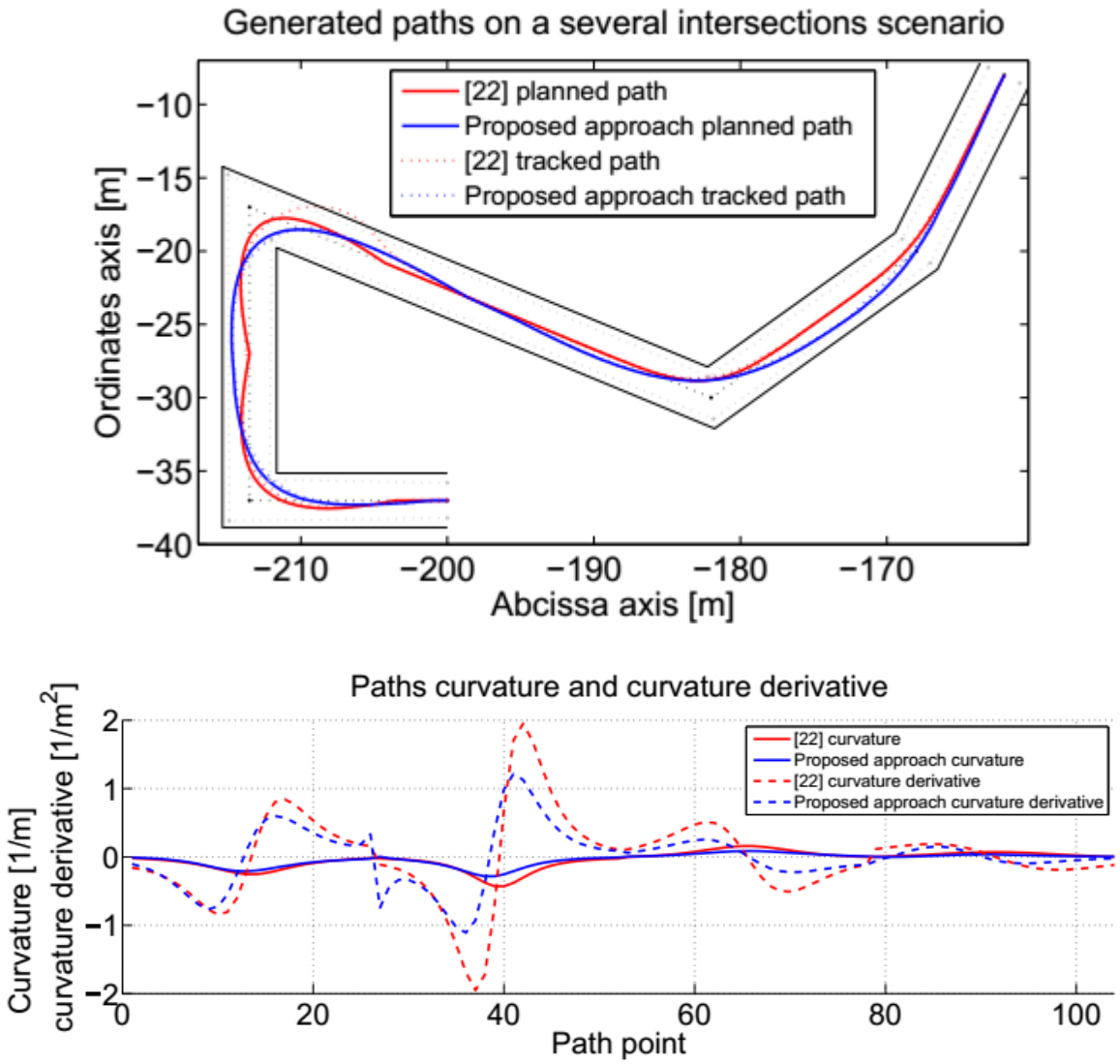


Figure 13 : (up)Example of urban scenario, (down) the study of the curvature and the derivatives of the curvature (VED).

Table 3: Curvature derivatives measurments

Algorithm	Measurements			
	$\lvert\mu_k\rvert\ (1/m)$	$\lvert k_{max}\rvert\ (1/m)$	$\lvert\mu'_k\rvert\ (1/m^2)$	$\lvert k'_{max}\rvert\ (1/m^2)$
González et al., 2014	0.0929	0.4295	0.4286	1.9522
Proposed approach	0.0657	0.2891	0.2916	1.2554

The validation has not been performed yet for this module for the Martha scenario. .

For the validation, different metrics are considered: the curvature, the derivative of the curvature, the lateral acceleration and the jerk.

The validation will be performed by considering one or more of these cases:

- Different highway scenarios to validate the different cost functions (curvatures, highway entries, highway exit, overtaking ...).
- 3rd, 4th and 5th order Bezier curves are explored.
- Different distances (emulating the decision process) are used to validate the expected effect on the trajectory planning.

4.3 E4.2 - Learning of intention from driver (HMT)

According to validation plan described in D3.4 “Metrics and plan for V&V of the concepts and algorithms in the 2nd cycle” online learning is validated by considering the performance of the updated model on a test data set D_{Test} in comparison to an initial model.

An initial model was trained with a separate data set D_{Train} which contains data from one of more drivers. The updated model is created by updating the

initial model with data from a data set D_{Train_Online}. The data sets D_{Test} and D_{Train_Online} contain only data from one and the same driver.

Each data set consists of m trials, where each trial is a sequence of recorded traffic situations consisting of a number of $n_j, j = \{1, \dots, m\}$ data samples $d_j^k = \left(i_j^k, b_j^k, \boldsymbol{o}_j^k\right), k = \{1, \dots, n_j\}$. Where $\boldsymbol{o}_j^k$ represents the available sensor input, i_j^k the driving intention, and b_j^k driving behaviour. For D_{Train} and D_{Test} the assumed correct driving intention and driving behaviour are annotated in advance by experts to gain a ground truth. For each sample d_j^k, the initial and the updated model are used to infer a probability distribution over the intentions $P\left(I_j^k | \boldsymbol{o}_j^{1:k}\right)$ and behaviours $P\left(B_j^k | \boldsymbol{o}_j^{1:k}\right)$ given all available sensory input in the corresponding time-series up to this sample. The output of the model is then defined as the most probable intention

$$i_{j,out}^k = \arg\max_{\mathrm{i}} P\left(I_j^k = i | \boldsymbol{o}_j^{1:k}\right)$$

and behaviour

$$b_{j,out}^k = \arg\max_{\mathrm{b}} P\left(B_j^k = b | \boldsymbol{o}_j^{1:k}\right).$$

Since the Driver Intention Recognition model delivers as intention the desired target lane the ground truth lane i_j^k and the predicted target lane from the model $i_{j,out}^k$ are mapped to actual lane changes. Thus, only if the current lane and the target lane intentions differ, a lane change intention is present.

We match the lane change intentions from the model with the lane change intentions from the ground truth to construct a confusion matrix as shown

Figure 14. The existence of a lane change intention is counted as a positive and the absence as a negative.

		Ground Truth: Positive	Ground Truth: Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Figure 14: Binary confusion matrix to visualize model output vs annotated ground truth

Based on this table we can calculate accuracy metric which is defined as:

$$ACC = \frac{TP + TN}{TP + FP + FN + TN}.$$

4.3.1 Dataset for Validation

The mentioned data sets D_{Test}, D_{Train}, and D_{Train_Online} were obtained from the simulator study conducted for training and evaluation of the probabilistic driver models (for a detailed description of the experiment, please see D2.4 "Sensor Platform and Models incl. V&V results from 2nd cycle").

4.3.2 Results

Table 4 shows the ACC of the initial model and the updated model (after several amounts of learning steps D_{Train_Online}) on the test data set D_{Test}.

Table 4: Accuracy of the initial model and the updated model after several learning steps for evaluation on D_{Test}

	Initial model	Updated10	Updated20	Updated26
ACC	0.804	0.934	0.941	0.938

The ACC values of the updated model increase in comparison to the initial model and reach saturation. The updated model outperforms the initial model on the test set in terms of indicating more seldom a lane change when there is no intention to perform one, thus producing less false alarms than the initial model.

As shown in Table 5, the results validate the requirements R_EN4_model2.2 stating "The model must be able to learn (online) the driver's preferred decisions in specific situations" and R_EN4_model2.1 stating "The model must learn (online) to perform manoeuvres by adapting performance parameters to the driver behaviour".

This is reached due to the fulfilment of requirement R_EN4_model2.3 stating "The algorithms must be able to update the parameters of the driver model".

Table 5: Requirements and metrics used for the technical validation of E4.2

Requirement	**Metric**	**Success criteria**
The model must be able to learn (online) the	**ACC**	ACC of the updated model > ACC of the

driver's preferred decisions in specific situations		initial model
The model must learn (online) to perform manoeuvres by adapting performance parameters to the driver behaviour	**ACC**	ACC of the updated model > ACC of the initial model
The algorithms must be able to update the parameters of the driver model	**Check: Y / N**	Y, model parameters are updated if new data is provided

4.4 E5.1 - Online risk assessment (OFF + DLR + HTM)

4.4.1 Safety Corridor Around Dynamic Objects

Following the plans previously described in D3.4 "Metrics and plan for V&V of the concepts and algorithms in the 2nd cycle", validation of online risk assessment was performed using a set of independent test data D_{Test}, representing ground truth time-series of traffic situations.

To recapitalize the overall validation process and metric used, let D_{Test} be composed by a number of m trials, where each trial j, $j = 1, \dots, m$, is a time-series consisting of a number of n_j data samples $d_j^k = \left(\boldsymbol{x}_{v_1}^k, \dots, \boldsymbol{x}_{v_{n_V}}^k\right), k = 1, \dots, n_j$.

For each sample d_j^k, and each object $v \in \boldsymbol{V}$, we used the Vehicle and Situation Modelling Module to predict a sequence of future states $p\left(\boldsymbol{S}_{j,v}^{k+i\Delta} | E_{j,v}^k = true, \boldsymbol{o}^{1:k}\right), i = 1, \ldots, \eta_{max}$, and derived the region that included the expected position of the vehicle with a probability of $(1 - \delta_v)$, choosing δ_v such that $\delta_v = 1 - \sqrt[n_v]{(1-\delta)}$.

Based on this prediction, the online risk assessment component was used to calculate a corresponding set of safety corridors $\boldsymbol{c}^{k:k+\eta_{max}\Delta} = \left(\boldsymbol{c}^{k:k+\Delta}, \boldsymbol{c}^{k+\Delta:k+2\Delta}, \ldots, \boldsymbol{c}^{k+(\eta_{max}-1)\Delta:k+\eta_{max}\Delta}\right)$, For each safety corridor $\boldsymbol{c}^{i:i+1}, k \leq i < \eta_{max}$, we then used the current and subsequent samples in the trial corresponding to the resp. temporal interval $[i, i+1]$ and checked for each such sample, whether *any* object $v \in \boldsymbol{V}$ penetrated the implied safety region defined by the conjunction of the polygons.

Denoting such an occurrence as a failure and resp. as a success otherwise, we used the metric

$$CR_\delta^i = \frac{\#_s}{\#_s + \#_f},$$

representing *the ratio of successes* $\#_s$ *and the sum of successes* $\#_s$ *and failures* $\#_f$ for a prediction horizon i and a specific level of δ for assessing the quality of online risk assessment.

4.4.1.1 Dataset for Validation

The test set D_{Test} was obtained from the simulator study conducted for training and evaluation of the probabilistic driver models (for a detailed

description of the experiment, please refer to D2.4 “Sensor Platform and Models incl. V&V results from 2nd cycle”).

The scenario comprised approx. 30.8 km of a rural road track inspired by the Peter scenario with one lane for each direction and a primary speed limit of 100 km/h. Traffic flow in both directions consisted of trucks and passenger cars with varying driving speed. The vehicles on the right lane were driving with an average speed of 72km/h. The oncoming traffic instead drove at different speeds: trucks between 65-75 km/h and passenger cars between 70-105 km/h.

Participants had to manually traverse the scenario by controlling a simulated vehicle until reaching a parking side at the end of the track. Participants were instructed to follow the traffic rules, but were free to overtake lead vehicles, if deemed necessary or desired. Each participant had to absolve a total of three trials, with the overall traffic conditions adapted between trials to encourage overtaking manoeuvres.

In total, 18 subjects participated the study; one of them experienced motion sickness in the very beginning. Thus, we ended up with valid data from 17 subjects. Subjects were 28 years old (SD= 7,1) on average (9 male and 8 female). Participants were licensed on average for 10 years (SD=6,6), and drove 16941km/year on average (SD= 13961). Post-analysis of the data resulted in the exclusion of the data of another participant, due to inconsistencies in the data recording.

The first and third trial of each participant was provided as experimental data for training of the models for intention recognition, while the second trial of

each participant was reserved as a general test set D_{Test} for validation of the models for intention recognition, the prediction of the temporal and spatial evolution of the traffic scene, and online risk assessment.

The experimental data was recorded with a frequency of 60Hz. As the component for online risk assessment internally works with a frequency of 20 Hz, we treat each time-series as information sequentially provided by the sensor and communication platform, and only use every third sample for the actual validation. As such, the test set D_{Test}, effectively consists of 406262 samples used for validation, covering approx. 338 minutes of simulated driving behaviour in the Peter scenario.

As stated in D3.4 “Metrics and plan for V&V of the concepts and algorithms in the 2nd cycle”, we wrongly believed the performance of online risk assessment to be upper bounded by the performance of the prediction of the temporal and spatial evolution of the traffic scene, and it was therefore planned to only use samples, for which the prediction of the temporal and spatial evolution of the traffic scene was correct.

As an adjustment to these plans, we use all samples $d \in D_{Test}$ and provide the performance of the prediction of the temporal and spatial evolution of the traffic scene as a comparison.

Due to the test set arising from a simulator study in which the traffic flow was automatically controlled by a traffic simulation, the resulting behaviour of traffic participants in the vicinity of the TeamMate vehicle is highly predictable and therefore potentially unrealistic. As a means to provide a more realistic assessment for humanly controlled traffic participants, we

additionally perform our validation on the safety regions for the humanly controlled “TeamMate” vehicle.

4.4.1.2 Results

We performed the validation, using a temporal step width $\Delta = 1s$ and a maximal number of steps $\eta_{max} = 10$, resulting in a prediction horizon of $\eta_{max}\Delta = 10s$, for five different levels of δ, $\delta_{0.5} = 0.5$, $\delta_{0.25} = 0.25$, $\delta_{0.1} = 0.1$, $\delta_{0.05} = 0.05$, and $\delta_{0.01} = 0.01$, expecting a ratio of $1 - \delta$ respectively. As requirement R_EN_model1.5 requires a correct rate of classification above 90% to be fulfilled, we define the requirement to be fulfilled for a specific temporal interval i and level of δ, when

$$CR_{\delta}^{i} > 0.9(1 - \delta).$$

Table 6 shows the result for surrounding traffic participants, for different temporal intervals $\boldsymbol{i}$ (corresponding to a temporal interval $[\boldsymbol{k} + (\boldsymbol{i} - \mathbf{1})\boldsymbol{s}{:}\,\boldsymbol{k} + \boldsymbol{is}]$) and different levels of $\boldsymbol{\delta}$. Bold values indicate that the ratio is above $\mathbf{0.9}(\mathbf{1} - \boldsymbol{\delta})$, therefore fulfilling R_EN_model1.5. For comparison, values in brackets represent the mean ratio of the spatial and temporal evolution for the start and end of the temporal interval. Bold values in brackets indicate that the ratio for both the start and end of the temporal interval are above $\mathbf{0.9}(\mathbf{1} - \boldsymbol{\delta})$. As apparent, the ratios are, for the most part, above the expected ratios of $1 - \delta$ and therefore above $0.9(1 - \delta)$. the Although a promising result, this mainly results from the highly predictable behaviour of the automatically controlled vehicles.

Table 6: Ratio of successes $\#_s$ and the sum of successes $\#_s$ and failures $\#_f$ for online risk assessment.

i	$\#_s + \#_f$	$CR^i_{\delta_{0.5}}$	$CR^i_{\delta_{0.25}}$	$CR^i_{\delta_{0.1}}$	$CR^i_{\delta_{0.05}}$	$CR^i_{\delta_{0.01}}$
1	20745587	0.9990 (0.9613)	0.9998 (0.9747)	1.000 (0.9815)	1.000 (0.9847)	1.000 (0.9888)
2	20174665	0.9880 (0.9340)	0.9920 (0.9592)	0.9951 (0.9706)	0.9966 (0.9749)	0.9985 (0.9803)
3	19615692	0.9785 (0.9430)	0.9810 (0.9675)	0.9838 (0.9772)	0.9860 (0.9798)	0.9898 (0.9826)
4	19067387	0.9668 (0.9362)	0.9712 (0.9651)	0.9740 (0.9757)	0.9753 (0.9788)	0.9780 (0.9823)
5	18530737	0.9559 (0.9296)	0.9619 (0.9626)	0.9661 (0.9752)	0.9682 (0.9786)	0.9714 (0.9825)
6	18005944	0.9475 (0.9248)	0.9551 (0.9604)	0.9603 (0.9759)	0.9631 (0.9794)	0.9675 (0.9833)
7	17489618	0.9388 (0.9209)	0.9480 (0.9604)	0.9547 (0.9775)	0.9582 (0.9810)	0.9639 (0.9847)
8	16984356	0.9298 (0.9187)	0.9402 (0.9635)	0.9482 (0.9789)	0.9525 (0.9823)	0.9595 (0.9858)

9	16491437	0.9203 (0.9178)	0.9317 (0.9669)	0.9407 (0.9799)	0.9458 (0.9831)	0.9542 (0.9864)
10	16006904	0.9109 (0.9181)	0.9230 (0.9681)	0.9328 (0.9804)	0.9384 (0.9834)	0.9479 (0.9865)

To provide a potentially more accurate picture, Table 7 shows the result for the ego-vehicle. In contrast to the online risk assessment for the automatically controlled vehicles, we see that the requirement of $CR_{\delta}^{i} > 0.9(1-\delta)$ is only partly fulfilled. As expected, the performance of online risk assessment is tightly coupled with the performance of the prediction of the temporal and spatial evolution of the traffic scene, in this case the future behaviour of the TeamMate vehicle. Interestingly, the performance of online risk assessment, for the most part, outperforms the prediction of the temporal and spatial evolution of the traffic scene.

Table 7: Ratio of successes $\#_s$ and the sum of successes $\#_s$ and failures $\#_f$ for online risk assessment

i	$\#_s + \#_f$	$CR_{\delta_{0.5}}^{i}$	$CR_{\delta_{0.25}}^{i}$	$CR_{\delta_{0.1}}^{i}$	$CR_{\delta_{0.05}}^{i}$	$CR_{\delta_{0.01}}^{i}$
1	8528142	0.9993 (0.9305)	0.9999 (0.9573)	1.000 (0.9729)	1.000 (0.9797)	1.000 (0.9877)
2	8521422	0.9634 (0.8258)	0.9818 (0.8854)	0.9901 (0.9202)	0.9931 (0.9358)	0.9967 (0.9557)

3	8514702	0.9133 (0.7602)	0.9458 (0.8344)	0.9609 (0.8788)	0.9671 (0.8981)	0.9770 (0.9243)
4	8507982	0.8712 (0.7103)	0.9136 (0.7958)	0.9342 (0.8495)	0.9414 (0.8717)	0.9523 (0.9027)
5	8501262	0.8295 (0.6763)	0.8801 (0.7677)	0.9050 (0.8241)	0.9148 (0.8487)	0.9288 (0.8842)
6	8494542	0.7857 (0.6513)	0.8446 (0.7481)	0.8741 (0.8041)	0.8862 (0.8303)	0.9031 (0.8683)
7	8487822	0.7432 (0.6338)	0.8069 (0.7347)	0.8422 (0.7915)	0.8566 (0.8174)	0.8762 (0.8549)
8	8481102	0.7026 (0.6229)	0.7690 (0.7271)	0.8090 (0.7840)	0.8262 (0.8086)	0.8494 (0.8443)
9	8474382	0.6632 (0.6163)	0.7319 (0.7219)	0.7753 (0.7779)	0.7948 (0.8018)	0.8225 (0.8355)
10	8467662	0.6250 (0.6091)	0.6955 (0.7158)	0.7421 (0.7711)	0.7636 (0.7953)	0.7954 (0.8279)

As shown in Table 8, results are used to assess the fulfilment of requirements R_EN5_model1.1 and R_EN_model1.5 stating that the "*online risk assessment must be able to calculate a context-dependent safety*

corridor based on a set of pre-defined metrics" (R_EN5_model1.1) and that the "*online risk assessment must determine the safety level of a planned trajectory based on a set of pre-defined metrics*" with a correct rate of classification above 90% to be fulfilled (R_EN_model1.5).

Table 8: Requirements and metrics used for the technical validation of E5.1

Requirement	**Metric**	**Success criteria**
Online risk assessment must be able to calculate a context-dependent safety corridor based on a set of pre-defined metrics	**Correct rate (CR) of classification**	CR > 90%
Online risk assessment must determine the safety level of a planned trajectory based on a set of pre-defined metrics	**Correct rate (CR) of classification**	CR > 90%

As it was expected that the quality of online risk assessment decreases with an increasing prediction horizon $\eta_{max}\Delta$, we report the fulfillment of the requirements up to a highest achieved prediction horizon. Based on the results for the TeamMate vehicle (Table 6) and surrounding traffic

participants (Table 7), we conclude the fulfilment of R_EN_model1.5 (for the used dataset D_{Test}) up to a prediction horizon of $\eta_{max}\Delta = 7s$ for $\delta_{0.01} = 0.01$ and the fulfilment of R_EN5_model1.1 in general.

4.4.2 Safety Corridor Between Road Boundaries

For the validation of the safety corridor between road boundaries, we used the *Intersection Over Union* metric

$$IOU = \frac{TP}{TP + FP + FN}$$

where TP, FP and FN are respectively the true positive, false positive and the false negative area. In Figure 15, the ground truth safety corridor L_{R_GT} is represented as green polyline. The estimated safety corridor L_{R_est} (see red polyline) deviates from the ground truth due to the ego-vehicle pose uncertainty in the lateral direction. Therefore, the true positive area is the intersection of L_{R_GT} and L_{R_est} area. The false negative area is L_{R_GT} area which is not covered by L_{R_est}. The false positive area corresponds to L_{R_est} region which is not overlapping with the ground truth.

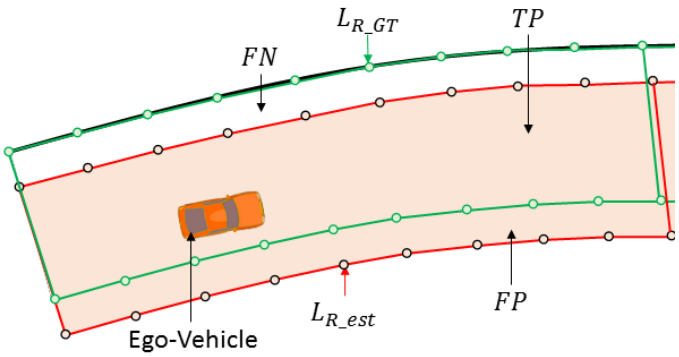


Figure 15: Ground truth (green) and estimated safety corridor (red) from road boundaries

To generate the IOU metric mentioned above, we are currently developing a comprehensive test procedure, with results from the procedures expected to be achieved at the beginning of the integration phase.

4.4.2.1 Dataset for Validation

The first dataset will be a synthetic dataset. This is a necessary precursor to the use of data from the real driving situations, in order to understand the algorithm's behaviour in a targeted variation of certain parameters. Another important reason is the possibilities of errors and certain erratic behaviour in the sensors being used for the recording of the real driving data. As an example, the ego vehicle position must be as accurate as possible, while the positioning of test vehicles on real roads often is fraught with inaccuracies.

The generation of the synthetic data will be specified with a test case definition. These test cases are produced by combining dimensions of interest, such as the manoeuvres, the velocity or accelerations of traffic objects in certain intervals.

Together with an open-drive-description in a .xodr-format the enabler computes the semantically enriched data. As this will be done across a number of test dimensions, a high number of varying test cases can be covered. This approach is pictured in Figure 16.

Eventually, real data will have to be used, since the validation data must be sampled from the module's target environment. This data must be very carefully pre-processed to exclude errors such as measurement errors. These types of flaws in the data are very common and can easily lead to false true / positives in the validation process.

The data therefore has to be inspected for errors, smoothed etc. More importantly, the labels for the ground truth necessary to compare predicted values with the real values usually do not exist and must be generated partially by hand.

An overview over the advantages and disadvantages of validation data is given in Table 9.

Table 9: Overview over approaches to acquire data for validation.

data	origin	advantage	disadvantage
synthetic test data	automatically generated	- any combination of possible test cases can be computed - complete control over experimental situation - no measurement errors - perfect repeatability - labels for the ground truth are known - cheap	- only what is being put in is being simulated; perhaps important of the real world which were not considered are not part of the validation environment
real test data	drives in the real world, on testing grounds or	- no transfer from the testing to the target environment	- expensive - much work with data processing until a test data

	public roads	necessary - do not have to be generated artificially	set is available - measurement errors such as for positioning - labels for the ground truth must be generated

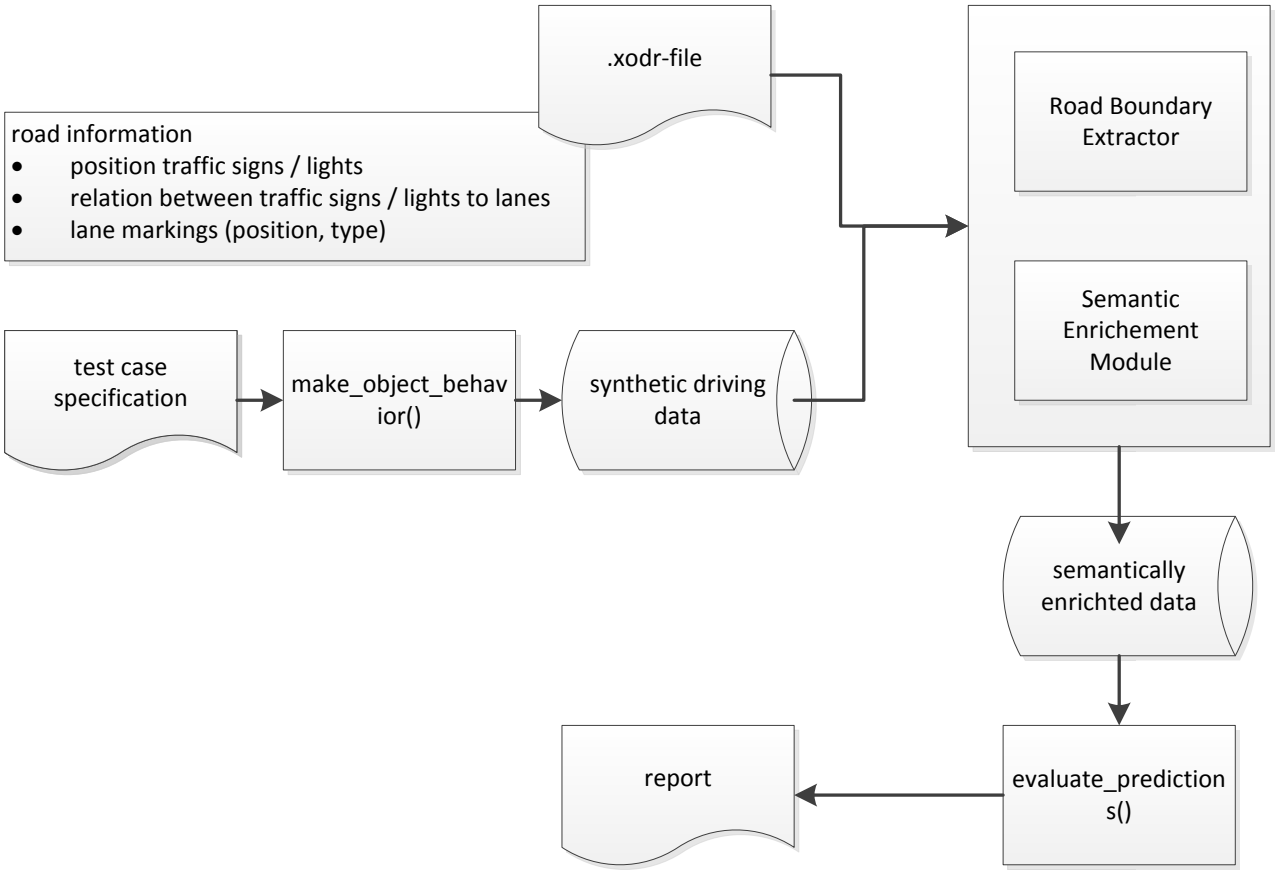


Figure 16: Enabler validation with experiments using synthetic data.

4.4.2.2 Results

As indicated above, the most important metric for the Road Boundary Extractor will be IOU (*intersection over union*), which can be easily computed for the synthetic data. For the real data, the quality of the metric relies on the accuracy of the vehicle positioning, which in turn will have to be taken into account. The computation of the metric itself can be achieved using the distances between the polylines at sampling indices, or with libraries able to operate on polygons such as Python's Shapely.

5 Conclusions

All enablers in WP3 (E4.1, E4.2 and E5.1) have been developed to implement the A2H cooperation, both in action and in perception.

The activities at this stage of WP3 have led to a marked advancement for Enabler 4 (Adaptive Driving Manoeuvre Planning, Execution and Learning) and Enabler 5 (Online Risk Assessment).

The planning and execution of trajectories have progressed significantly and are on a good way. In case this approach to trajectory planning (developed by ULM) should prove to be not as successful as required by safety or other criteria, another trajectory planner (developed by VED) exists both as a backup solution and a baseline.

The module to estimate driver intention has been greatly improved to handle continuous data, advancing the possibility to apply it to online driving data.

Finally, the concept for the online risk assessment has been much clarified. It will consist of modules computing safe driving zones between lane boundaries and away from other objects. Both sub-modules do exist in first versions and are being validated, with the final validation results expected at the beginning of the integration phase.

6 References

[1] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for Bertha—A local, continuous method,” in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, 2014, pp. 450–457.

[2] D. González and J. Pérez, “Control architecture for cybernetic transportation systems in urban environments,” in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, 2013, pp. 1119–1124.

[3] J. P. Rastelli, R. Lattarulo, and F. Nashashibi, “Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles,” in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, 2014, pp. 510–515.

[4] L. Han, H. Yashiro, Nejad, Hossein Tehrani Nik, Q. H. Do, and S. Mita, “Bezier curve based path planning for autonomous vehicle in urban environment,” in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, 2010, pp. 1036–1042.

[5] F. A. Sohel, G. C. Karmakar, L. S. Dooley, and J. Arkinstall, “Enhanced Bezier curve models incorporating local information,” in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP’05). IEEE International Conference on*, 2005, pp. iv-253.

[6] K. Yang and S. Sukkarieh, “An analytical continuous-curvature path-smoothing algorithm,” *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010.

[7] J.-w. Choi, R. E. Curry, and G. H. Elkaim, “Real-time obstacle avoiding path planning for mobile robots,” in *Proceedings of the AIAA Guidance, Navigation and Control Conference, AIAA GNC*, 2010.

[8] D. J. Walton, D. S. Meek, and J. M. Ali, “Planar G 2 transition curves composed of cubic Bézier spiral segments,” *Journal of Computational and Applied Mathematics*, vol. 157, no. 2, pp. 453–476, 2003.

[9] J. A. Bilmes and others, “A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models,” *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[10] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*: MIT press, 2009.